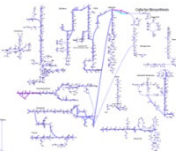




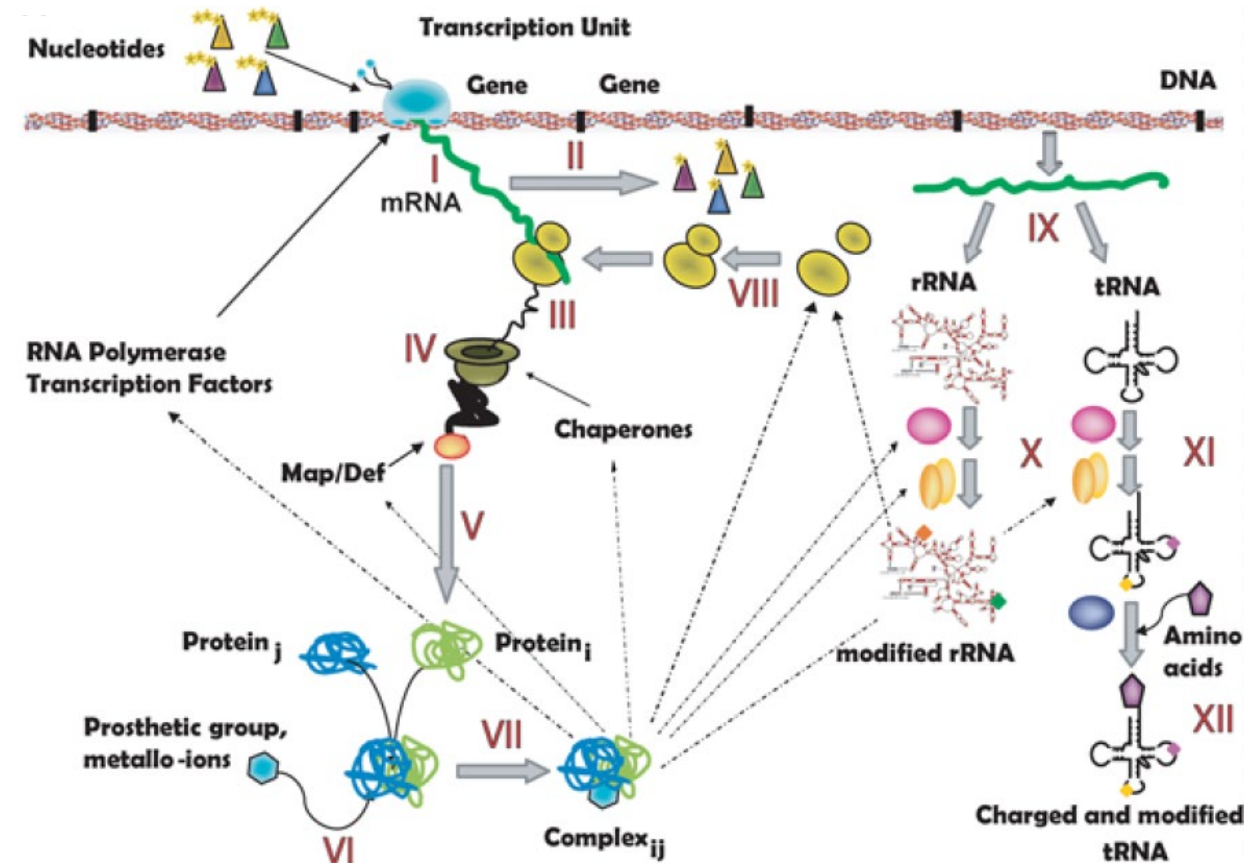
Understanding COBRAmc & ECOLImc



Lesson Outline

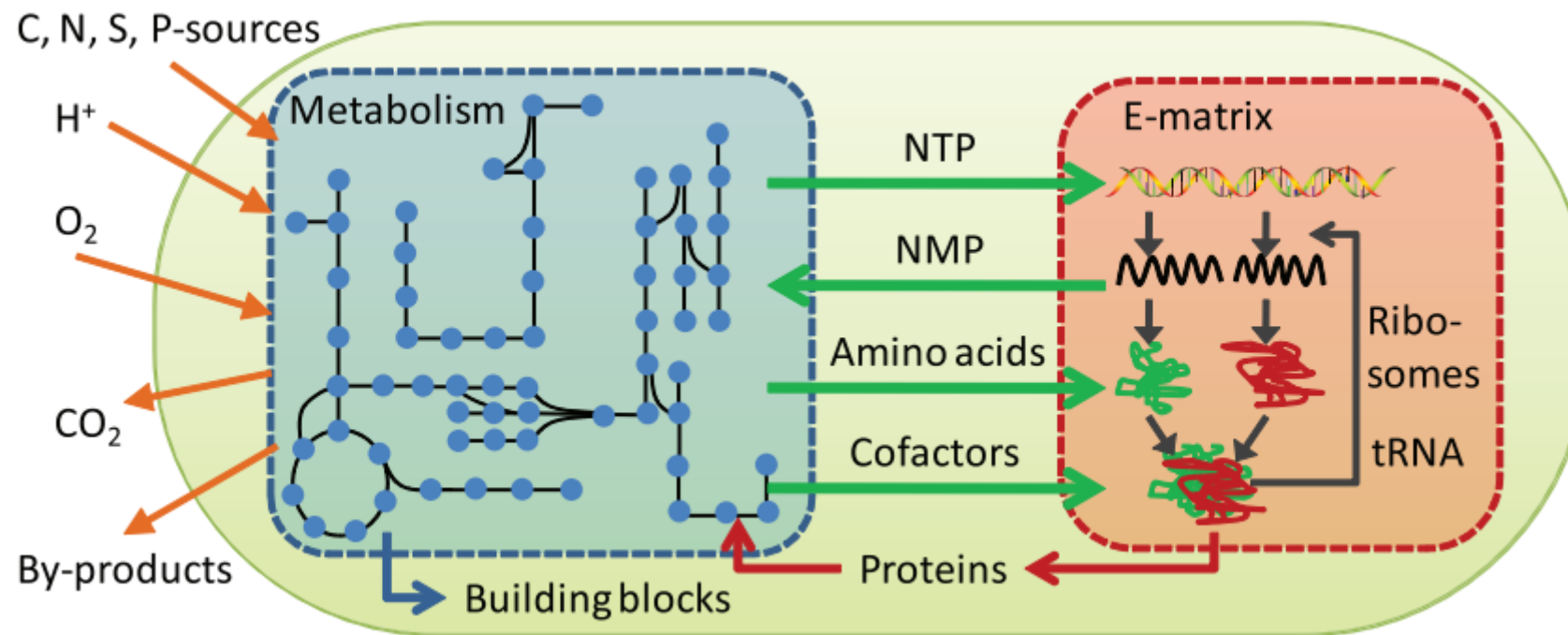
- ➔ • Metabolism (M) vs. Metabolism & Expression (ME) Models
 - COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
 - ECOLIme
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRAMe Execution
 - ECOLIme Operation

Metabolism and Expression (ME) Model



Thiele I, Jamshidi N, Fleming RMT, Pálsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312"

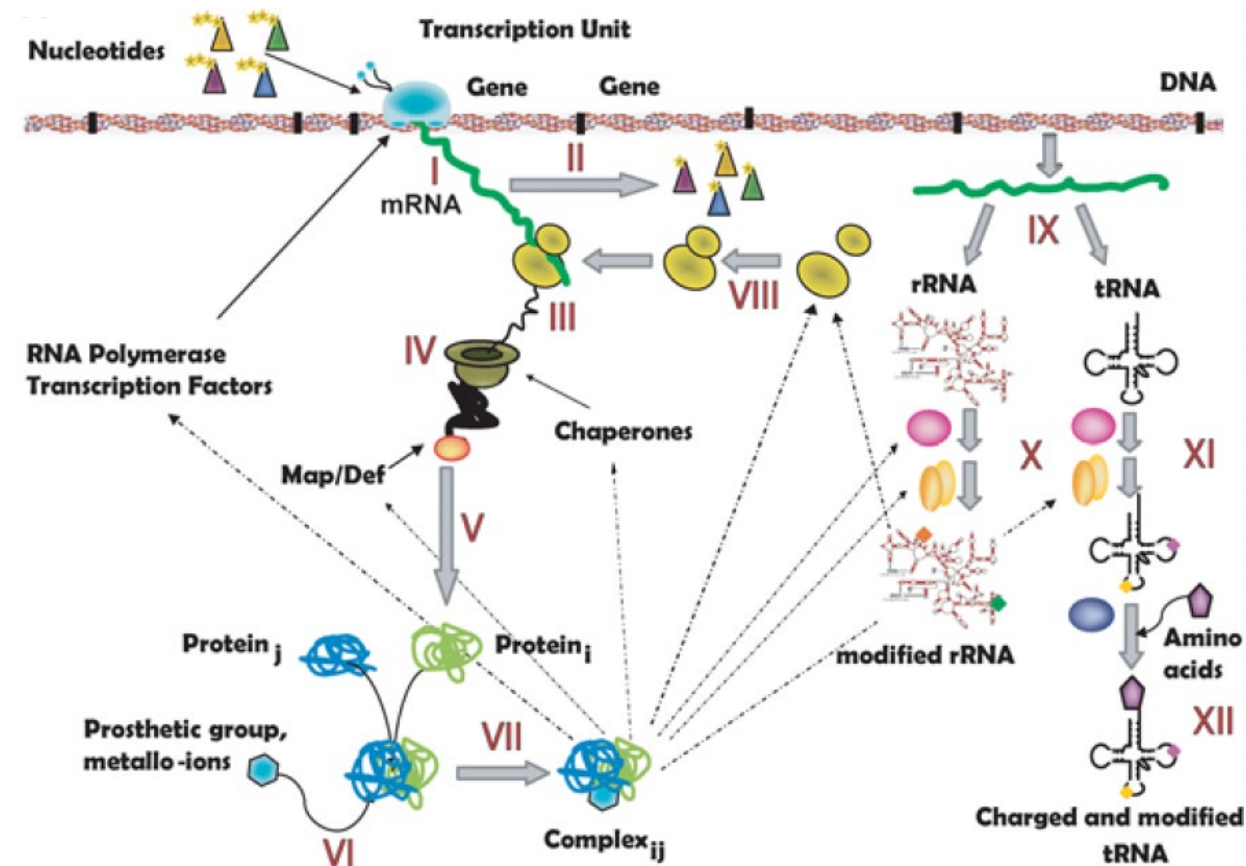
Functional Synergy Between Metabolism and Macromolecular Synthesis



Thiele I, Fleming RMT, Que R, Bordbar A, Diep D, Palsson BO (2012) Multiscale Modeling of Metabolism and Macromolecular Synthesis in E. coli and Its Application to the Evolution of Codon Usage. PLoS ONE 7(9)

M vs ME Constraint-based Models

- Metabolism models (M-models) can be extended to include the synthesis of the gene expression machinery which enables models to compute the entire metabolic and gene expression proteome in a growing cell.
- ME-models integrate Metabolism and Expression on the genome scale, and are capable of computing a large percentage of the proteome by mass.
- ME-models not only compute optimal metabolic flux states, as with M-models, but they additionally compute the optimal proteome composition required to sustain the metabolic phenotype.
- COBRaME, a software tool for creating ME models.
- ECOLIme is a tool to create the COBRaME-based ME-model of *E. coli*.



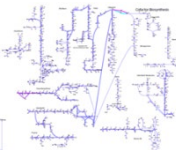
Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRaME: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7): e1006312. doi:10.1371/journal.pcbi.1006312

Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312



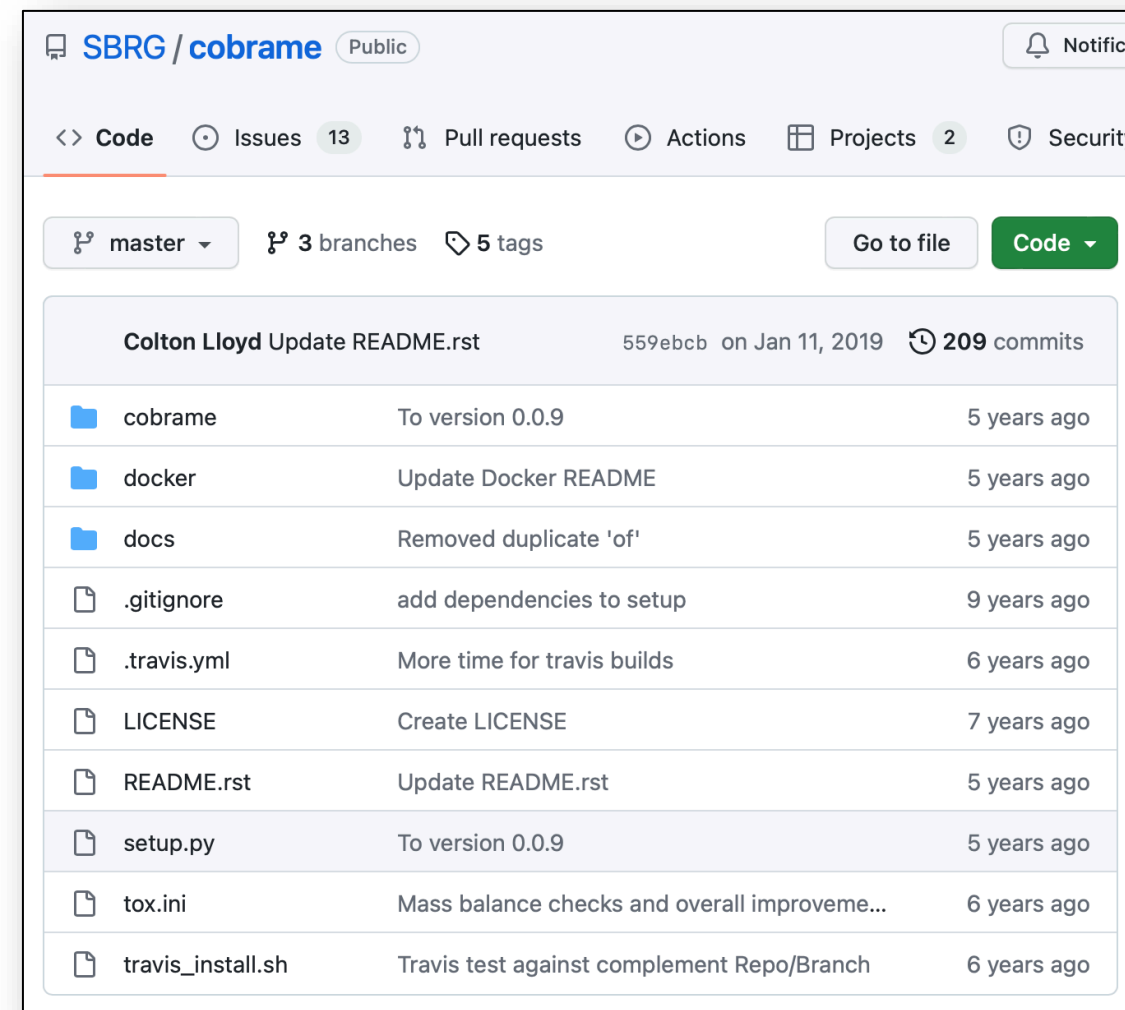
Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- • COBRaME
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLiME
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRaME Execution
 - ECOLiME Operation

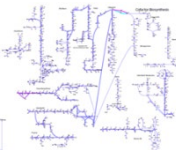


COBRame

- COBRame, a software tool for creating ME models, is designed to:
 1. support any organism with an existing M-model;
 2. use protocols and commands familiar to current users of COBRapy;
 3. represent ME-models with an intuitive collection of Python classes;
 4. solve FBA simulations orders of magnitude faster than previous ME-models
- COBRame is written in Python and extends the widely used COBRapy software that only supports M-models.
- COBRame was developed by C. J. Lloyd *et al* from the Bernard Palsson's group at UCSD.
- COBRame publication: "Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRame: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7)"



<https://github.com/SBRG/cobrame>



COBRaMe vs ECOLIme

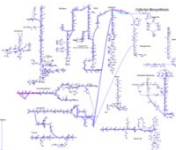
- The ME-model of *E. coli* is reconstructed using the two Python packages: COBRaMe and ECOLIme.
- COBRaMe contains the class definitions and necessary methods to facilitate building and editing a working ME-model. It supplies the computational framework underlying the ME-model
- COBRaMe is written to be organism-agnostic so that it can be applied to ME-models for any organism.
- ECOLIme contains the *E. coli* specific information (e.g., the *E. coli* ribosome composition) as well as functions required to process files containing *E. coli* reaction information (e.g., the text file containing transcription unit definitions) and associate them with the ME-model being constructed.
- ECOLIme is required to assemble the reaction and gene expression information that comprises iJL1678b-ME.
- The package composition along with further demonstrations of the utility of each of these packages is outlined in the COBRaMe Documentation.

Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRaMe: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7): 1–15.

The screenshot shows the GitHub repository for SBRG/ecolime. The repository is public and has 3 issues, 1 pull request, and 1 project. The master branch is selected. The file list includes:

File	Description	Commit Date
ecolime	To version 0.0.9	5 years ago
.gitattributes	Python 3.x compatibility and simple testing	6 years ago
.gitignore	Add AA by sequence not as subreaction data	7 years ago
.travis.yml	Improve scripts to output model stats	6 years ago
LICENSE	Create LICENSE	7 years ago
README.md	Rename to iJL1678b	6 years ago
setup.py	To version 0.0.9	5 years ago
tox.ini	Add me_models to package data	6 years ago
travis_install.sh	Travis test against complement Repo/Branch	6 years ago

<https://github.com/SBRG/ecolime>



Docker/GitHub Working Environment

1. Start Docker Desktop
2. In a Windows terminal window load the COBRame Docker image (converts Windows terminal window to Linux window)

```
docker run -p 8888:8888 --rm -i -t sbrg/cobrame:master bash
```

3. In a Docker terminal window clone the GitHub repository (associated with the "Container" page of Docker Desktop)

```
git clone https://github.com/hscotthinton/me_files
```

4. Identify the token needed for the Jupyter notebook from the Docker's new created "bash" window

```
jupyter notebook --ip=0.0.0.0 --port=8888
```

5. Click the port link on the Docker "Container" page to start the Jupyter notebook and enter the token

```
scotthinton - com.docker.cli - docker run -p 8888:8888 --rm -i -t sbrg/cobrame:master bash - 110x24
total 8
 4 drwxr-xr-x   2 meuser  users      4096 Jan 11  2019 me_models
 4 -rw-r--r--   1 meuser  users      2993 Jan 11  2019 solve_demo.ipynb
bash-4.3$ jupyter notebook --ip=0.0.0.0 --port=8888
[I 16:35:01.826 NotebookApp] Writing notebook server cookie secret to /home/.local/share/jupyter/runtime/notebook_cookie_secret
[I 16:35:02.001 NotebookApp] Serving notebooks from local directory: /home/meuser
[I 16:35:02.001 NotebookApp] 0 active kernels
[I 16:35:02.001 NotebookApp] The Jupyter Notebook is running at:
[I 16:35:02.001 NotebookApp] http://1f5632914f82:8888/?token=65c476b49b2e05b80e1bd775c5b4d505dc94dfa9d06ed1c7
[I 16:35:02.001 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[W 16:35:02.001 NotebookApp] No web browser found: could not locate runnable browser.
[C 16:35:02.002 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://1f5632914f82:8888/?token=65c476b49b2e05b80e1bd775c5b4d505dc94dfa9d06ed1c7&token=65c476b49b2e05b80e1bd775c5b4d505dc94dfa9d06ed1c7
```

Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another.

☐ Only show running containers

<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)
<input type="checkbox"/>	stoic_kepler 1f5632914f82	sbrg/cobrame:master	Running	8888:8888



Docker Development Environment

1. Start Docker Desktop
2. Open a terminal window and move to the shared drobox folder `C:\Users\hinton\Dropbox\me-share`
3. Load the COBRame Docker image (converts Windows "terminal" window to Linux "bash" window)
`docker run -p 8888:8888 --rm -it -v ${PWD}:/home/meuser/me-share/ -t sbrg/cobrame:master bash`
4. Using the Linux Bash window, identify the Jupyter security token with
`jupyter notebook --ip=0.0.0.0 --port=8888`
5. Click the port link on the Docker "Container" page to start the Jupyter notebook



6. After the Jupyter window opens and asks for the token, enter the token generated from the previous command

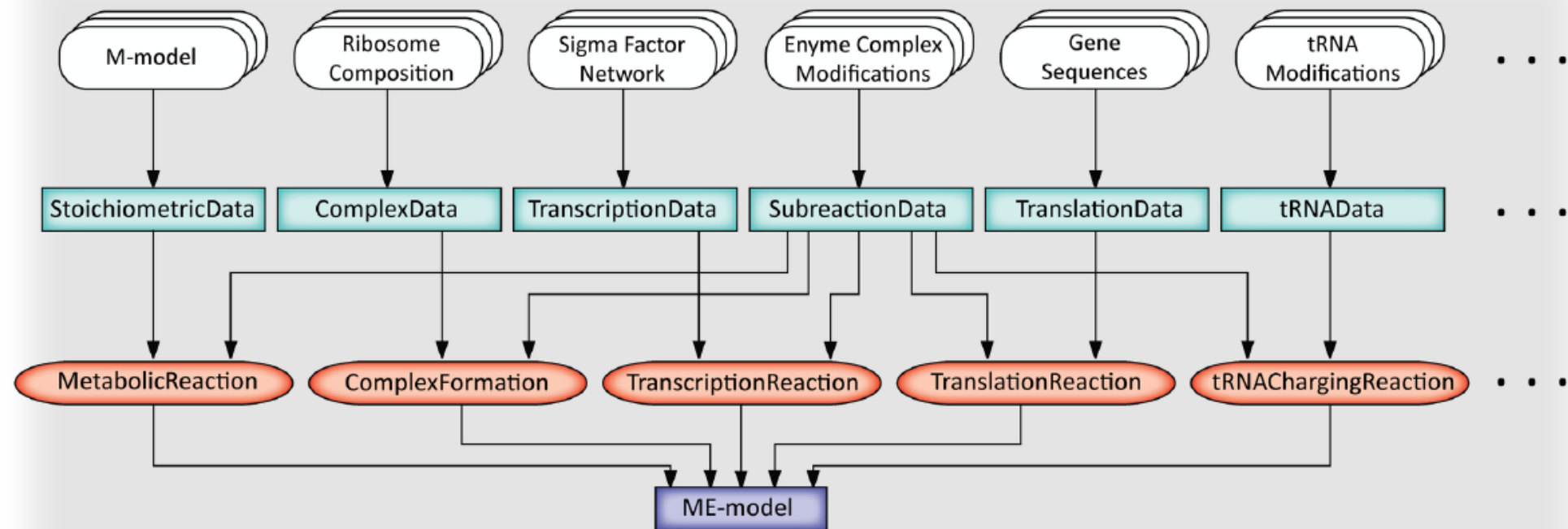


Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - • ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRAMe Execution
 - ECOLIme Operation

Flow of Information from Input Data to the ME-Model

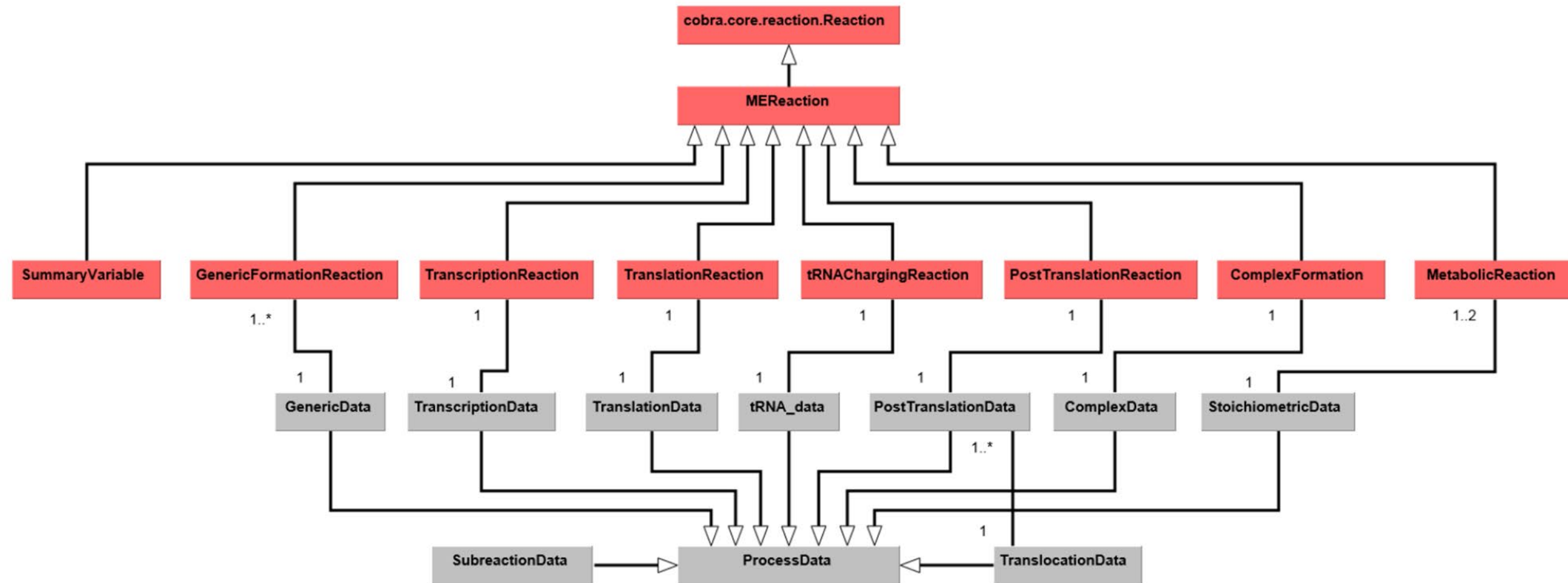
- The 'build_me_model' workflow uses the ECOLIme package to load and process the *E. coli* M-model along with all supplied files containing information defining gene expression processes/reactions (white ovals).
- This information is then used to populate the different ProcessData classes (turquoise boxes) and link them to the appropriate ME Reaction classes (red ovals), all of which are defined in the COBRAME package.
- The entirety of the ME Reactions comprise a working ME-model.



Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRAME: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7): e1006111.

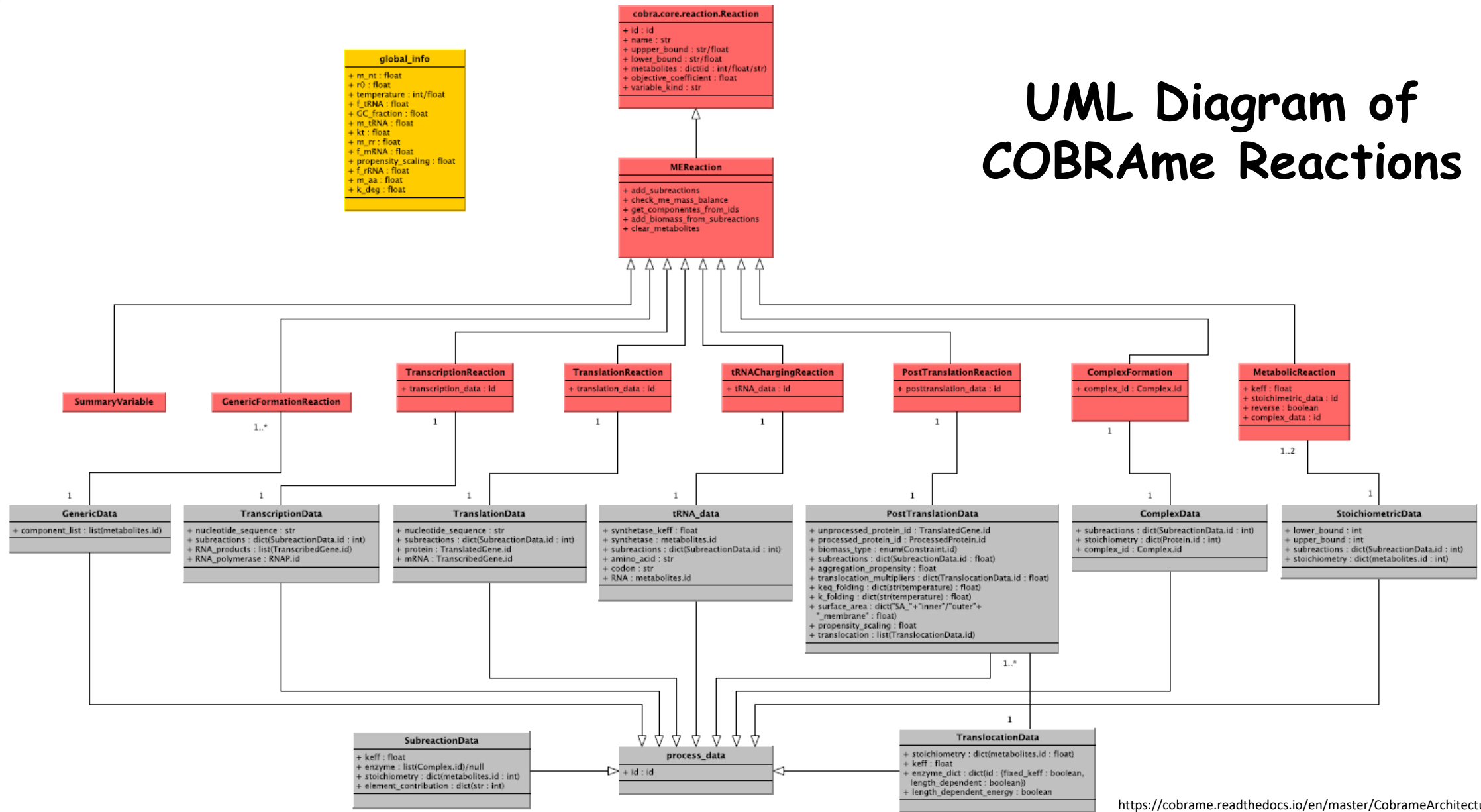


UML Diagram of COBRaMe Architecture (Reactions)

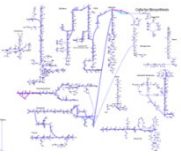


<https://cobrame.readthedocs.io/en/master/CobrameArchitectureOverview.html>

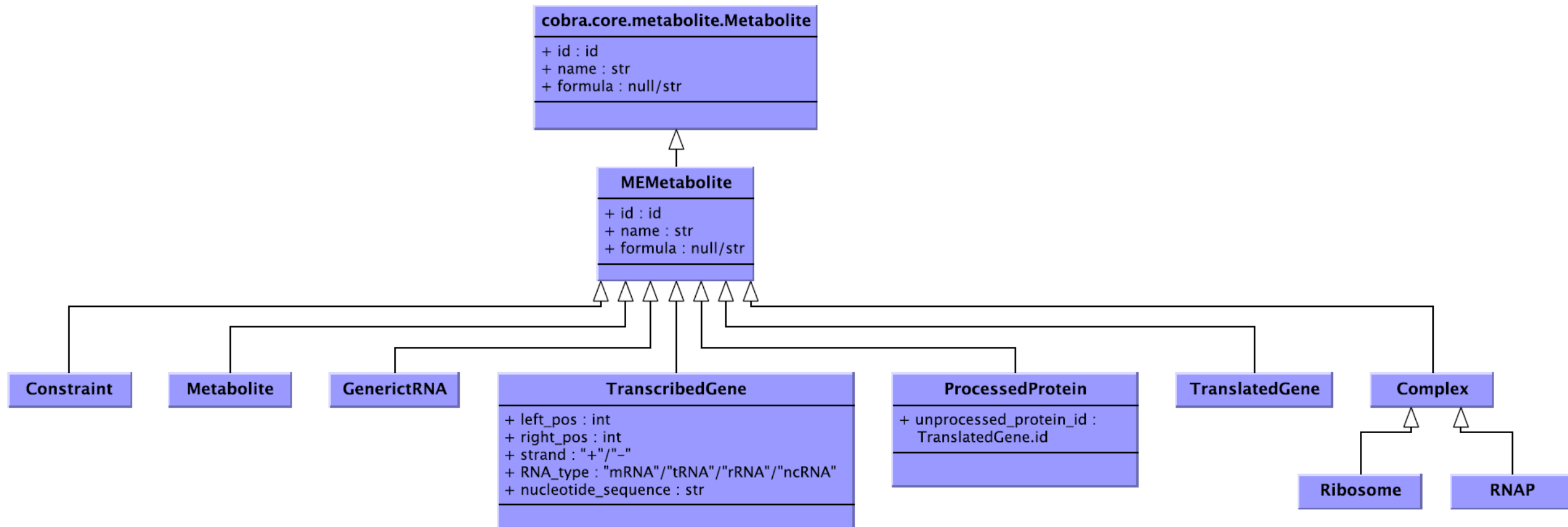
UML Diagram of COBRaMe Reactions

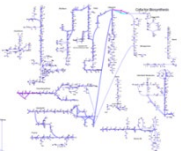


<https://cobrame.readthedocs.io/en/master/CobrameArchitectureOverview.html>



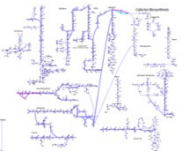
UML Diagram of COBRaMe Metabolites





Classes of ME-Model Metabolites

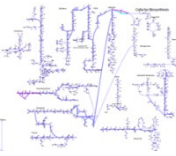
- COBRAme supports 10 metabolite classes, they include:
 1. GenericComponent - Broadly Used Metabolites
 2. ProcessedProtein - Lipoprotein
 3. GenericRNA - Charged tRNA
 4. Constraint - Biomass Metabolites
 5. Complex - Complex Metabolites
 6. Metabolite - M-Model Metabolites
 7. RNAP - RNA Polymerases
 8. Ribosome - Ribosomes
 9. TranscribedGene - mRNA, rRNA, ncRNA, tRNA
 10. TranslatedGene - Protein peptide
- Most metabolites effectively behave the same as a typical COBRA metabolite. The main exception is the "TranscribedGene" metabolite type which contains additional sequence information about the RNA it represents. In addition, each metabolite includes the types of ME Reactions that they participate in.
- The COBRAme compartments include: "p"="Periplasm", "e"="Extra-organism", "c"="Cytosol", "im"="Inner Membrane", "om"="Outer Membrane", "mc"="ME-model Constraint", "m"="Membrane"!
- See the online documentation (<https://cobrame.readthedocs.io>) and look at the ME metabolites table (S7) included in the supplementary material (<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006302>)



ME-Model Metabolite Classes

Metabolite Class	Purpose	Primary Producing Reaction Types	Primary Consuming Reaction Types
GenericComponent	Broadly Used Metabolites	GenericFormationReaction	ComplexFormation, ...
ProcessedProtein	Lipoprotein	ComplexFormation	PostTranslationReaction
GenerictRNA	Charged tRNA	tRNAChargingReaction	TranslationReaction
Constraint	Biomass Metabolites	TranscriptionReaction, ...	SummaryVariable
Complex	Complex Metabolites	MetabolicReactions, ...	ComplexFormation, ...
Metabolite	M-Model Metabolites	MetabolicReactions, ...	MetabolicReactions, ...
RNAP	RNA Polymerases	ComplexFormation	TranscriptionReaction
Ribosome	Ribosomes	ComplexFormation	TranslationReaction
TranscribedGene	mRNA, rRNA, ncRNA, tRNA	TranscriptionReaction	TranslationReaction, ...
TranslatedGene	Protein peptide	TranslationReaction	ComplexFormation, ...

Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRAME: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7): e1006111.



Classes of ME-Model Reactions

- COBRaME supports 9 classes of reactions, they include:
 1. MetabolicReaction - Create M-Model Metabolic Reactions
 2. GenericFormationReaction - Synthesize GenericComponents
 3. TranscriptionReaction - Synthesize Transcribed Genes
 4. tRNAChargingReaction - Synthesize Charged tRNA
 5. TranslationReaction - Synthesize Protein Peptides
 6. PostTranslationReaction - Synthesize Membrane Proteins
 7. ComplexFormation - Synthesize Protein Complexes
 8. SummaryVariable - Biomass Formation
 9. MEReaction - Create Demand/Exchange Reactions
- See the online documentation (<https://cobrame.readthedocs.io>) and look at the ME metabolites table (S7) included in the supplementary material (<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006302>)



ME-Model Reaction Classes

Reaction Class	Purpose	Primary Reactant Metabolite Classes	Primary Product Metabolite Classes
MetabolicReaction	Create M-Model Metabolic Reactions	Metabolite	Metabolite
GenericFormationReaction	Synthesize GenericComponents	Complex, ...	GenericComponent
TranscriptionReaction	Synthesize Transcribed Genes	RNAP, ...	TranscribedGene, ...
tRNAChargingReaction	Synthesize Charged tRNA	TranscribedGene, ...	GenerictRNA, ...
TranslationReaction	Synthesize Protein Peptides	Ribosome, TranscribedGene, Metabolite, GenerictRNA, ...	TranslatedGene, ...
PostTranslationReaction	Synthesize Membrane Proteins	TranslatedGene, Complex, ...	ProcessedProtein, ...
ComplexFormation	Synthesize Protein Complexes	TranslatedGene, ProcessedProtein	Complex
SummaryVariable	Biomass Formation	Constraint, ...	Biomass
MEReaction	Create Demand/Exchange Reactions	Metabolite	-

Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRAME: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7): e1006111. doi:10.1371/journal.pcbi.1006111



Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - • Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRAMe Execution
 - ECOLIme Operation

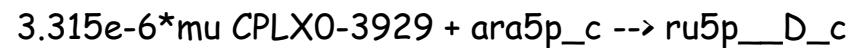
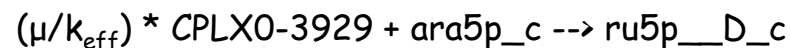
Coupling Constraints for Metabolic Reactions

- Coupling constraints are required in an ME-model in order to **couple the synthesis of a reaction's enzyme (macromolecule) flux to the value of the metabolic reaction flux**.
- The coupling of enzyme synthesis cost to metabolic reaction flux scales with the growth-rate (μ) to represent the dilution of enzymes as they are passed on the daughter cells.
- For a metabolic reaction, The flux of the catalyzing enzyme, v_{enzyme} , is calculated from

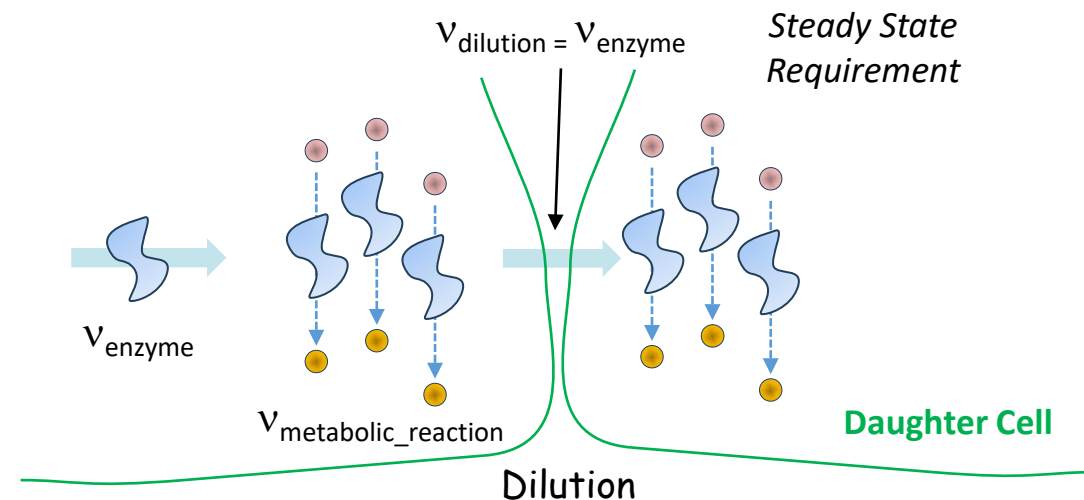
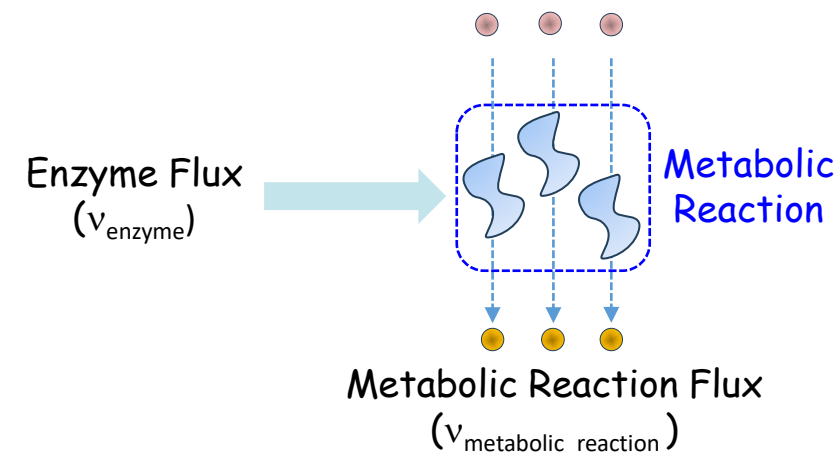
$$v_{\text{enzyme}} = (\mu/k_{\text{eff}}) * v_{\text{metabolic_reaction}}$$

where $v_{\text{metabolic_reaction}}$ is the flux passing through the metabolic reaction and k_{eff} is the effective turnover rate, which is the number of metabolic products the enzyme can catalyze per cell division.

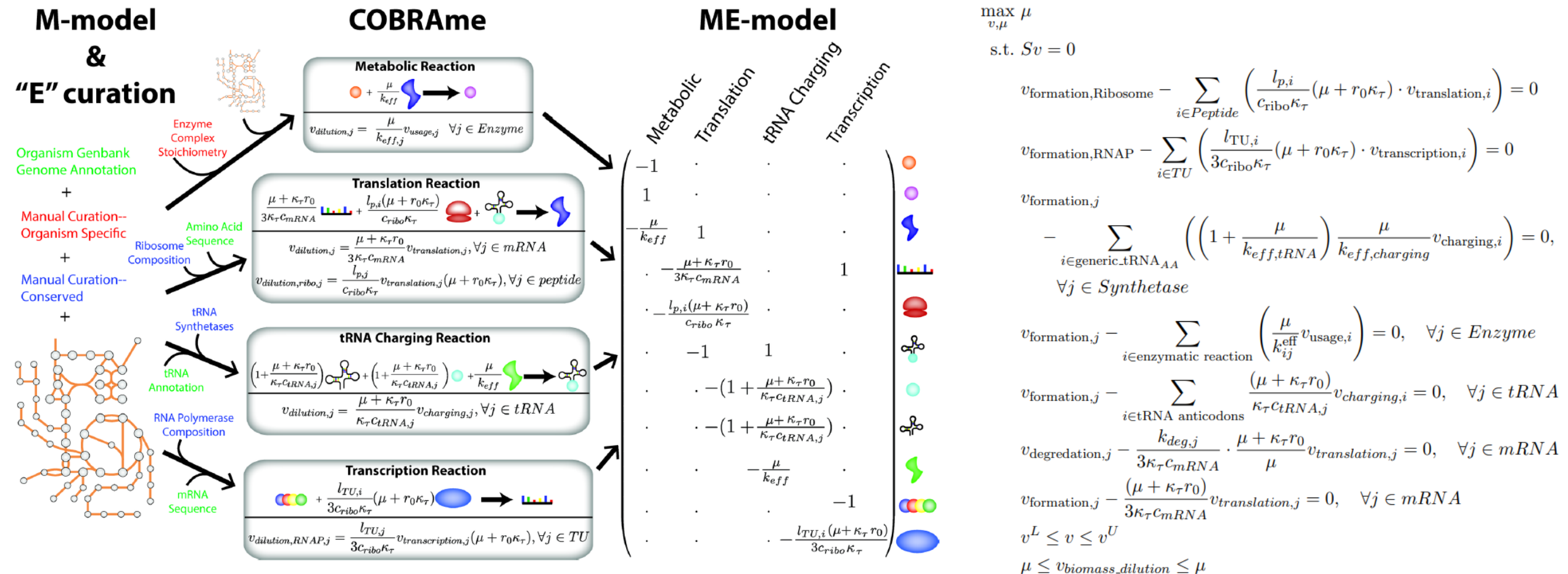
- With COBRaME the dilution of coupled macromolecules to the daughter cell is accounted for by applying a coupling coefficient directly in the reaction in which the macromolecule is used. Example,



Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRaME: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7):



COBRame Dilution and Optimization



Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRame: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7):

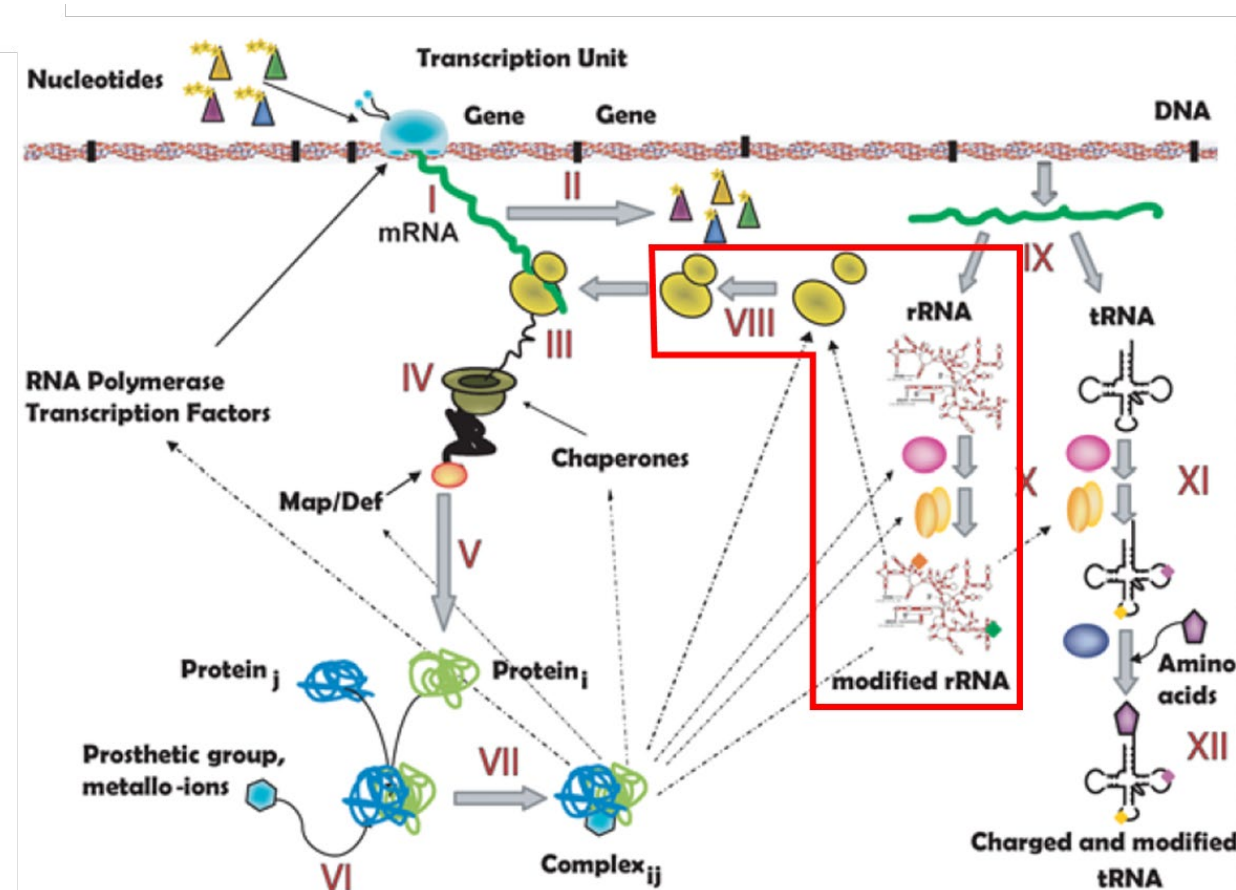


Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRaME
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - • Reaction Lumping
- ECOLiME
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRaME Execution
 - ECOLiME Operation

Reaction Lumping

- Splitting the model into ProcessData and ME Reactions allows for a variety of model simplifications. For example, reactions that occur in a number of individual steps or sub-reactions (i.e., ribosome formation, translation, etc.) can be lumped into a single reaction.
- The single lumped ME Reaction can be constructed by associating it with the multiple ProcessData instances that detail the individual sub-reactions involved in the overall reaction.
- All sub-reaction information is further accessible through the ME Reaction instance itself which allows the information to be queried, edited, and updated throughout the reaction.
- If the sub-reaction participates in many different reactions, the sub-reaction changes can be applied throughout the entire model.
- This lumping has the obvious benefit of reducing the number of model reactions, thus shortening the solve time.
- Lumping intricate reactions has the added benefit of making the ME-model much more modular in nature.
- This simplifies the process of adding or removing new processes associated with the ME-model reaction.



Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O'Brien EJ, et al. (2018) COBRAME: A computational framework for genome-scale models of metabolism and gene expression. PLoS Comput Biol 14(7): e1006312. doi:10.1371/journal.pcbi.1006312

Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312



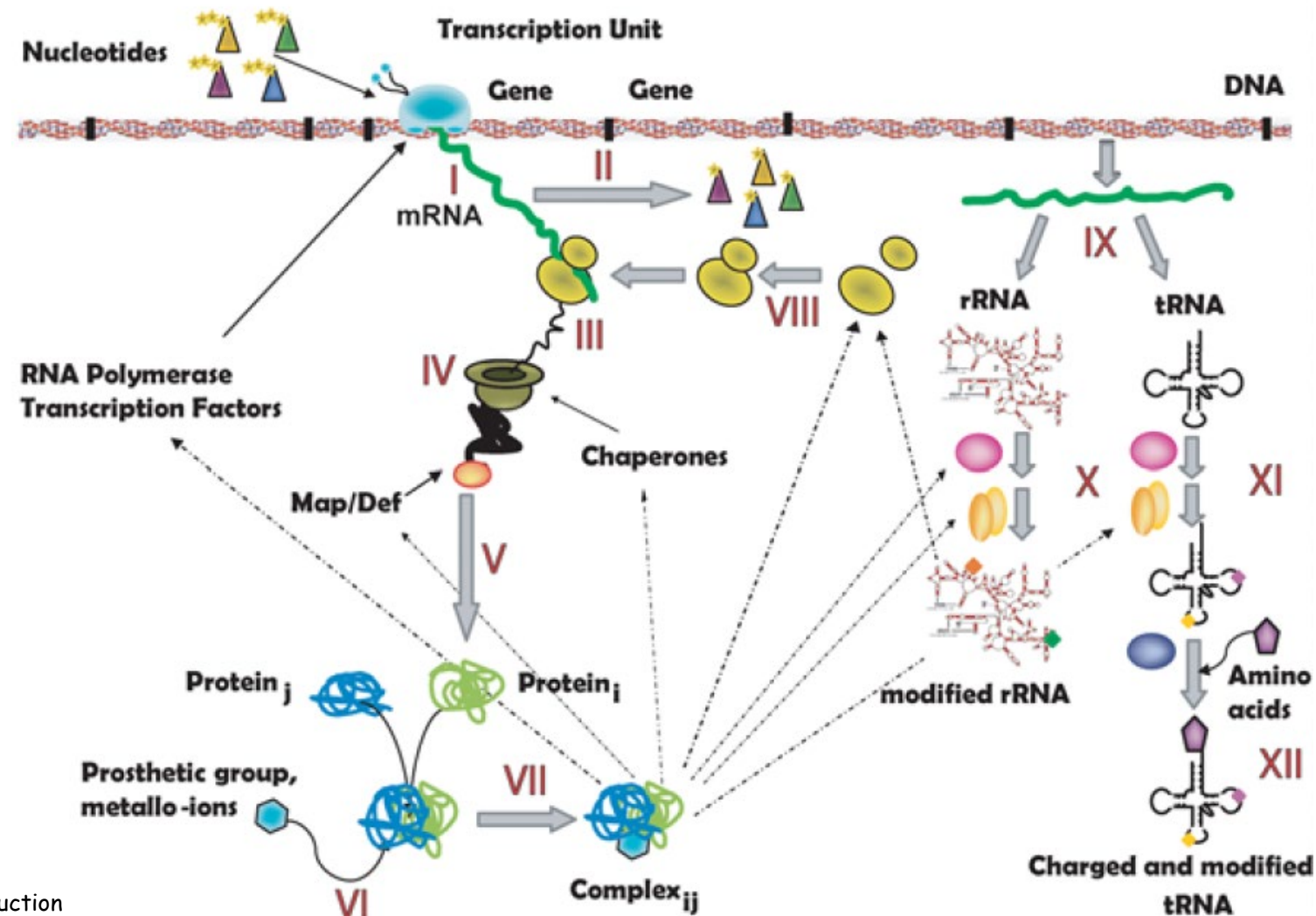
Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - ➔ • Macromolecular Reactions
 - Transcription
 - Translation
 - COBRAMe Execution
 - ECOLIme Operation

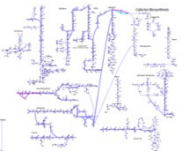
Expression Content

Schematic representation of the network components and reactions is shown. In addition to the macromolecular synthesis of RNA and proteins, rRNA and tRNA processing reactions were included in the reconstruction.

- I: transcription;
- II: mRNA degradation;
- III: translation;
- IV: protein maturation;
- V: protein folding;
- VI: metallo-ion binding;
- VII: protein complex formation;
- VIII: ribosome assembly;
- IX: RNA processing;
- X: rRNA modification;
- XI: tRNA modification;
- XII: tRNA charging.



Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312"

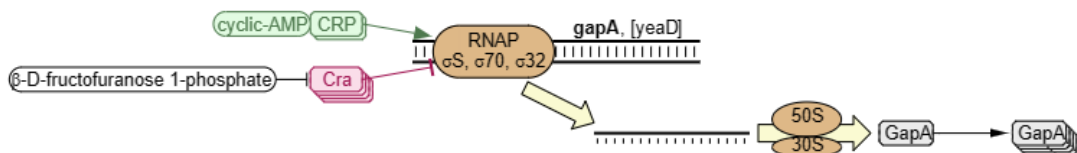


EcoCyc Transcription Units for *gapA* (b1779)

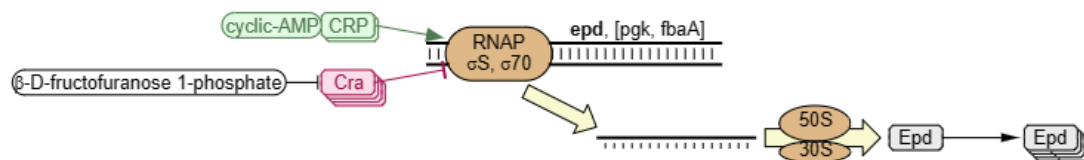


<https://www.biocyc.org/gene?orgid=ECOLI&id=EG10367#tab=TU>

B1779-based Reactions



<https://www.biocyc.org/gene?orgid=ECOLI&id=GAPDH-A-MONOMER>



<https://www.biocyc.org/gene?orgid=ECOLI&id=EG10368>

MetabolicReaction

TranscriptionReaction

Nucleotide Sequence

b1779

transcription_TU0_4904_from_RPOH_MONOMER,
transcription_TU0_4905_from_RPOS_MONOMER,
transcription_TU0_4905_from_RpoD_mono,
transcription_TU0_4906_from_RpoD_mono, or
transcription_TU0_4907_from_RpoD_mono.

TranscribedGene

RNA_b1779

MEReaction

DM_RNA_b1779 →

TranslationReaction

translation_b1779

TranslatedGene

protein_b1779

ComplexFormation

formation_GAPDH-A-CPLX

Complex

GAPDH-A-CPLX

E4PD_FWD_GAPDH-A-CPLX, or
E4PD_REV_GAPDH-A-CPLX.

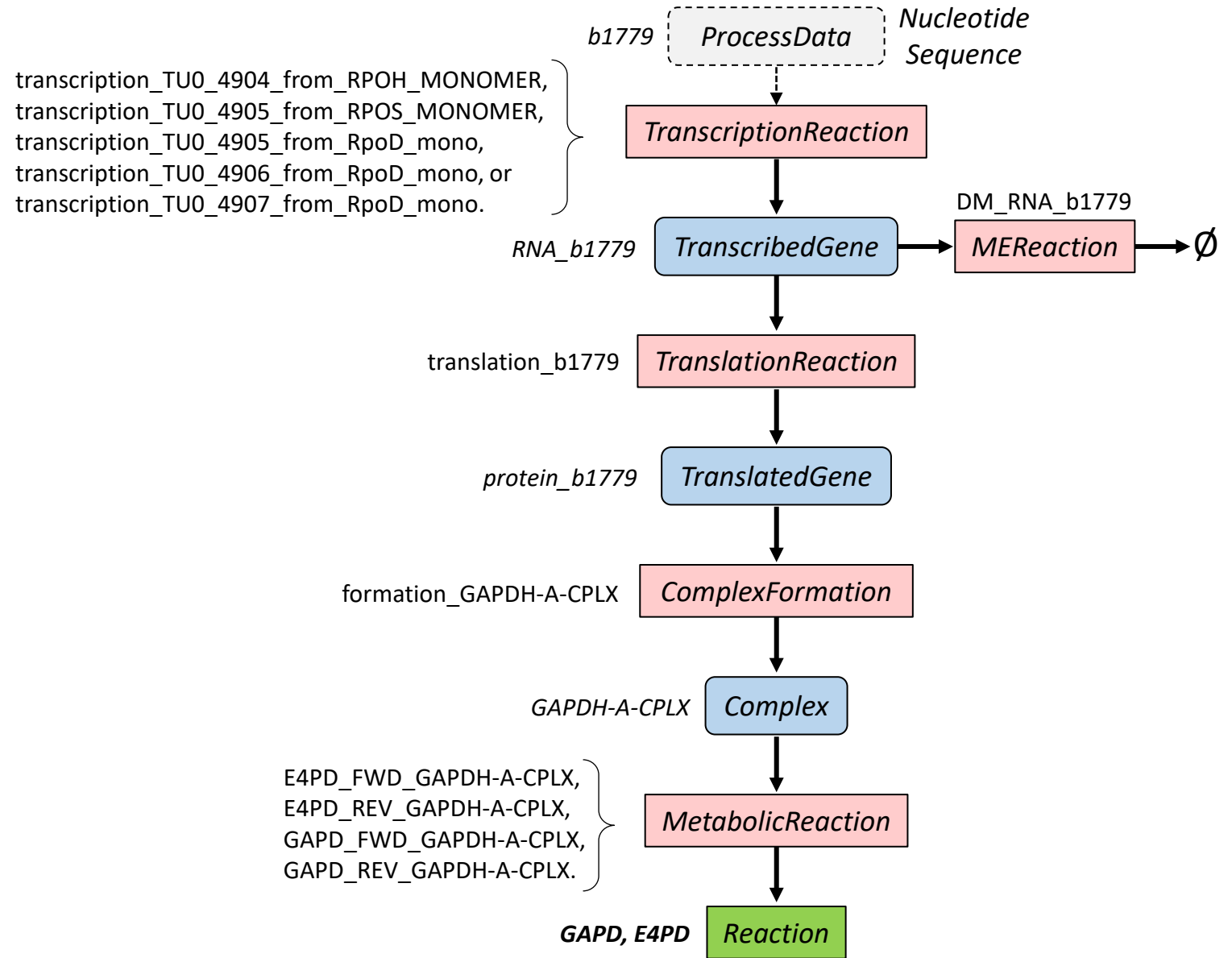
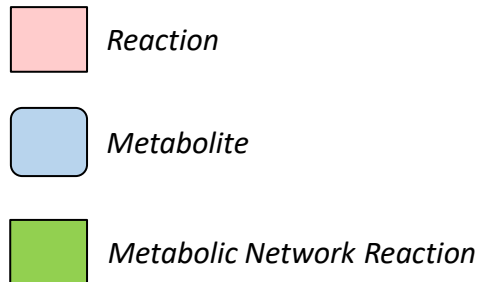
GAPD_FWD_GAPDH-A-CPLX, or
GAPD_REV_GAPDH-A-CPLX.

Reaction

E4PD


GAPD

"MetabolicReaction" (Enzyme) Production Process





Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - Macromolecular Reactions
 -  • Transcription
 - Translation
 - COBRAMe Execution
 - ECOLIme Operation

Transcription Reaction Lumping

Schematic representation of the network components and reactions is shown. In addition to the macromolecular synthesis of RNA and proteins, rRNA and tRNA processing reactions were included in the reconstruction.

I: transcription;

II: mRNA degradation;

III: translation;

IV: protein maturation;

V: protein folding;

VI: metallo-ion binding;

VII: protein complex formation;

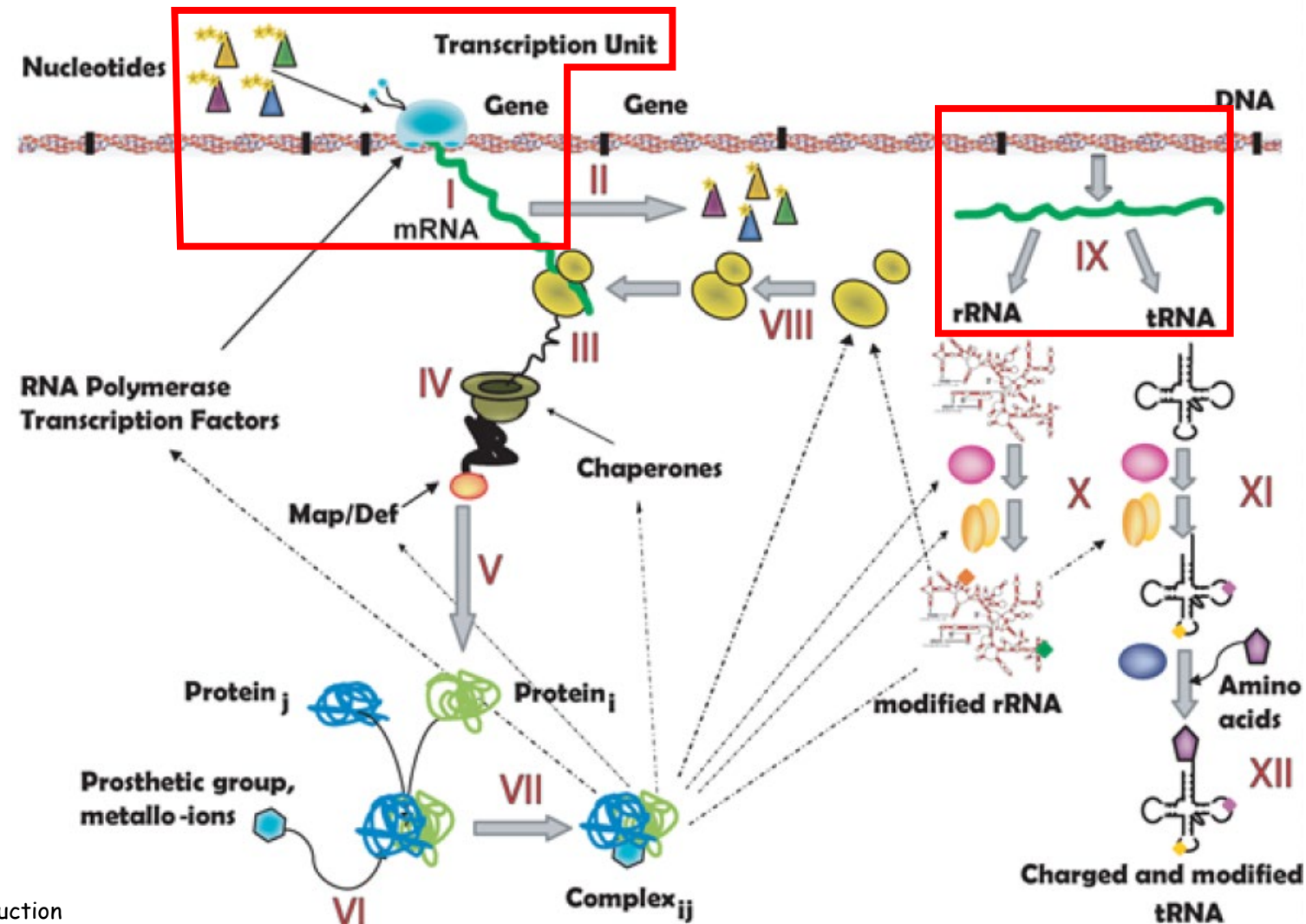
VIII: ribosome assembly;

IX: RNA processing;

X: rRNA modification;

XI: tRNA modification;

XII: tRNA charging.

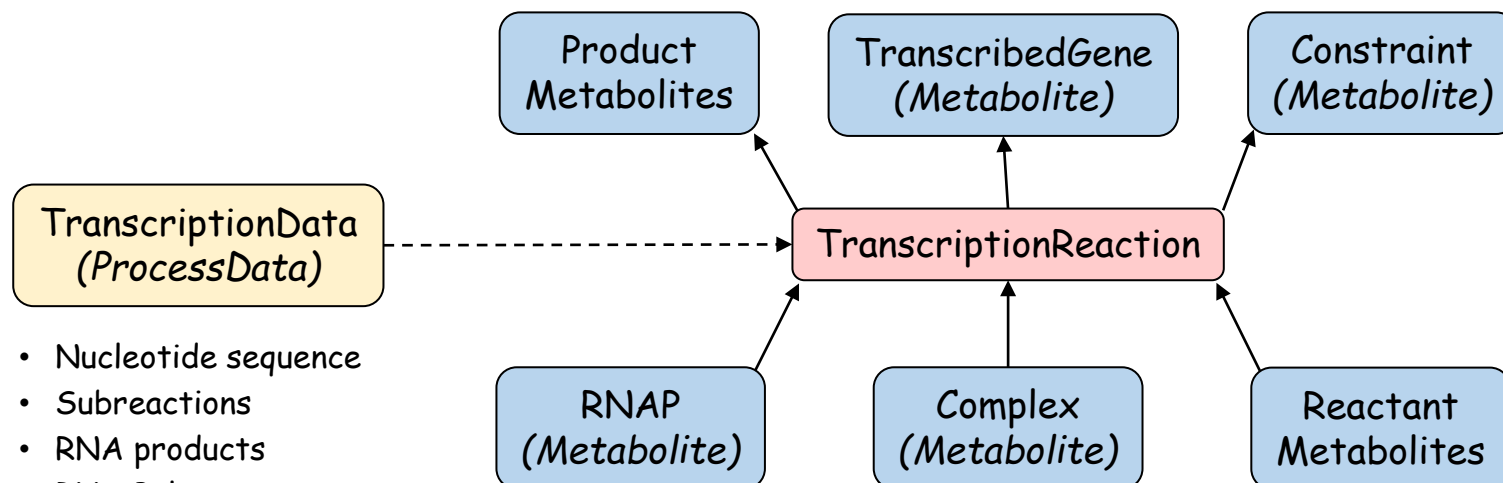
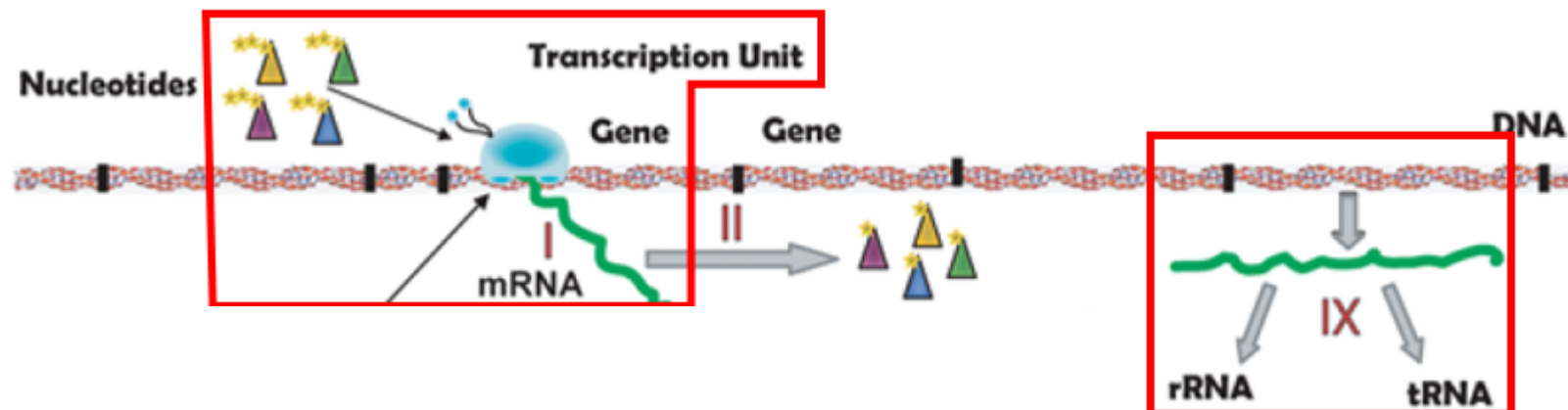


Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312"

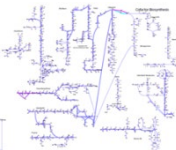
TranscriptionReaction

A "TranscriptionReaction" includes all the metabolites required to transcribe the DNA associated with a specific EcoCyc defined transcription unit to either RNA or mRNA. These metabolites include:

1. The nucleotide metabolites required for the reaction (derived from the nucleotide sequence located in the "processdata" class)
2. The metabolites required for transcription elongation
3. The metabolites required for transcription repair
4. The RNA polymerase and associated metabolites
5. The termination factor metabolites
6. The metabolites required for mRNA cleavage



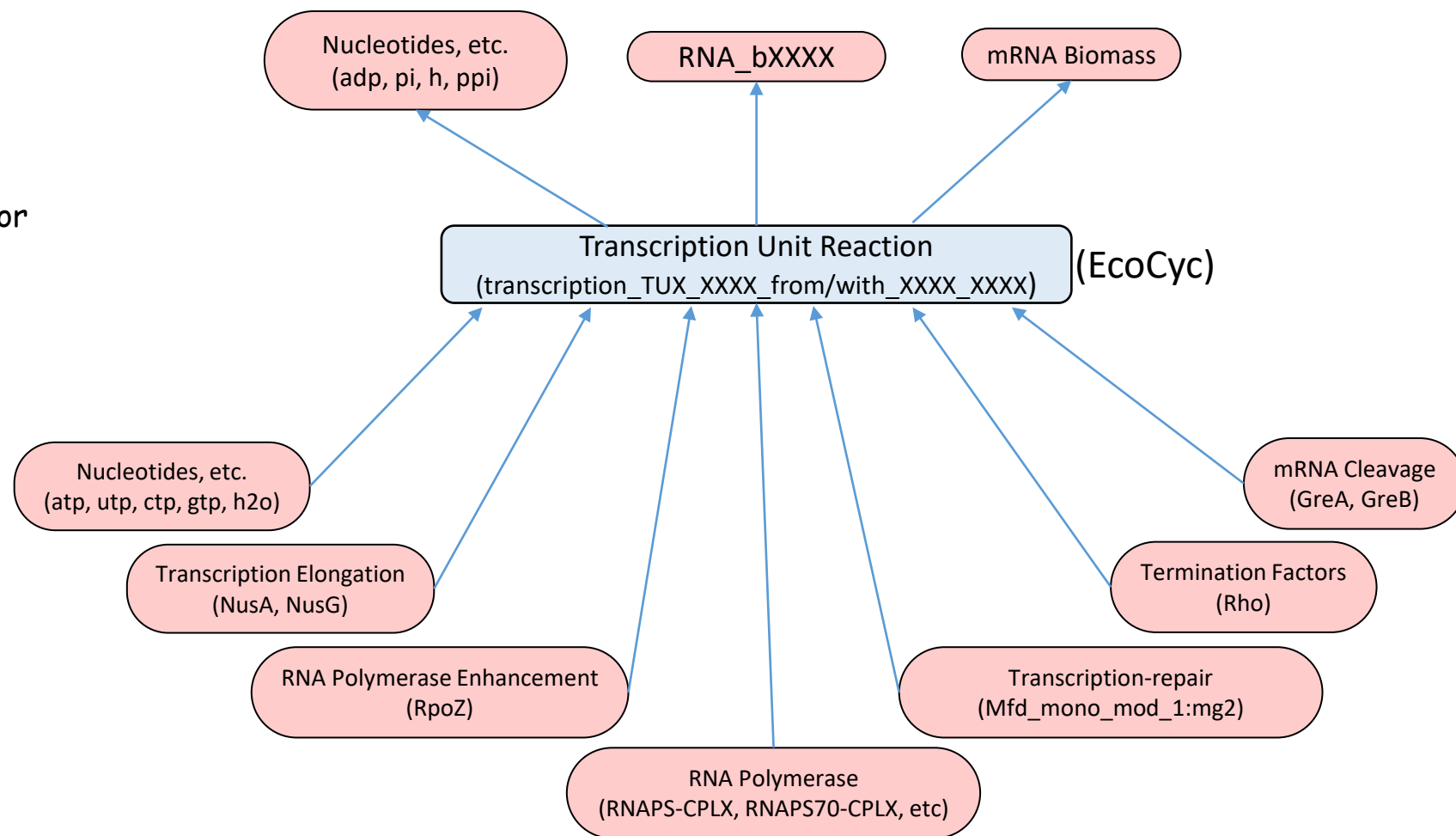
Metabolites required for elongation, mRNA cleavage, termination factors, and repair

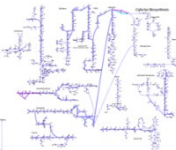


TranscriptionReactions

The reaction associated with each transcription unit (*TranscriptionReaction*) includes the following:

1. The nucleotide metabolites required for the reaction (derived from the nucleotide sequence located in the "processdata" class)
2. The metabolites required for transcription elongation
3. The metabolites required for transcription repair
4. The RNA polymerase and associated metabolites
5. The termination factor metabolites
6. The metabolites required for mRNA cleavage

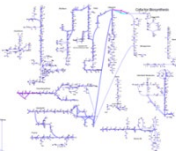




List of "TranscriptionReaction" Supporting Metabolites

- **Mfd_mono_mod_1:mg2**: transcription-repair coupling factor
- **NusA_mono**: transcription termination/antitermination protein NusA
- **NusB_mono**: transcription antitermination protein NusB
- **NusG_mono**: transcription termination/antitermination factor NusG
- **GreA_mono**: transcription elongation factor GreA
- **GreB_mono**: transcription elongation factor GreB
- **monocistronic_excision_machinery**: Processing of monocistronic mRNA
- **polycistronic_wout_rRNA_excision_machinery**: Processing of polycistronic mRNA.
- **Rho_hexa_mod_3:mg2**: transcription termination factor Rho
- **RplC_mono**: 50S ribosomal subunit protein L3
- **RNA_degradosome**: multiprotein complex involved in RNA degradation
- **RplD_mono**: 50S ribosomal subunit protein L4
- **RplM_mono**: 50S ribosomal subunit protein L13
- **RpoZ_mono_mod_1:mg2**: RNA polymerase subunit w
- **RpsD_mono**: 30S ribosomal subunit protein S4
- **RpsJ_mono**: 30S ribosomal subunit protein S10
- **rRNA_containing_excision_machinery**: processing of rRNA

Transcription.ipynb



Transcription “Complex” Metabolites

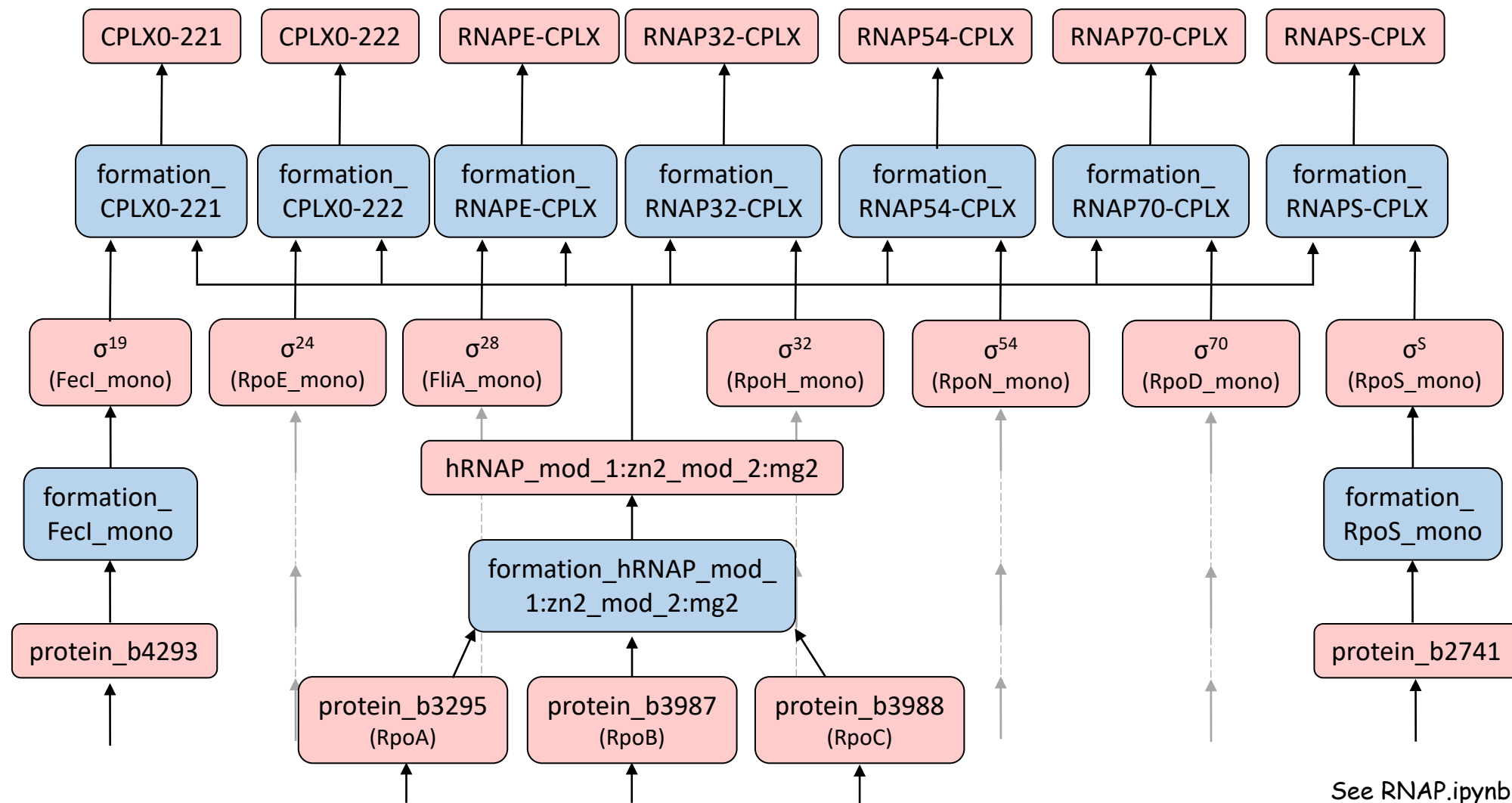
- **Mfd_mono_mod_1:mg2: transcription-repair coupling factor**
The Mfd protein, also known as transcription-repair coupling factor (TRCF), is responsible for ATP-dependent displacement of stalled RNA polymerase (RNAP) from DNA lesions. Mfd facilitates the repair of template strand lesions. (<https://ecocyc.org/gene?orgid=ECOLI&id=EG11619>)
- **NusA_mono: transcription termination/antitermination protein NusA**
Transcription termination/antitermination L factor (NusA) is a key component in both prevention and enhancement of transcriptional termination. It is important in both Rho-dependent and intrinsic termination, as well as in lambda and other phage antitermination systems. (<https://ecocyc.org/gene?orgid=ECOLI&id=EG10665>)
- **NusG_mono: transcription termination/antitermination factor NusG**
Transcription termination factor NusG is required for some kinds of Rho-dependent termination as well as for transcription antitermination. NusG is also involved in antitermination during lambda phage transcription (<https://ecocyc.org/gene?orgid=ECOLI&id=EG10667>)
- **GreA_mono: transcription elongation factor GreA**
GreA stimulates the mRNA cleavage activity of RNA polymerase, which acts to release a polymerase complex that has stalled or has incorporated an incorrect nucleotide. GreA (and GreB) is also required for wild-type transcription of some regulatory regions within lambda phage. (<https://ecocyc.org/gene?orgid=ECOLI&id=EG10415>)
- **GreB_mono: transcription elongation factor GreB**
GreB stimulates the mRNA cleavage activity of RNA polymerase, which acts to release a polymerase complex that has stalled. GreB (and GreA) is also required for wild-type transcription of some regulatory regions within lambda phage.
- **RpoZ_mono_mod_1:mg2: RNA polymerase subunit ω**
RpoZ copurifies with RNA polymerase and may play a structural role in that complex.
- **Rho_hexa_mod_3:mg2: transcription termination factor Rho**
Rho is required for one of the two major types of termination of RNA transcription.



RNA Polymerase

The iJL1678b (ECOLIme) model RNA polymerase includes:

- A unique sigma factor is included in each RNAP metabolite
- There are seven different RNAP metabolites, each corresponding to a different sigma factor.



Modeling the RNA Polymerases in COBRAME

1. Load the python packages

```
In [1]: from __future__ import print_function, division, absolute_import

# python imports
import re
from os.path import join
from collections import defaultdict
import pickle
import pandas as pd

# third party imports
import pandas
import tabulate
import cobra
pd.set_option('display.max_columns', 100)
pd.set_option('display.width', 100)
pd.set_option('display.max_colwidth', 100)

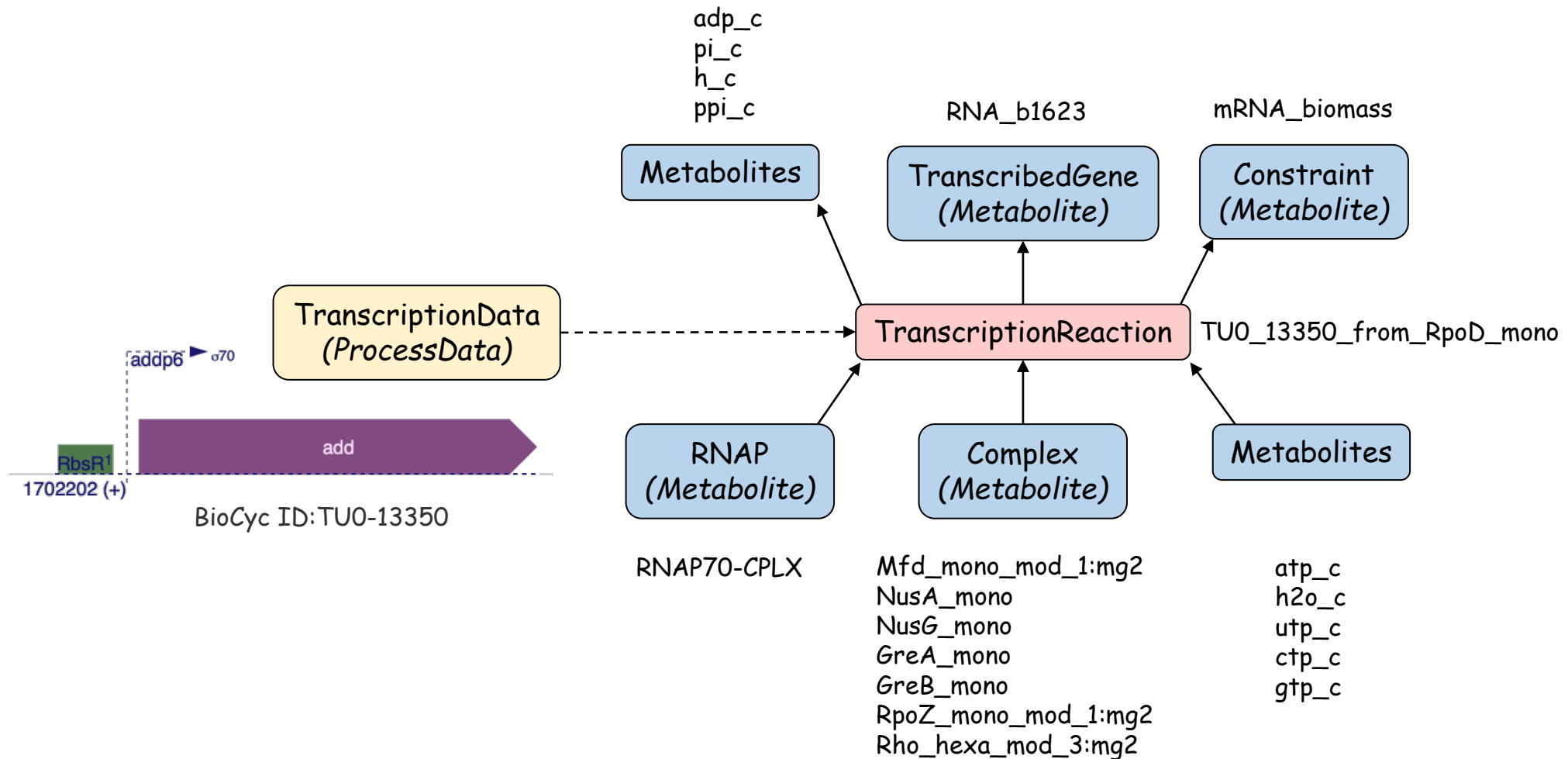
# ECOLIME
import ecolime
from ecolime import (transcription, translation, flat_files, generics, formulas, compartments)

# COBRAME
import cobrame
from cobrame.util import building, mu, me_model_interface
#from cobrame.io.json import save_json_me_model, save_reduced_json_me_model
```

RNAP.ipynb

TranscriptionReaction

Example: "transcription_TU0_13350_from_RpoD_mono"



Modeling Transcription in the iJL1678b (ECOLIme) Model

1. Load the python packages

```
In [1]: from __future__ import print_function, division, absolute_import

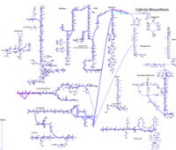
# python imports
import re
from os.path import join
from collections import defaultdict
import pickle
import pandas as pd

# third party imports
import pandas
import tabulate
import cobra
pd.set_option('display.max_columns', 100)
pd.set_option('display.width', 100)
pd.set_option('display.max_colwidth', 100)

# ECOLIme
import ecolime
from ecolime import (transcription, translation, flat_files, generics, formulas, compartments)

# COBRAME
import cobrame
from cobrame.util import building, mu, me_model_interface
#from cobrame.io.json import save_json_me_model, save_reduced_json_me_model
```

Transcription.ipynb



Creating a Transcription Reaction

1. Define the RNA id, rna type, nucleotide sequence

- `gene = cobra.CobraGene('RNA_a', 'mRNA', sequence)`
- `me.add_metabolites([gene])`

2. Add the transcription data to the model

- `transcription_data = cobra.TranscriptionData('TU_a', me, rna_products={'RNA_a'})`
- `transcription_data.nucleotide_sequence = sequence`

3. Update the transcription reaction

- `transcription_rxn.update()`

4. Incorporate the RNA Polymerase in the model

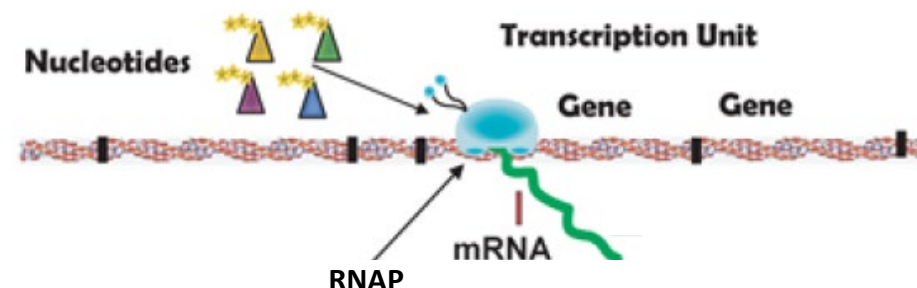
- `RNAP = cobra.RNAP('RNA_polymerase')`
- `me.add_metabolites(RNAP)`

5. Associate the RNA polymerase with the transcription data

- `for data in me.transcription_data:`
- `data.RNA_polymerase = RNAP.id`
- `me.reactions.transcription_TU_a.update()`

6. Examine reaction

- $0.00088887053605567 \mu + 0.000347992814865795 \text{ RNA_polymerase} + 86 \text{ atp_c} + 38 \text{ ctp_c} + 12 \text{ gtp_c} + 50 \text{ utp_c}$
→ $\text{RNA_a} + 59.172286 \text{ mRNA_biomass} + 186 \text{ ppi_c}$



https://cobrame.readthedocs.io/en/master/building_a_model.html



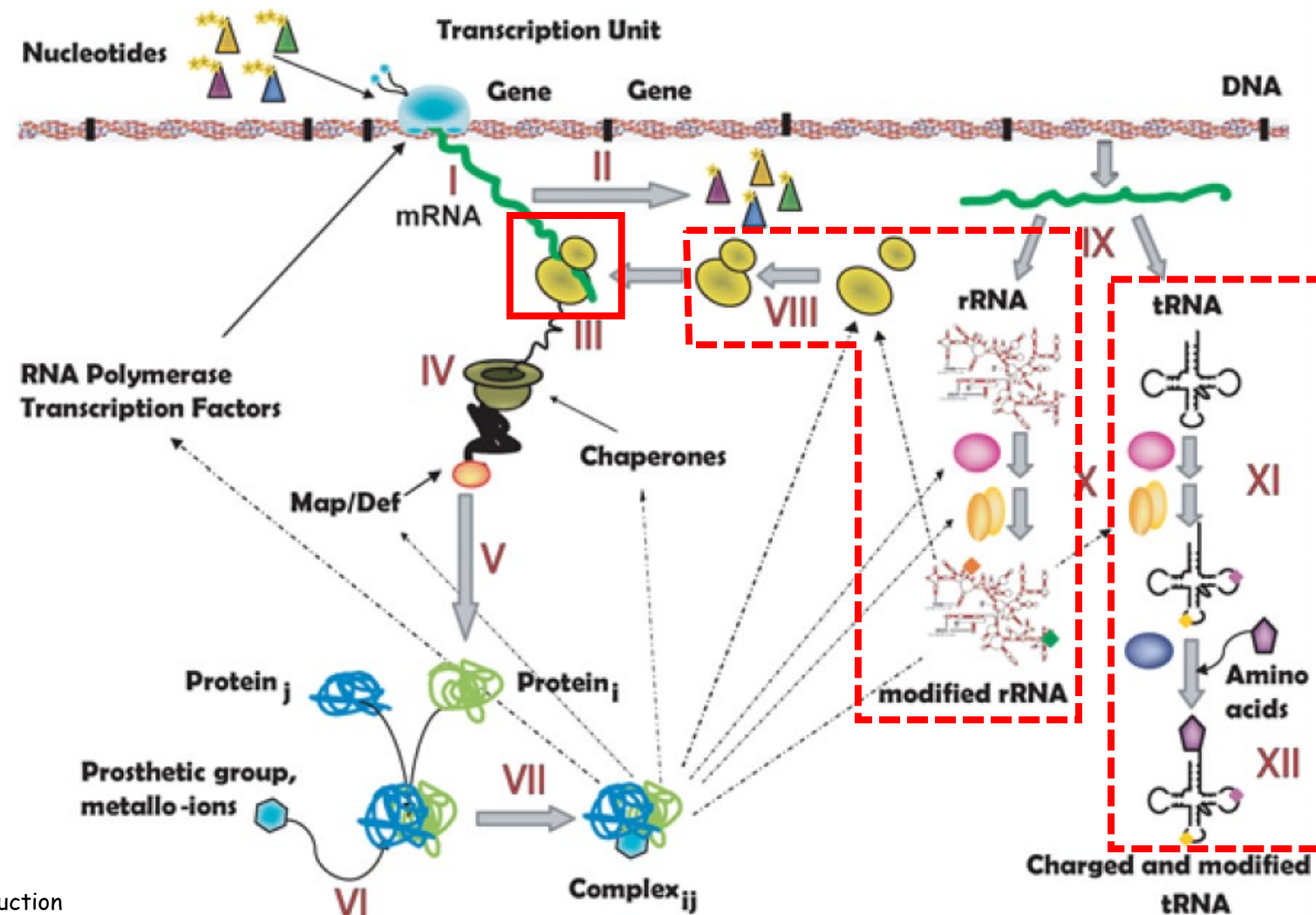
Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - Macromolecular Reactions
 - Transcription
 - • Translation
 - COBRAMe Execution
 - ECOLIme Operation

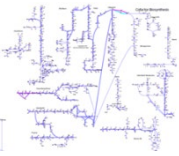
Translation Reaction Lumping

Schematic representation of the network components and reactions is shown. In addition to the macromolecular synthesis of RNA and proteins, rRNA and tRNA processing reactions were included in the reconstruction.

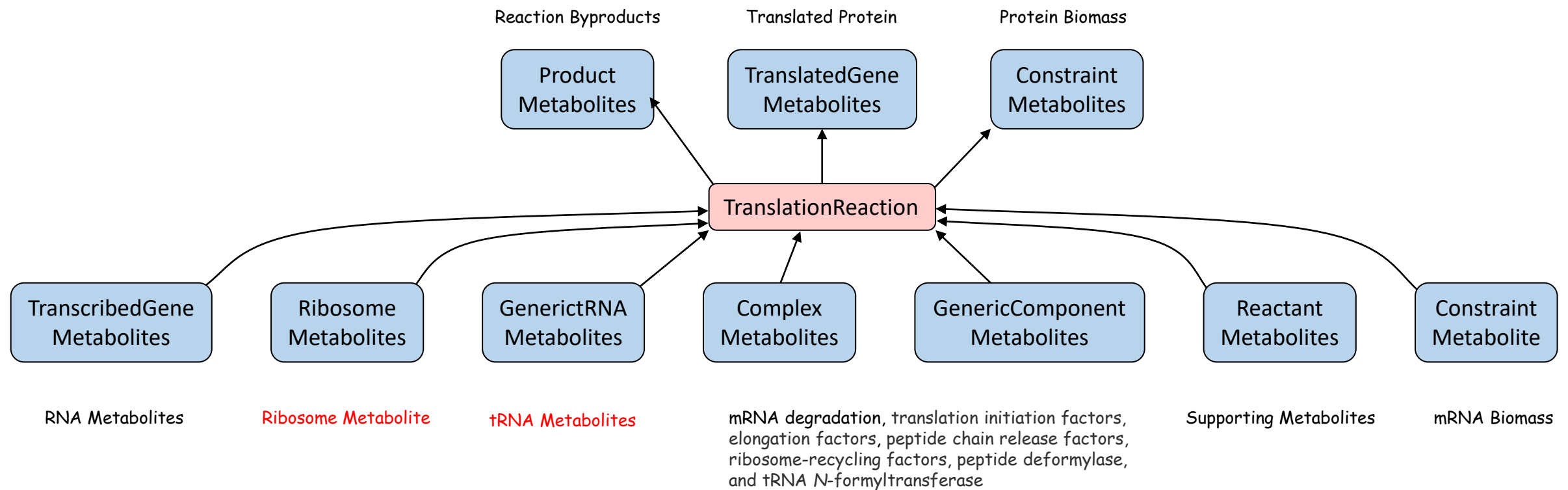
- I: transcription;
- II: mRNA degradation;
- III: translation;**
- IV: protein maturation;
- V: protein folding;
- VI: metallo-ion binding;
- VII: protein complex formation;
- VIII: ribosome assembly;
- IX: RNA processing;
- X: rRNA modification;
- XI: tRNA modification;
- XII: tRNA charging.



Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312"



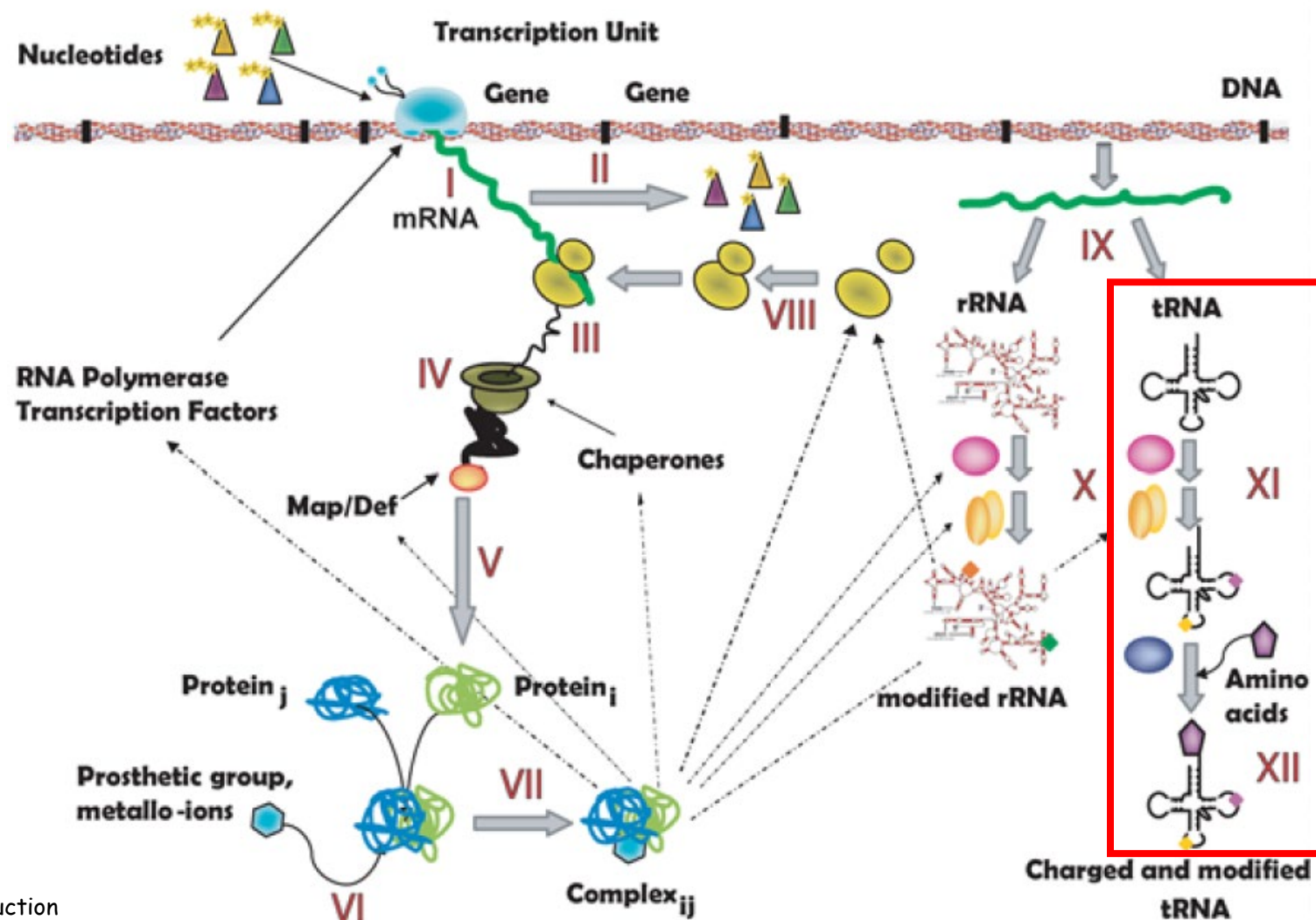
iJL1678b (ECOLIme) Translation Overview



tRNA Biosynthesis

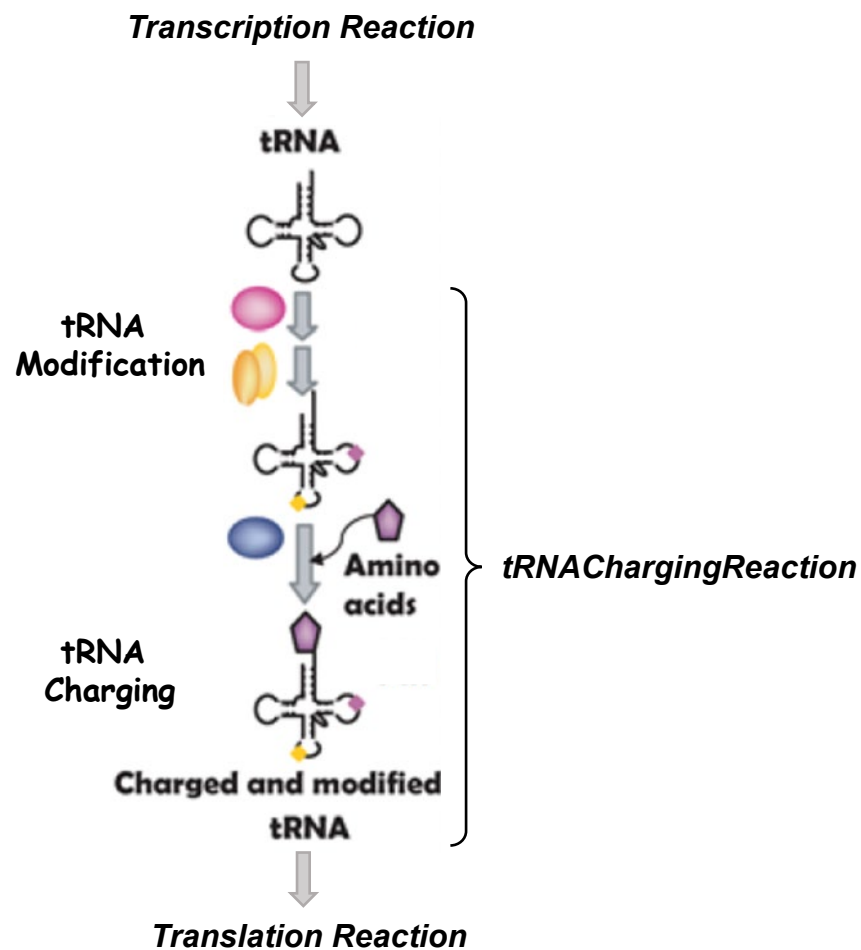
Schematic representation of the network components and reactions is shown. In addition to the macromolecular synthesis of RNA and proteins, rRNA and tRNA processing reactions were included in the reconstruction.

- I: transcription;
- II: mRNA degradation;
- III: translation;
- IV: protein maturation;
- V: protein folding;
- VI: metallo-ion binding;
- VII: protein complex formation;
- VIII: ribosome assembly;
- IX: RNA processing;
- X: rRNA modification;
- XI: tRNA modification;**
- XII: tRNA charging.**

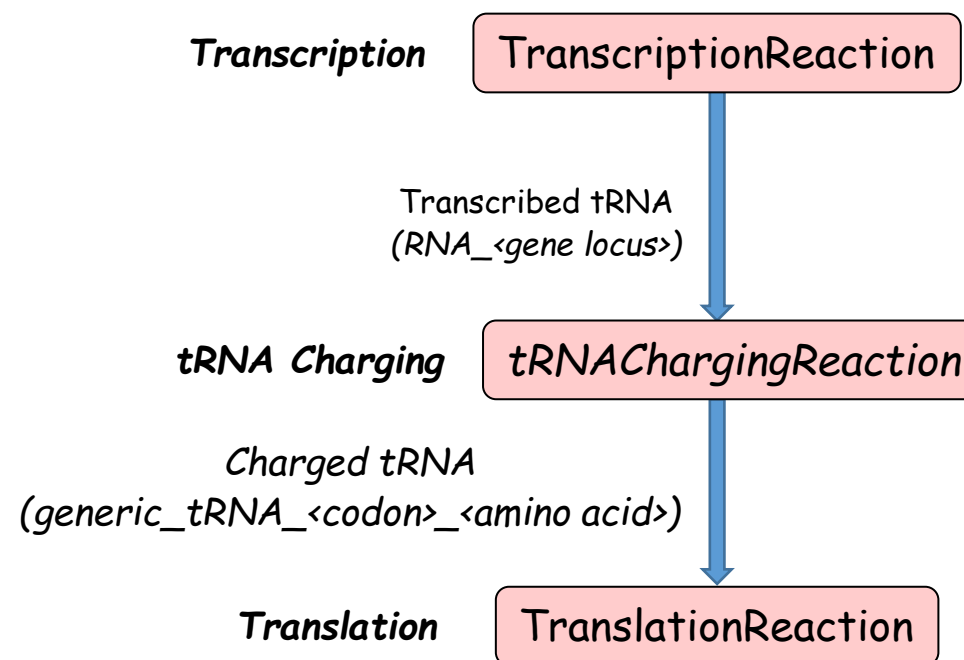


Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312"

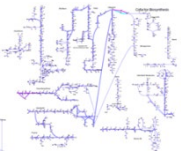
tRNA Bioynthesis in COBRame Models



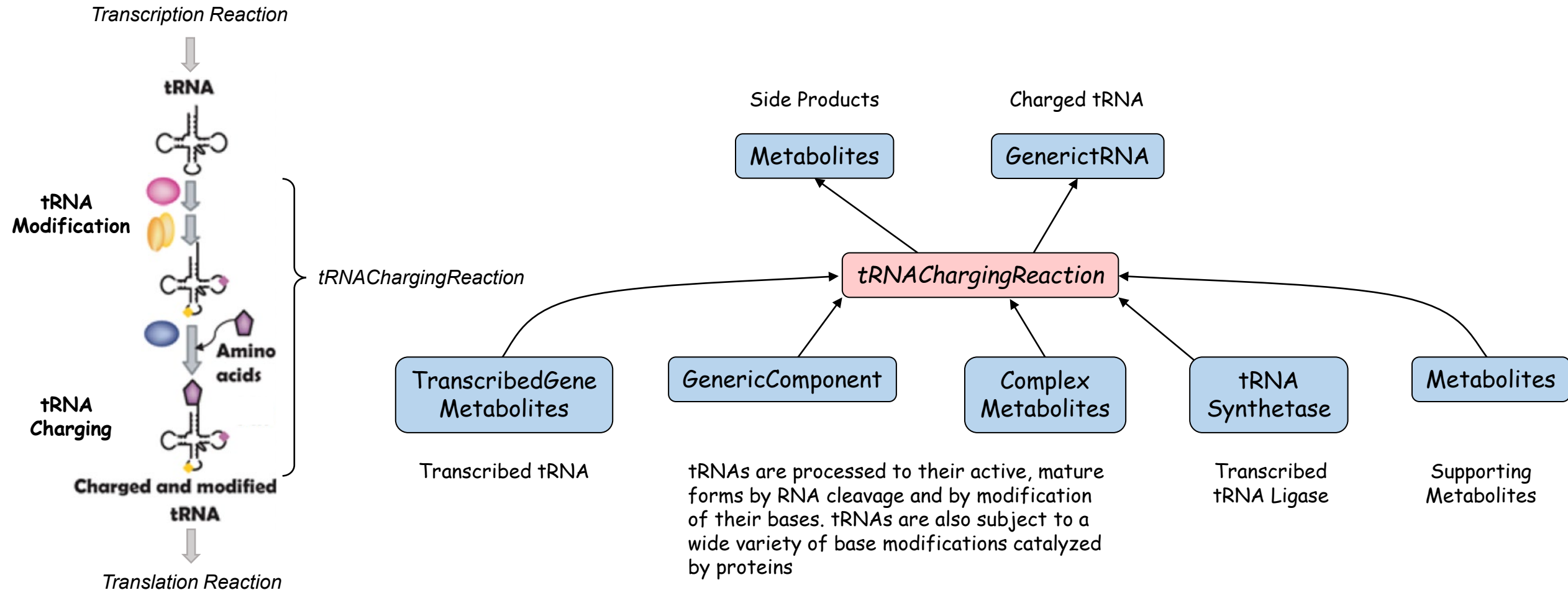
Original ME Model



COBRame Models



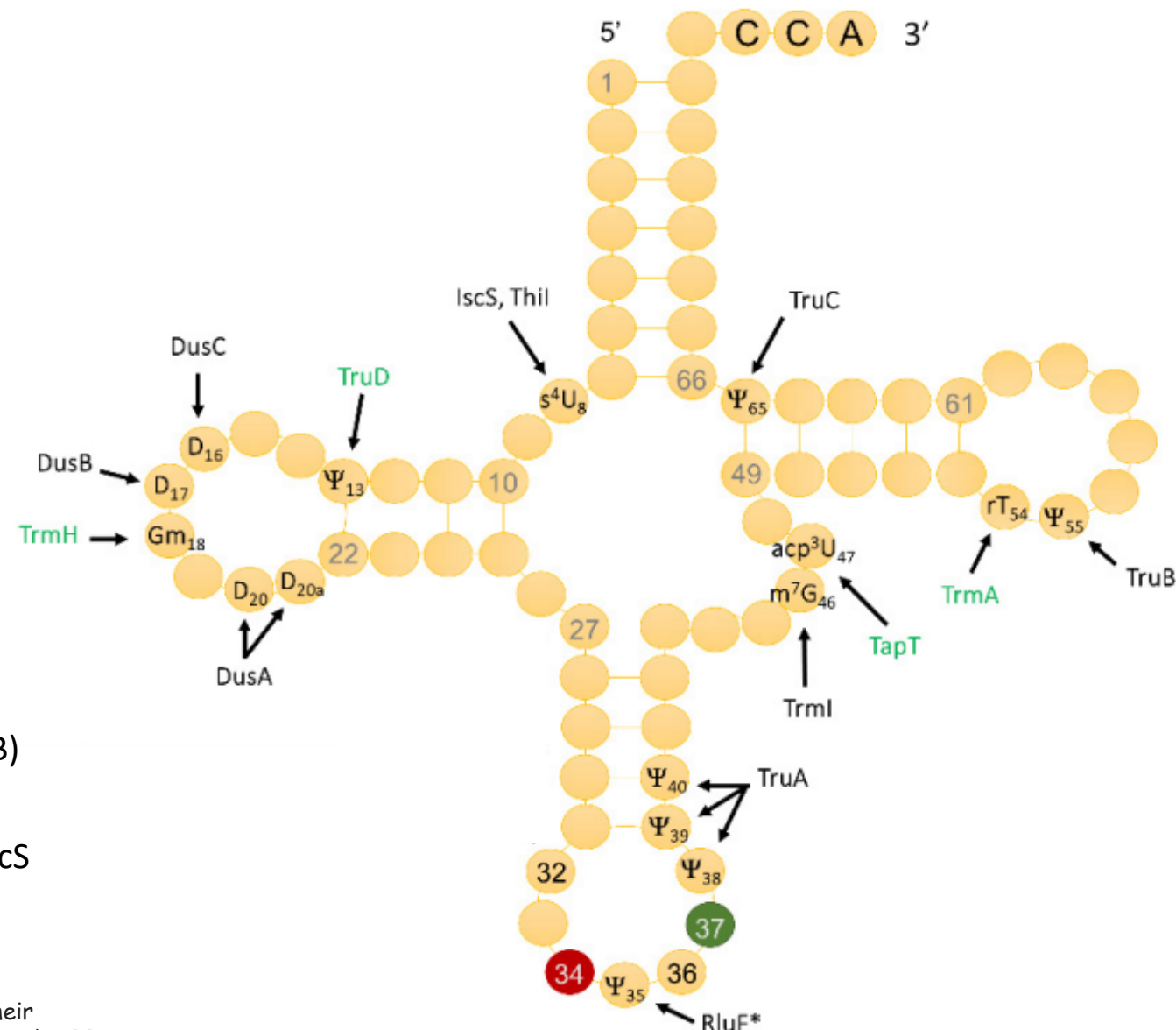
iJL1678b (ECOLIme) tRNA Overview



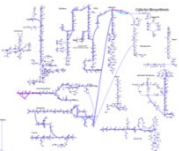
Escherichia coli tRNA Modifications and Corresponding Enzymes

Examples of glycine tRNA modifications:

- 'Thil_mono'- tRNA uridine 4-sulfurtransferase
- 'Gly_RS_tetra'- Glycine—tRNA ligase
- 'generic_Dus'- tRNA-dihydrouridine synthase A
- 'generic_Dus'- tRNA-dihydrouridine synthase B
- 'generic_Dus'- tRNA-dihydrouridine synthase C
- 'YggH_mono'- tRNA m⁷G₄₆ methyltransferase (also TrmB)
- 'TruB_mono'- tRNA pseudouridine₅₅ synthase
- 'IscS_mod_2:pydx5p_mod_1:SH'- cysteine desulfurase IscS
- 'TrmA_mono'- tRNA m⁵U₅₄ methyltransferase

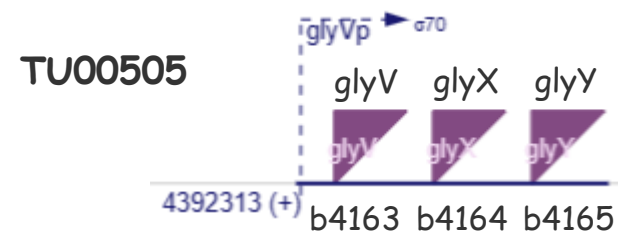
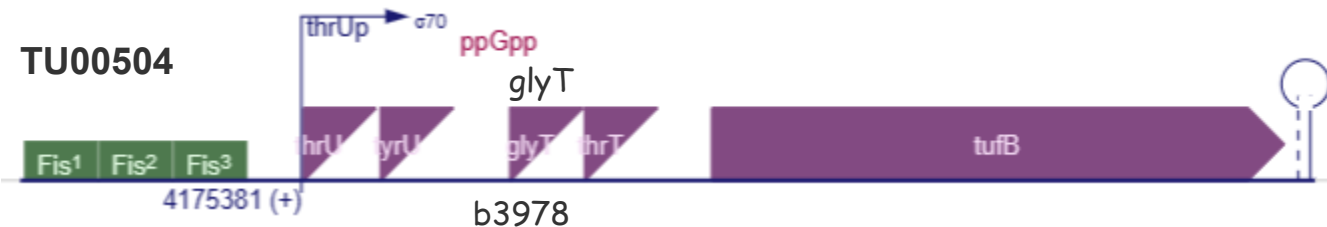


de Crecy-Lagard, Valerie, et al. "Survey and validation of tRNA modifications and their corresponding genes in Bacillus subtilis sp subtilis strain 168." *Biomolecules* 10.7 (2020): 977.

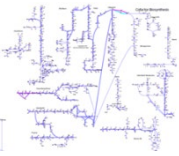


Example: Glycine tRNA Transcription Units

tRNA Gene	Glycine tRNA Transcription Reactions	Glycine RNA Metabolite
b1911	[transcription_TU00512_from_RpoD_mono]	RNA_b1911
b2864	[transcription_TU0_13770_from_RpoD_mono, transcription_TU00516_from_RpoD_mono]	RNA_b2864
b3978	[transcription_TU00504_from_RpoD_mono]	RNA_b3978
b4163	[transcription_TU00505_from_RpoD_mono]	RNA_b4163
b4164	[transcription_TU00505_from_RpoD_mono]	RNA_b4164
b4165	[transcription_TU00505_from_RpoD_mono]	RNA_b4165

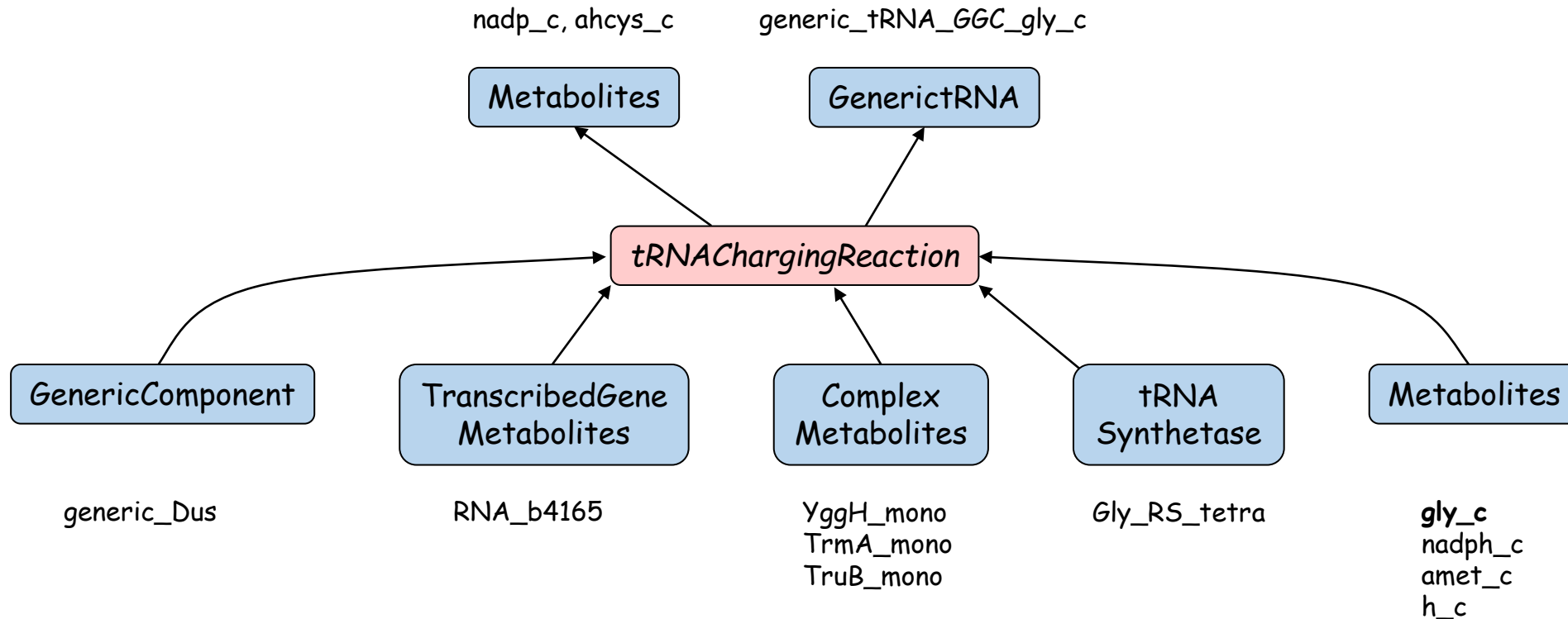


<https://ecocyc.org/>



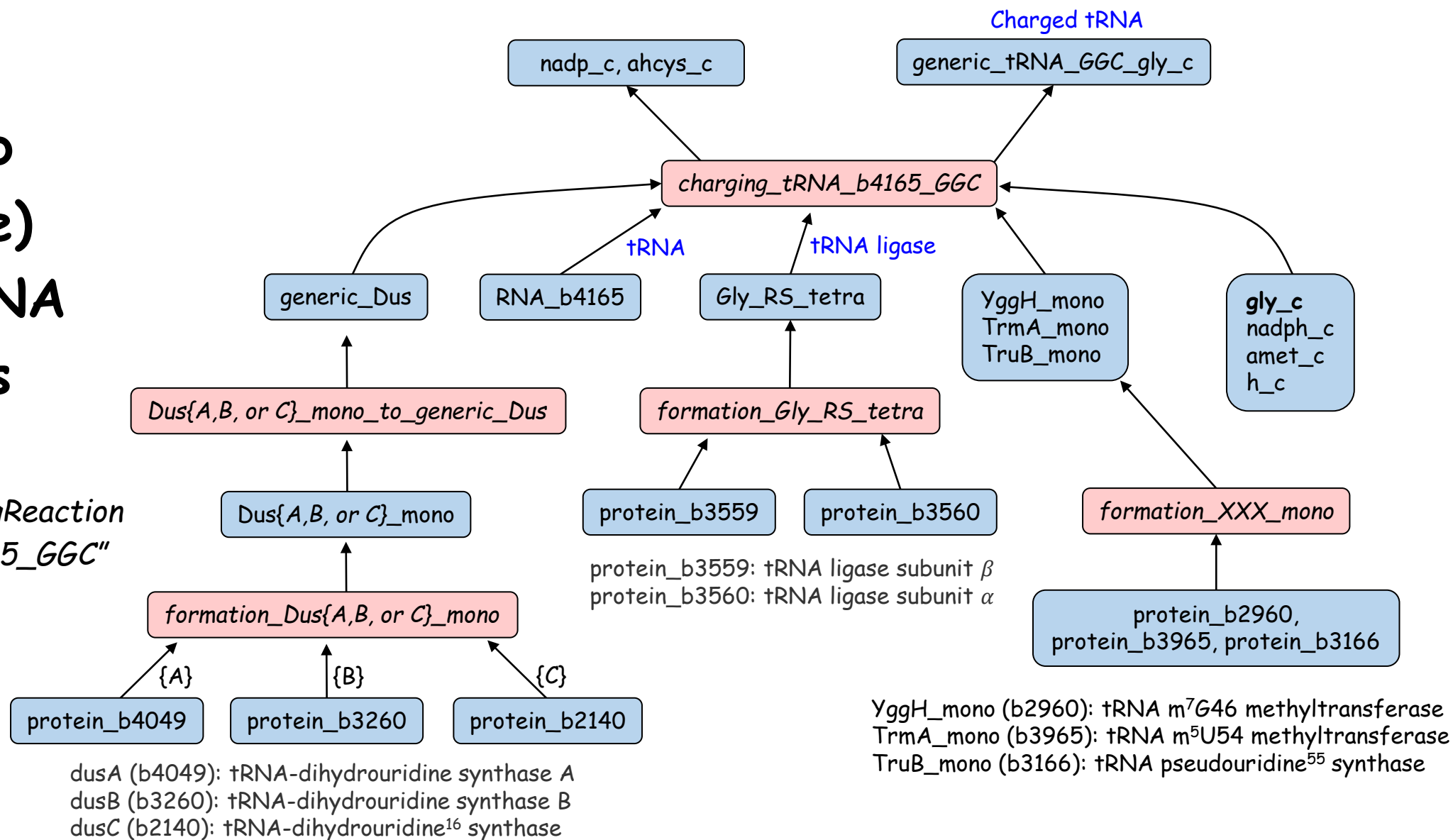
Example: iJL1678-ME ECOLIme Glycine tRNA

Example: *tRNAChargingReaction* - "charging_tRNA_b4165_GGC"

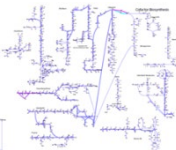


iJL1678b (ECOLIme) Glycine tRNA Pathways

Example: tRNAChargingReaction
"charging_tRNA_b4165_GGC"



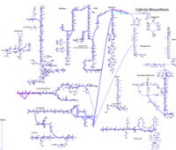
tRNA_glycine_pathways.ipynb



Key Glycine tRNA Reactions and Metabolites

tRNA Gene	Glycine tRNA Transcription Reactions	Glycine RNA Metabolite	Glycine Charging Reactions	Glycine Charged tRNA
b1911	[transcription_TU00512_from_RpoD_mono]	RNA_b1911	[charging_tRNA_b1911_GGC, charging_tRNA_b1911_GGU]	[generic_tRNA_GGC_gly_c, generic_tRNA_GGU_gly_c]
b2864	[transcription_TU0_13770_from_RpoD_mono transcription_TU00516_from_RpoD_mono]	RNA_b2864	[charging_tRNA_b2864_GGG]	[generic_tRNA_GGG_gly_c]
b3978	[transcription_TU00504_from_RpoD_mono]	RNA_b3978	[charging_tRNA_b3978_GGA]	[generic_tRNA_GGA_gly_c]
b4163	[transcription_TU00505_from_RpoD_mono]	RNA_b4163	[charging_tRNA_b4163_GGC, charging_tRNA_b4163_GGU]	[generic_tRNA_GGC_gly_c, generic_tRNA_GGU_gly_c]
b4164	[transcription_TU00505_from_RpoD_mono]	RNA_b4164	[charging_tRNA_b4164_GGC, charging_tRNA_b4164_GGU]	[generic_tRNA_GGC_gly_c, generic_tRNA_GGU_gly_c]
b4165	[transcription_TU00505_from_RpoD_mono]	RNA_b4165	[charging_tRNA_b4165_GGC, charging_tRNA_b4165_GGU]	[generic_tRNA_GGC_gly_c, generic_tRNA_GGU_gly_c]

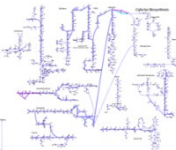
tRNA_glycine_pathways.ipynb



Flux for Key Glycine tRNA Reactions

Transcription Reactions	Transcription Reactions Flux	Charging Reactions	Charging Reactions Flux
transcription_TU00512_from_RpoD_mono	0.000006	charging_tRNA_b1911_GGC	0.044674
transcription_TU0_13770_from_RpoD_mono	0.000005	charging_tRNA_b1911_GGU	0.000000
transcription_TU00516_from_RpoD_mono	0.000000	charging_tRNA_b2864_GGG	0.033864
transcription_TU00504_from_RpoD_mono	0.000003	charging_tRNA_b3978_GGA	0.022934
transcription_TU00505_from_RpoD_mono	0.000012	charging_tRNA_b4163_GGC	0.019518
transcription_TU00505_from_RpoD_mono	0.000012	charging_tRNA_b4163_GGU	0.066318
transcription_TU00505_from_RpoD_mono	0.000012	charging_tRNA_b4164_GGC	0.000000
		charging_tRNA_b4164_GGU	0.085836
		charging_tRNA_b4165_GGC	0.085836
		charging_tRNA_b4165_GGU	0.000000

tRNA_glycine_pathways.ipynb



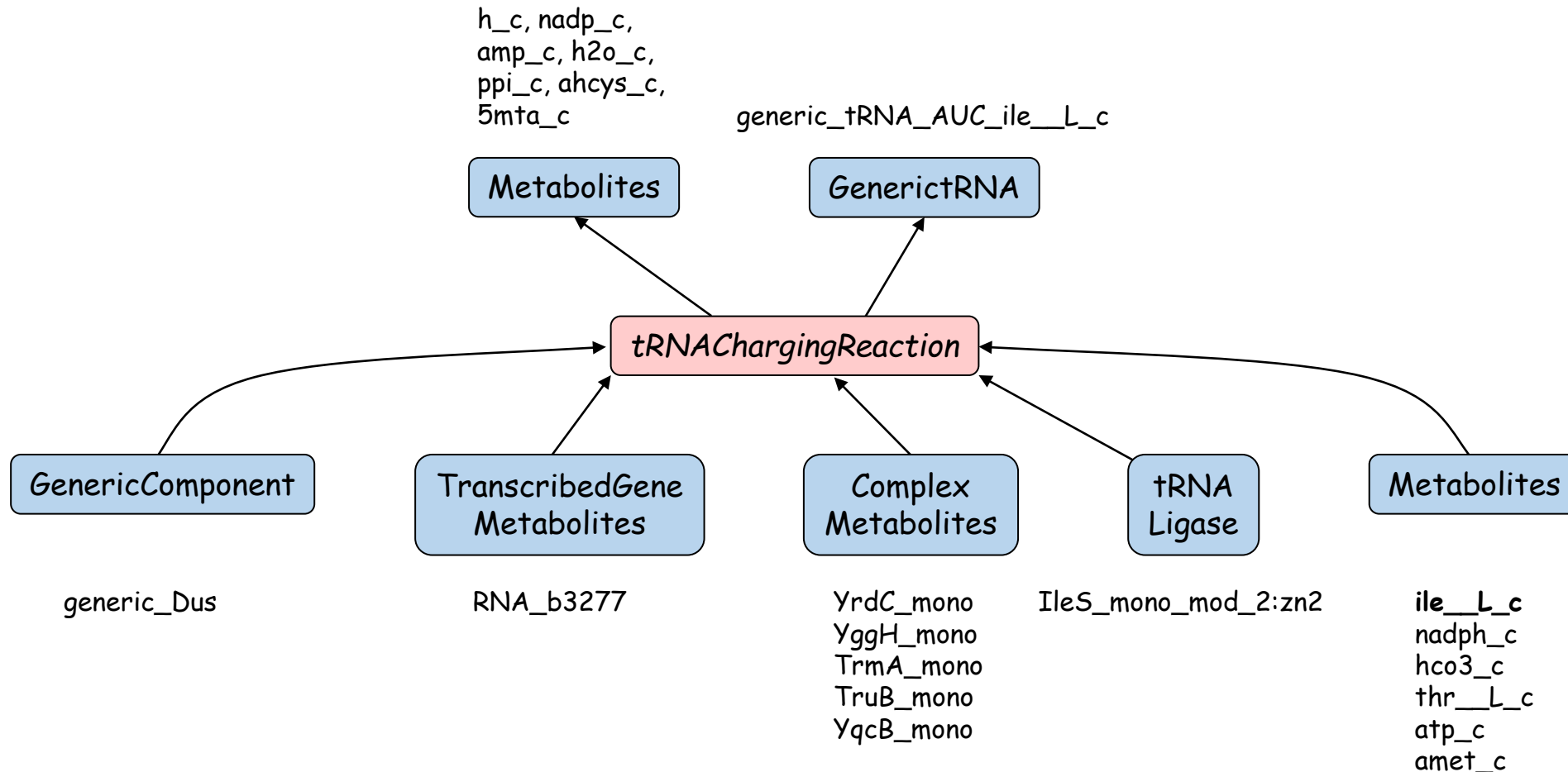
Flux of Support Glycine tRNA Reactions

Glycine Support Metabolites	Glycine Support Reactions	tRNA Modifications	Glycine Support Reactions Flux
Gly_RS_tetra	formation_Gly_RS_tetra	tRNA uridine 4-sulfurtransferase	1.270606e-06
TrmA_mono	formation_TrmA_mono	tRNA-dihydrouridine synthase A	2.284456e-09
generic_Dus	DusA_mono_to_generic_Dus	Glycine—tRNA ligase	0.000000e+00
generic_Dus	DusB_mono_to_generic_Dus	tRNA-dihydrouridine synthase B	0.000000e+00
generic_Dus	DusC_mono_to_generic_Dus	tRNA-dihydrouridine synthase C	4.679154e-09
YggH_mono	formation_YggH_mono	tRNA m7G46 methyltransferase	1.635159e-09
Thil_mono	formation_Thil_mono	tRNA pseudouridine55 synthase	1.130280e-09
TruB_mono	formation_TrkB_mono	cysteine desulfurase IscS	2.339411e-09
IscS_mod_2:pydx5p_mod_1:SH	ICYSDS1_FWD_CPLX_dummy	tRNA m5U54 methyltransferase	1.060364e-03

tRNA_glycine_pathways.ipynb

Example: iJL1678-ME ECOLIme Isoleucine tRNA

Example: *tRNAChargingReaction* - "charging_tRNA_b3277_AUC"



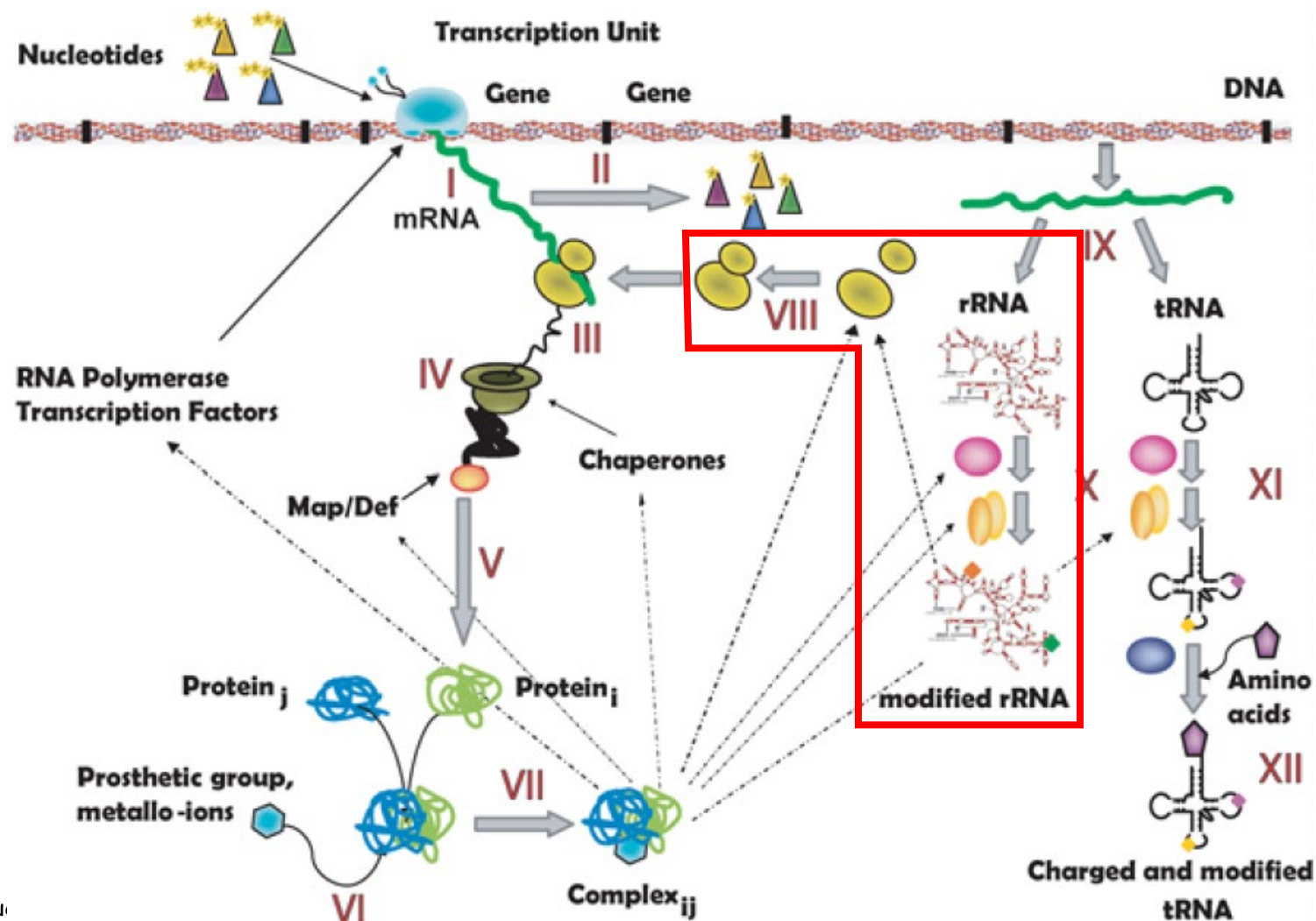
Example: tRNAChargingReaction
"charging_tRNA_b3277_AUC"



Ribosome Assembly

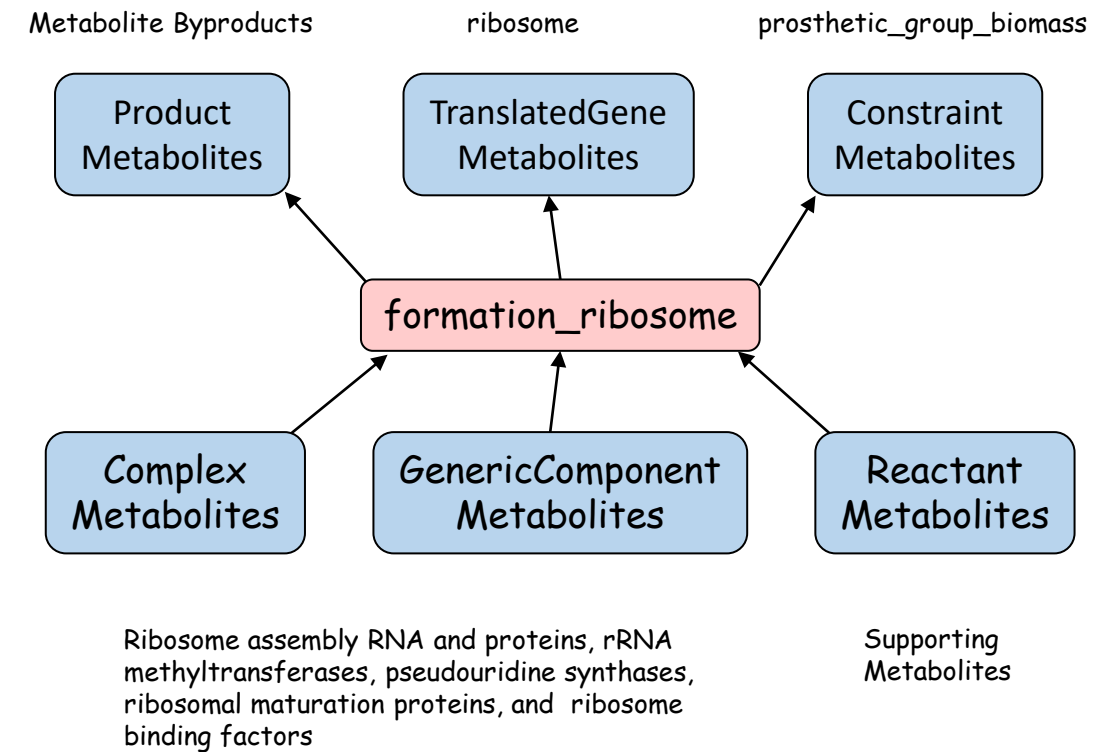
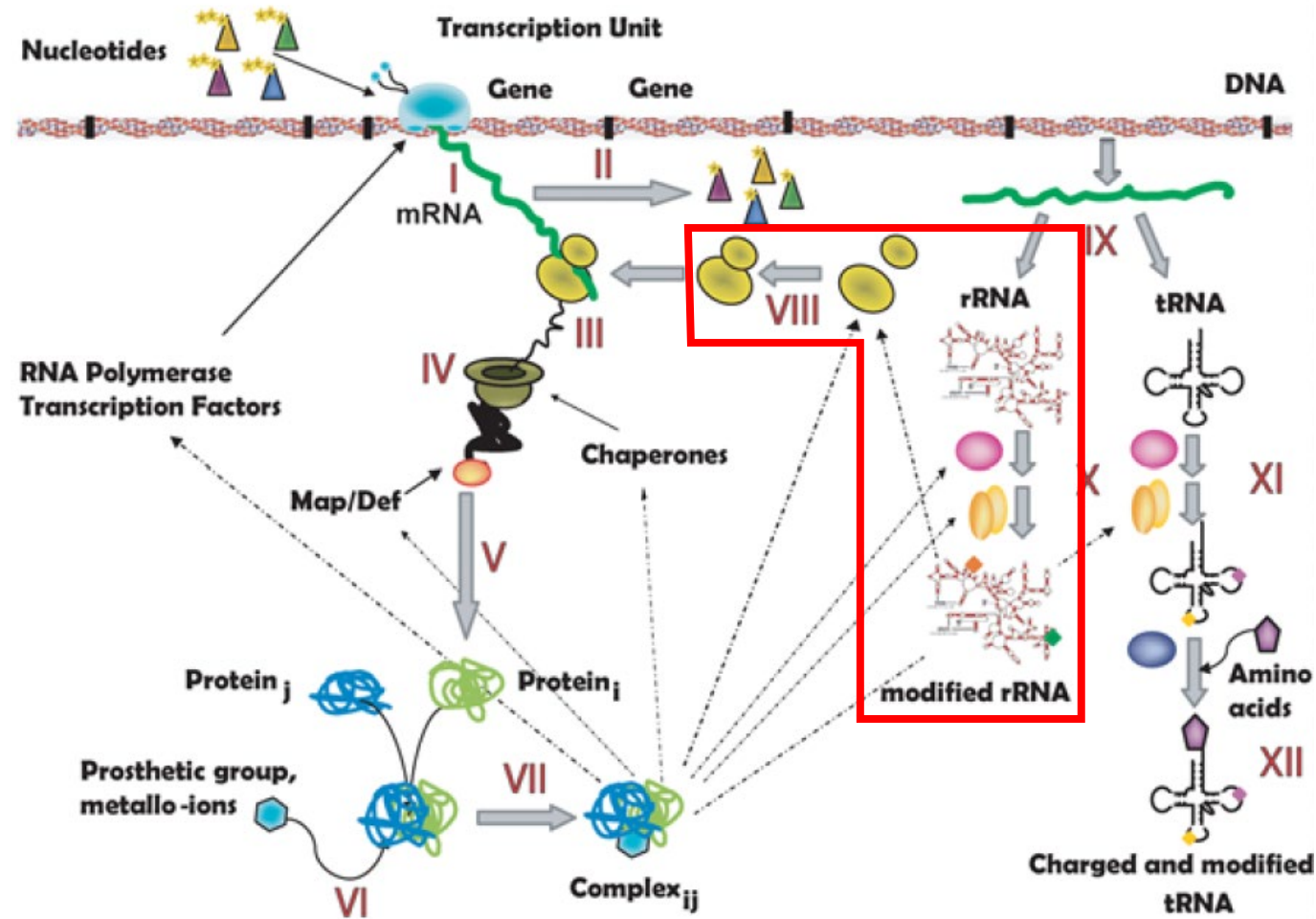
Schematic representation of the network components and reactions is shown. In addition to the macromolecular synthesis of RNA and proteins, rRNA and tRNA processing reactions were included in the reconstruction.

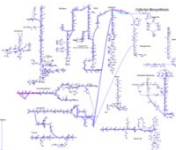
- I: transcription;
- II: mRNA degradation;
- III: translation;
- IV: protein maturation;
- V: protein folding;
- VI: metallo-ion binding;
- VII: protein complex formation;
- VIII: ribosome assembly;**
- IX: RNA processing;
- X: rRNA modification;
- XI: tRNA modification;
- XII: tRNA charging.



Thiele I, Jamshidi N, Fleming RMT, Palsson BØ (2009) Genome-Scale Reconstruction of Escherichia coli's Transcriptional and Translational Machinery: A Knowledge Base, Its Mathematical Formulation, and Its Functional Characterization. PLoS Comput Biol 5(3): e1000312. doi:10.1371/journal.pcbi.1000312"

iJL1678b (ECOLIme) Ribosome Formation





Ribosome Assembly Genes: "formation_ribosome"

30S ribosomal subunit RNA
B3851

30S ribosomal subunit proteins
b0911, b0169, b3314, b3296, b3303, b4200, b3341,
b3306, b3230, b3321, b3297, b3329, b3298, b3307,
b3165, b2609, b3311, b4202, b3316, b0023, b3065

50S ribosomal subunit RNA
b3854, b3855

50S ribosomal subunit Protein
(b3984, b3317, b3320, b3319, b3308, b3305, b3986,
b3985, b3986, b4203, b3985, b3983, b3986, b3231,
b3310, b3301, b3313, b3294, b3304, b2606, b1716,
(b3186, b3315, b3318, b3309, b2185, b3185, b3637,
b3312, b3302, b3936, b1089, b3636, b3703, b1717,
b3299

16S rRNA methyltransferases
b0051, b3289, b4371, b3465, b2946, b1835, b3740

23S rRNA methyltransferases
b1822, b4180, b0859, b2785, b3179, b0807, b3084, b0636,
b0967, b0948, b2806, b2517

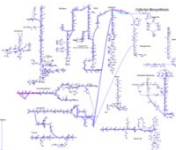
23S rRNA pseudouridine synthase
b0058, b1269, b1086, b2594, b1135, b4022

16S rRNA pseudouridine synthase
B2183

30S ribosomal maturation
b2566, b2608, b0436, b1480

30S ribosome binding factor
b3167

See Ribosome.ipynb



Ribosome Assembly Reactions: "formation_ribosome"

30S ribosomal subunit RNA

generic_16s_rRNAs,

30S ribosomal subunit proteins

RpsA_mono, RpsB_mono, RpsC_mono, RpsD_mono, RpsE_mono,
RpsF_mono, RpsG_mono, RpsH_mono, RpsI_mono, RpsJ_mono,
RpsK_mono, RpsL_mono, RpsM_mono, RpsN_mono, RpsO_mono,
RpsP_mono, RpsQ_mono, RpsR_mono, RpsS_mono, RpsT_mono,
RpsU_mono

50S ribosomal subunit RNA

generic_23s_rRNAs, generic_5s_rRNAs,

50S ribosomal subunit proteins

RplA_mono, RplB_mono, RplC_mono, RplD_mono, RplE_mono, RplF_mono,
rplL7/12_mod_1:acetyl, RplJ_mono, RplI_mono, RplJ_mono, RplK_mono,
RplL_mono, RplM_mono, RplN_mono, RplO_mono, RplP_mono,
RplQ_mono, RplR_mono, RplS_mono, RplT_mono, RplU_mono,
RplV_mono, RplW_mono, RplX_mono, RplY_mono, RpmA_mono,
RpmB_mono, RpmC_mono, RpmD_mono, RpmE_mono, RpmF_mono,
RpmG_mono, RpmH_mono, RpmI_mono, RpmJ_mono

16S rRNA methyltransferases

KsgA_mono, RsmB_mono, RsmC_mono, RsmD_mono, YggJ_dim (rsmE), RsmF_mono,
RsmG_mono,

23S rRNA methyltransferases

RrmA_dim_mod_2:zn2, RlmB_dim, RumB_mono_mod_1:4fe4s, RumA_mono_mod_1:4fe4s,
RrmJ_mono, RlmF_mono, RlmG_mono, RlmH_dim, RlmI_dim, RlmL_dim, RlmM_mono,
RlmN_mono_mod_1:4fe4s,

23S rRNA pseudouridine synthase

RluA_mono, RluB_mono, RluC_mono, RluD_mono_mod_1:mg2, YmfC_mono, YjbC_mono
(rluF),

16S rRNA pseudouridine synthase

RsuA_mono,

30S ribosomal maturation

Era_dim, RimM_mono, Tig_mono, Sra_mono,

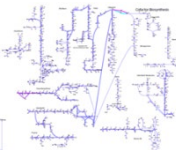
30S ribosome binding factor

RbfA_mono,

Metabolites

gtp_c, amet_c, nadh_c, h2o_c, mg2_c

See Ribosome.ipynb

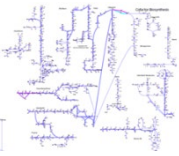


Ribosome Reaction (formation_reaction)

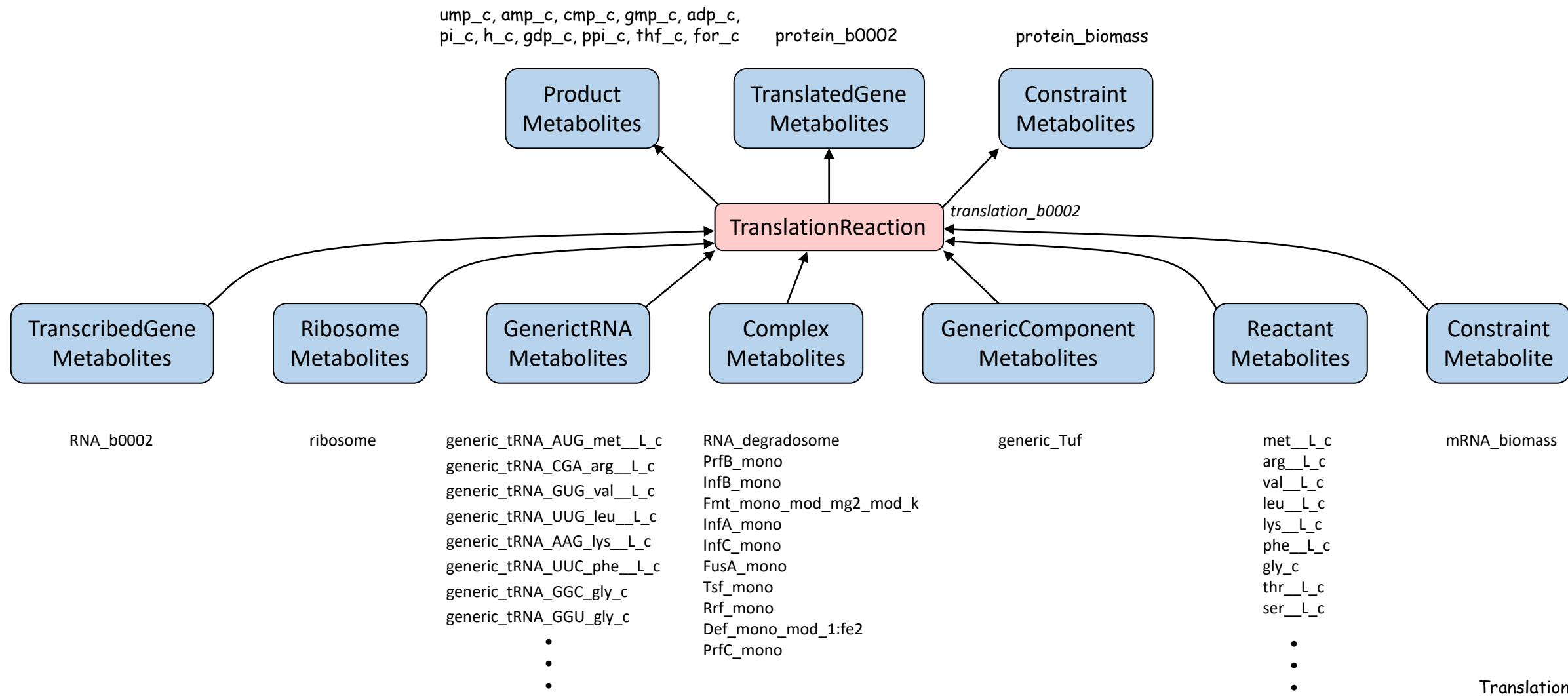
$4.27350427350427e-6 \mu \text{Era_dim} + 8.54700854700855e-6 \mu \text{KsgA_mono} + 4.27350427350427e-6 \mu \text{RbfA_mono} + 4.27350427350427e-6 \mu \text{RimM_mono} + 4.27350427350427e-6 \mu \text{RlmB_dim} + 4.27350427350427e-6 \mu \text{RlmF_mono} + 4.27350427350427e-6 \mu \text{RlmG_mono} + 4.27350427350427e-6 \mu \text{RlmH_dim} + 4.27350427350427e-6 \mu \text{RlmI_dim} + 8.54700854700855e-6 \mu \text{RlmL_dim} + 4.27350427350427e-6 \mu \text{RlmM_mono} + 4.27350427350427e-6 \mu \text{RlmN_mono_mod_1:4fe4s} + 4.27350427350427e-6 \mu \text{RluA_mono} + 4.27350427350427e-6 \mu \text{RluB_mono} + 1.28205128205128e-5 \mu \text{RluC_mono} + 1.28205128205128e-5 \mu \text{RluD_mono_mod_1:mg2} + \text{RplA_mono} + \text{RplB_mono} + \text{RplC_mono} + \text{RplD_mono} + \text{RplE_mono} + \text{RplF_mono} + \text{RplI_mono} + \text{RplJ_mono} + \text{RplK_mono} + \text{RplM_mono} + \text{RplN_mono} + \text{RplO_mono} + \text{RplP_mono} + \text{RplQ_mono} + \text{RplR_mono} + \text{RplS_mono} + \text{RplT_mono} + \text{RplU_mono} + \text{RplV_mono} + \text{RplW_mono} + \text{RplX_mono} + \text{RplY_mono} + \text{RpmA_mono} + \text{RpmB_mono} + \text{RpmC_mono} + \text{RpmD_mono} + \text{RpmE_mono} + \text{RpmF_mono} + \text{RpmG_mono} + \text{RpmH_mono} + \text{RpmI_mono} + \text{RpmJ_mono} + \text{RpsA_mono} + \text{RpsB_mono} + \text{RpsC_mono} + \text{RpsD_mono} + \text{RpsE_mono} + \text{RpsF_mono} + \text{RpsG_mono} + \text{RpsH_mono} + \text{RpsI_mono} + \text{RpsJ_mono} + \text{RpsK_mono} + \text{RpsL_mono} + \text{RpsM_mono} + \text{RpsN_mono} + \text{RpsO_mono} + \text{RpsP_mono} + \text{RpsQ_mono} + \text{RpsR_mono} + \text{RpsS_mono} + \text{RpsT_mono} + \text{RpsU_mono} + 4.27350427350427e-6 \mu \text{RrmA_dim_mod_2:zn2} + 4.27350427350427e-6 \mu \text{RrmJ_mono} + 4.27350427350427e-6 \mu \text{RsmB_mono} + 4.27350427350427e-6 \mu \text{RsmC_mono} + 4.27350427350427e-6 \mu \text{RsmD_mono} + 4.27350427350427e-6 \mu \text{RsmF_mono} + 4.27350427350427e-6 \mu \text{RsmG_mono} + 4.27350427350427e-6 \mu \text{RsuA_mono} + 4.27350427350427e-6 \mu \text{RumA_mono_mod_1:4fe4s} + 4.27350427350427e-6 \mu \text{RumB_mono_mod_1:4fe4s} + \text{Sra_mono} + \text{Tig_mono} + 4.27350427350427e-6 \mu \text{YggJ_dim} + 4.27350427350427e-6 \mu \text{YjbC_mono} + 4.27350427350427e-6 \mu \text{YmfC_mono} + 27.0 \text{amet_c} + 4.27350427350427e-6 \mu \text{generic_16Sm4Cm1402} + \text{generic_16s_rRNAs} + \text{generic_23s_rRNAs} + \text{generic_5s_rRNAs} + 3.0 \text{gtp_c} + 2.0 \text{h2o_c} + 171.0 \text{mg2_c} + \text{nadh_c} + 2.0 \text{rpL7/12_mod_1:acetyl}$

$\rightarrow 27.0 \text{ahcys_c} + 2.0 \text{gdp_c} + 28.0 \text{h_c} + \text{nad_c} + 2.0 \text{pi_c} + 4.5368885400000005 \text{prosthetic_group_biomass} + \text{ribosome}$

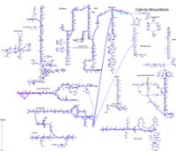
See Ribosome.ipynb



Example: iJL1678b (ECOLIme) RNA_b0002 Translation



Translation.ipynb



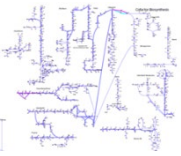
Translation Supporting Metabolites

- RNA_degradosome: Degradation of mRNA
- InfA_mono: translation initiation factor IF-1
- InfB_mono: translation initiation factor IF-2 β '
- InfC_mono: translation initiation factor IF-3
- PrfB_mono: peptide chain release factor RF2
- PrfC_mono: peptide chain release factor RF3
- Fmt_mono_mod_mg2_mod_k: 10-formyltetrahydrofolate:L-methionyl-tRNA^{fMet} N-formyltransferase
- FusA_mono: elongation factor G
- Tsf_mono: protein chain elongation factor EF-Ts
- Rrf_mono: ribosome-recycling factor
- Def_mono_mod_1:fe2: peptide deformylase



Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - Macromolecular Reactions
 - Transcription
 - Translation
 - • COBRAMe Execution
 - ECOLIme Operation



Executing the iJL1678b model

```
from __future__ import print_function, division, absolute_import

# python imports
import re

from os.path import join

from collections import defaultdict

import pickle

import pandas as pd

# third party imports
import tabulate

import cobra

# ECOLIme
import ecolime

from ecolime import (transcription, translation, flat_files, generics, formulas, compartments)

# COBRAMe
import cobrame

from cobrame.util import building, mu, me_model_interface
```

```
# Load the iJL1678b model
with open('iJL1678b.pickle', 'rb') as f:
    me = pickle.load(f)

# Methods
def solve_me_model(me, max_mu, precision=1e-6, min_mu=0, using_soplex=False,
                   compiled_expressions=None):
    if using_soplex:
        from cobrame.solve.algorithms import binary_search
        binary_search(me, min_mu=min_mu, max_mu=max_mu, debug=True, mu_accuracy=precision,
                      compiled_expressions=compiled_expressions)
    else:
        from qminospy.me1 import ME_NLP1
        # The object containing solveME methods--composite that uses a ME model object
        me_nlp = ME_NLP1(me, growth_key='mu')
        # Use bisection for now (until the NLP formulation is worked out)
        muopt, hs, xopt, cache = me_nlp.bisectmu(precision=precision, mumax=max_mu)
        me.solution.f = me.solution.x_dict['biomass_dilution']
```

```
solve_me_model(me, 1., min_mu = .1, precision=1e-2, using_soplex=False)
```

[solve_demo.ipynb](#)



solve_demo.ipynb

```
import pickle
import cobrame
from cobrame.io.json import load_reduced_json_me_model, load_json_me_model
```

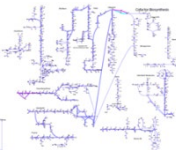
```
with open('./me_models/iJL1678b.pickle', 'rb') as f:
    me = pickle.load(f)
```

```
def solve_me_model(me, max_mu, precision=1e-6, min_mu=0, using_soplex=True,
                   compiled_expressions=None):
    if using_soplex:
        from cobrame.solve.algorithms import binary_search
        binary_search(me, min_mu=min_mu, max_mu=max_mu, debug=True, mu_accuracy=precision,
                      compiled_expressions=compiled_expressions)
    else:
        from qminospy.me1 import ME_NLP1
        # The object containing solveME methods--composite that uses a ME model object
        me_nlp = ME_NLP1(me, growth_key='mu')
        # Use bisection for now (until the NLP formulation is worked out)
        muopt, hs, xopt, cache = me_nlp.bisectmu(precision=precision, mumax=max_mu)
        me.solution.f = me.solution.x_dict['biomass_dilution']
```

```
def show_escher_map(me, solution=None):
    import escher
    view = escher.Builder("iJ01366.Central metabolism")
    view.reaction_data = me.get_metabolic_flux(solution=solution)
    return view
```

```
solve_me_model(me, 1., min_mu = .1, precision=1e-2, using_soplex=False)
```

```
# Display metabolic flux on escher map
show_escher_map(me).display_in_notebook()
```



Solving a COBRaMe Model

```
import pickle
import cobrame
from cobrame.io.json import load_reduced_json_me_model, load_json_me_model

with open('./me_models/iJL1678b.pickle', 'rb') as f:
    me = pickle.load(f)

def solve_me_model(me, max_mu, precision=1e-6, min_mu=0, using_soplex=True,
                  compiled_expressions=None):
    if using_soplex:
        from cobrame.solve.algorithms import binary_search
        binary_search(me, min_mu=min_mu, max_mu=max_mu, debug=True, mu_accuracy=precision,
                      compiled_expressions=compiled_expressions)
    else:
        from qminosp.py.me1 import ME_NLP1
        # The object containing solveME methods--composite that uses a ME model object
        me_nlp = ME_NLP1(me, growth_key='mu')
        # Use bisection for now (until the NLP formulation is worked out)
        muopt, hs, xopt, cache = me_nlp.bisectmu(precision=precision, mumax=max_mu)
        me.solution.f = me.solution.x_dict['biomass_dilution']

def show_escher_map(me, solution=None):
    import escher
    view = escher.Builder("iJ01366.Central metabolism")
    view.reaction_data = me.get_metabolic_flux(solution=solution)
    return view
```

solve_demo.ipynb

```
solve_me_model(me, 1., min_mu = .1, precision=1e-2, using_soplex=False)
```

```
iter    muopt      a      b      mu1      stat1
Finished compiling expressions in 67.532865 seconds
Finished substituting S,lb,ub in 4.933790 seconds
Finished makeME_LP in 0.852235 seconds
Getting MINOS parameters from ME_NLP...
1 0.5 0.5 1.0 0.5 optimal
Finished substituting S,lb,ub in 5.459582 seconds
Finished makeME_LP in 0.926348 seconds
Getting MINOS parameters from ME_NLP...
2 0.75 0.75 1.0 0.75 optimal
Finished substituting S,lb,ub in 5.111826 seconds
Finished makeME_LP in 0.894545 seconds
Getting MINOS parameters from ME_NLP...
3 0.75 0.75 0.875 0.875 1
Finished substituting S,lb,ub in 4.831290 seconds
Finished makeME_LP in 0.847434 seconds
Getting MINOS parameters from ME_NLP...
4 0.8125 0.8125 0.875 0.8125 optimal
Finished substituting S,lb,ub in 4.792310 seconds
Finished makeME_LP in 0.844745 seconds
Getting MINOS parameters from ME_NLP...
5 0.8125 0.8125 0.84375 0.84375 1
Finished substituting S,lb,ub in 4.794153 seconds
Finished makeME_LP in 0.848020 seconds
Getting MINOS parameters from ME_NLP...
6 0.828125 0.828125 0.84375 0.828125 optimal
Finished substituting S,lb,ub in 5.176051 seconds
Finished makeME_LP in 0.852741 seconds
Getting MINOS parameters from ME_NLP...
7 0.828125 0.828125 0.8359375 0.8359375 1
Bisection done in 242.881 seconds
```

Escher Plot of COBRAME Simulation

```
import pickle
import cobrame
from cobrame.io.json import load_reduced_json_me_model, load_json_me_model

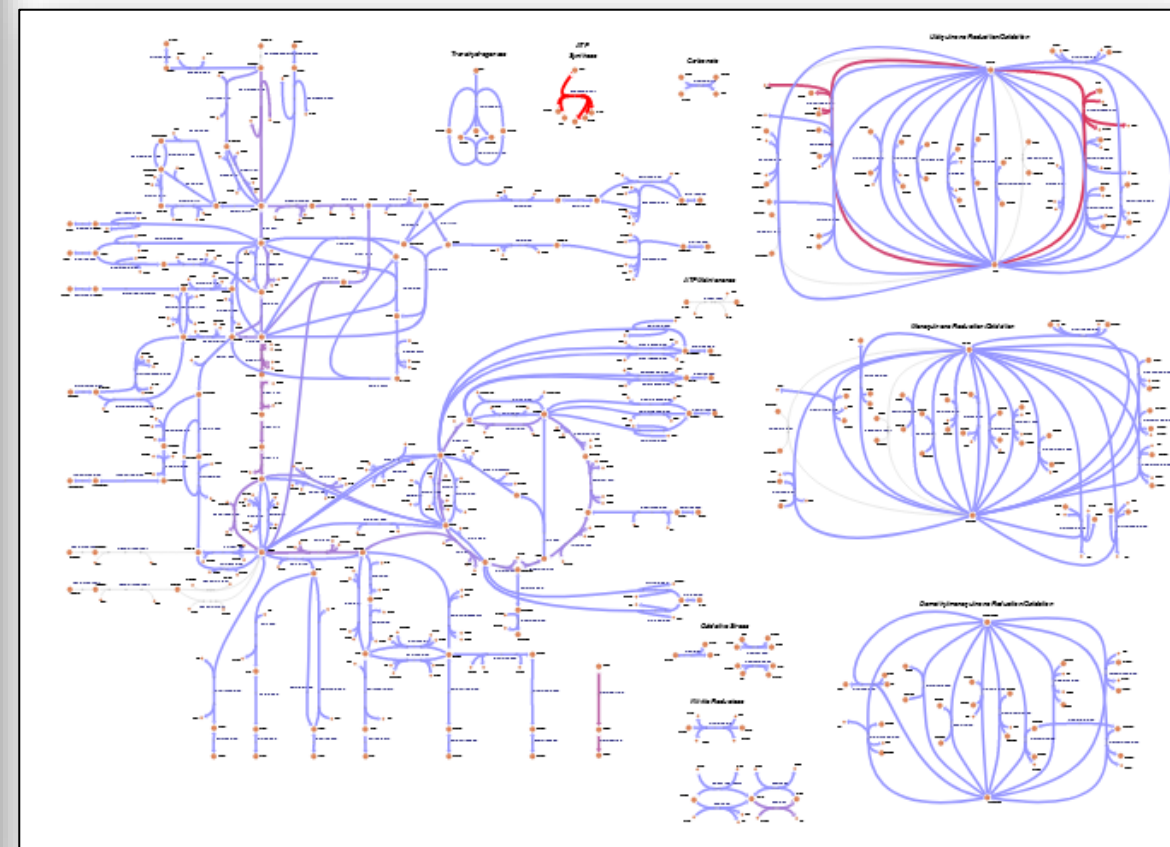
with open('./me_models/iJL1678b.pickle', 'rb') as f:
    me = pickle.load(f)

def solve_me_model(me, max_mu, precision=1e-6, min_mu=0, using_soplex=True,
                    compiled_expressions=None):
    if using_soplex:
        from cobrame.solve.algorithms import binary_search
        binary_search(me, min_mu=min_mu, max_mu=max_mu, debug=True, mu_accuracy=precision,
                      compiled_expressions=compiled_expressions)
    else:
        from qminospy.me1 import ME_NLP1
        # The object containing solveME methods--composite that uses a ME model object
        me_nlp = ME_NLP1(me, growth_key='mu')
        # Use bisection for now (until the NLP formulation is worked out)
        muopt, hs, xopt, cache = me_nlp.bisectmu(precision=precision, mumax=max_mu)
        me.solution.f = me.solution.x_dict['biomass_dilution']

def show_escher_map(me, solution=None):
    import escher
    view = escher.Builder("iJ01366.Central metabolism")
    view.reaction_data = me.get_metabolic_flux(solution=solution)
    return view

solve_me_model(me, 1., min_mu = .1, precision=1e-2, using_soplex=False)

# Display metabolic flux on escher map
show_escher_map(me).display_in_notebook()
```

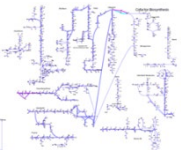


solve_demo.ipynb

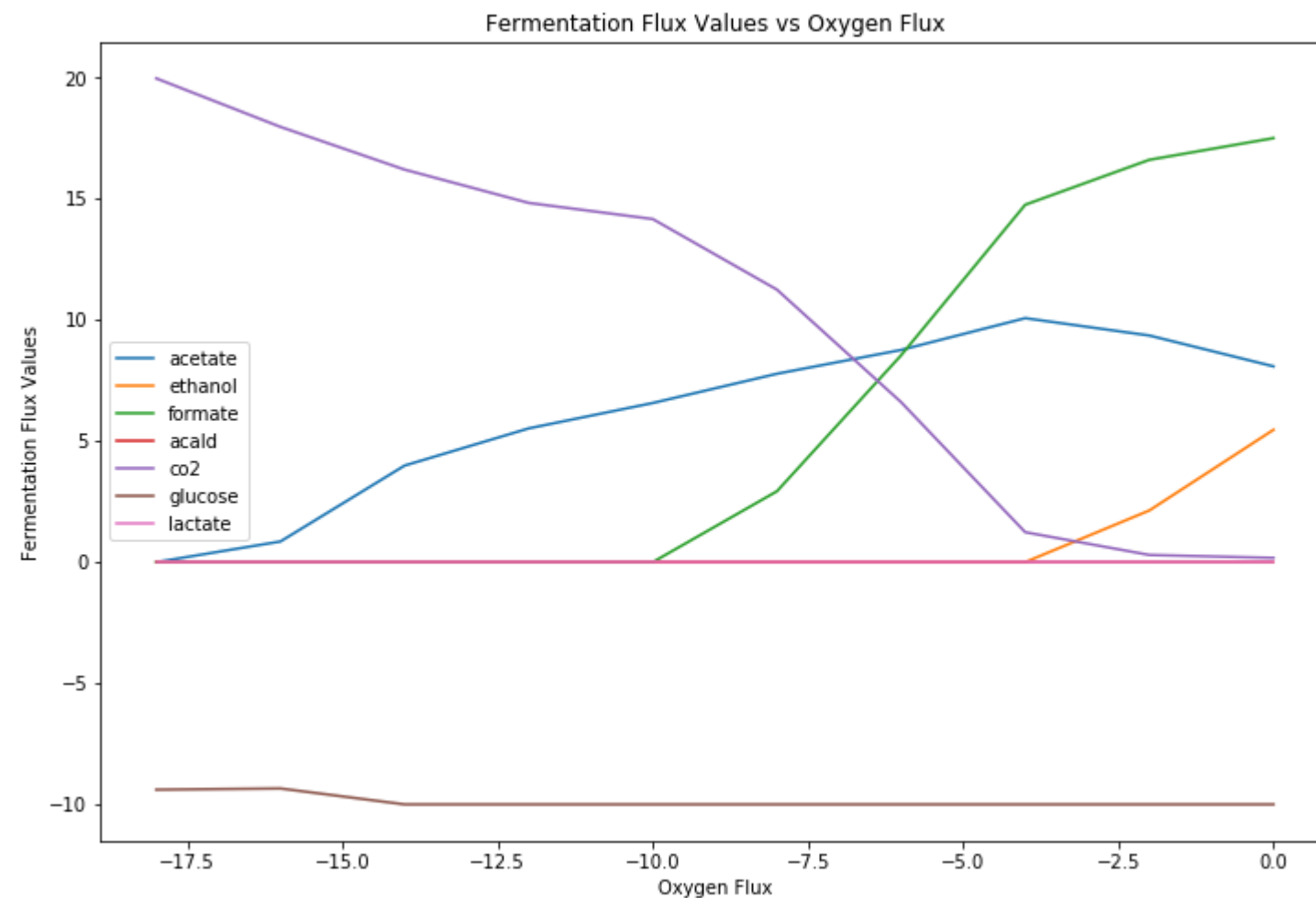
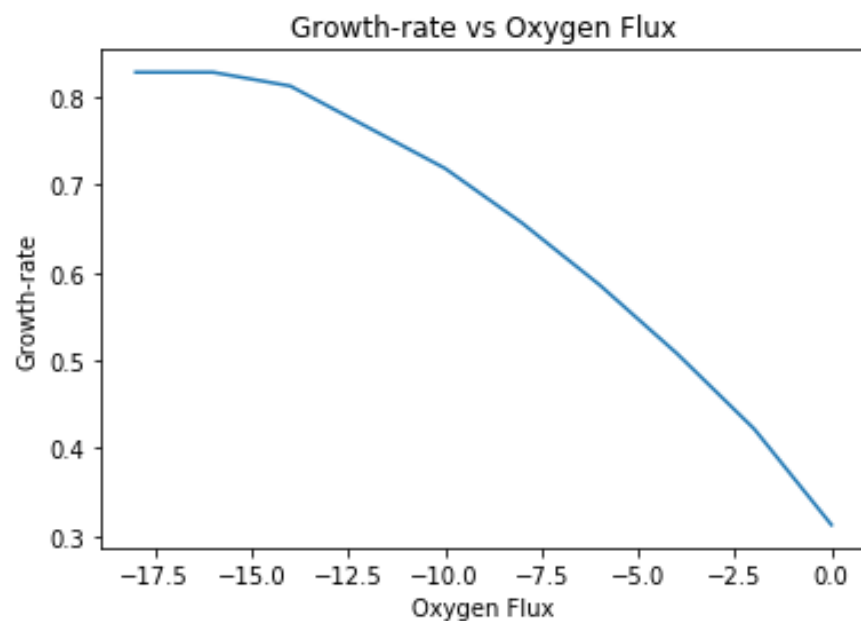


Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRAMe
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLIme
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRAMe Execution
 - • ECOLIme Operation



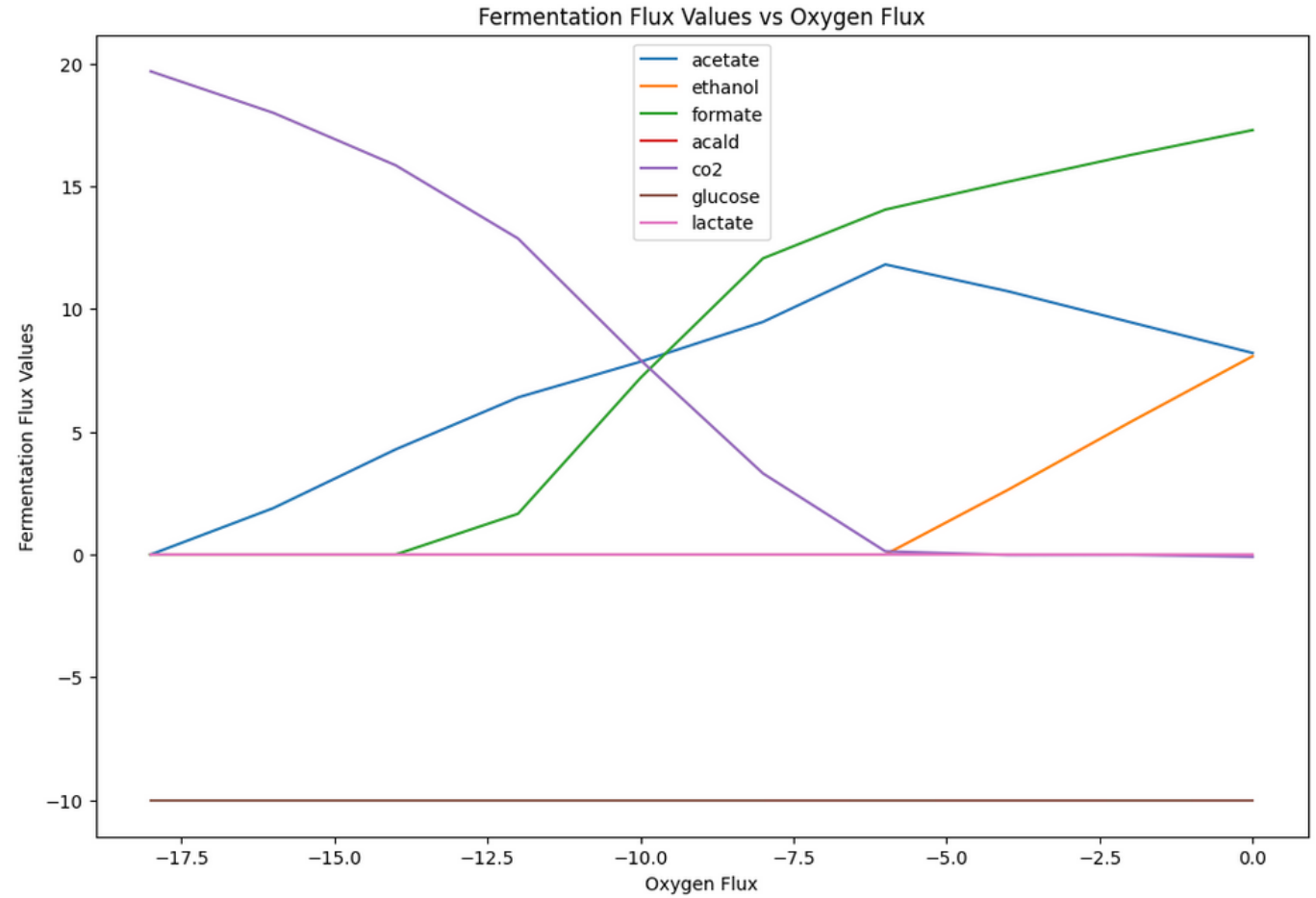
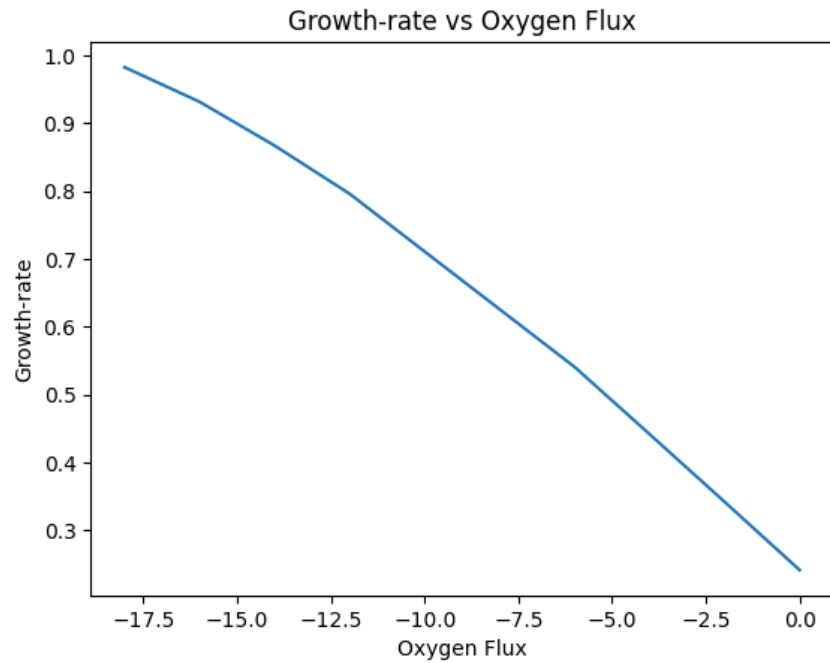
iJL1678b Fermentation

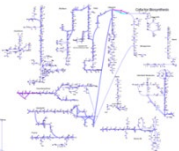


`me.reactions.get_by_id('EX_glc__D_e').lower_bound = -10`

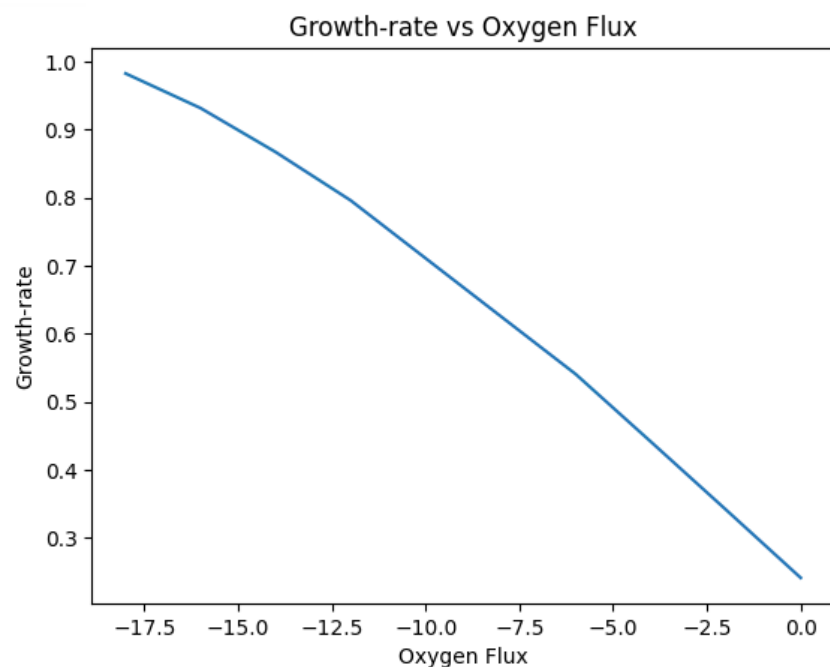


iJO1366 Fermentation

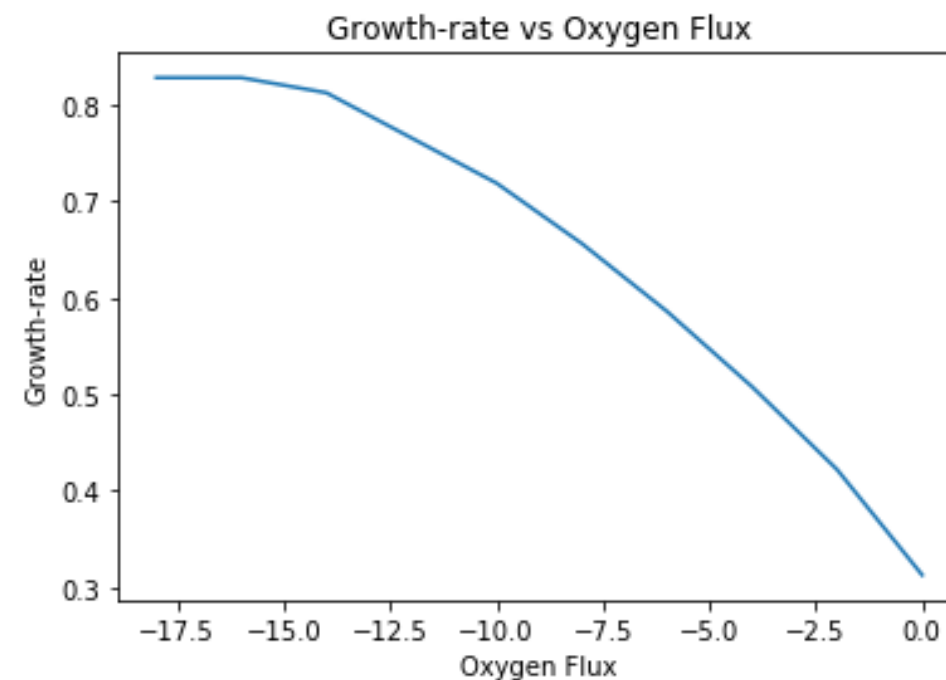




iJO1366 vs iJL1678b Growth-rate

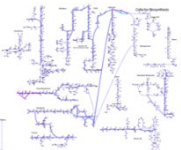


iJO1366

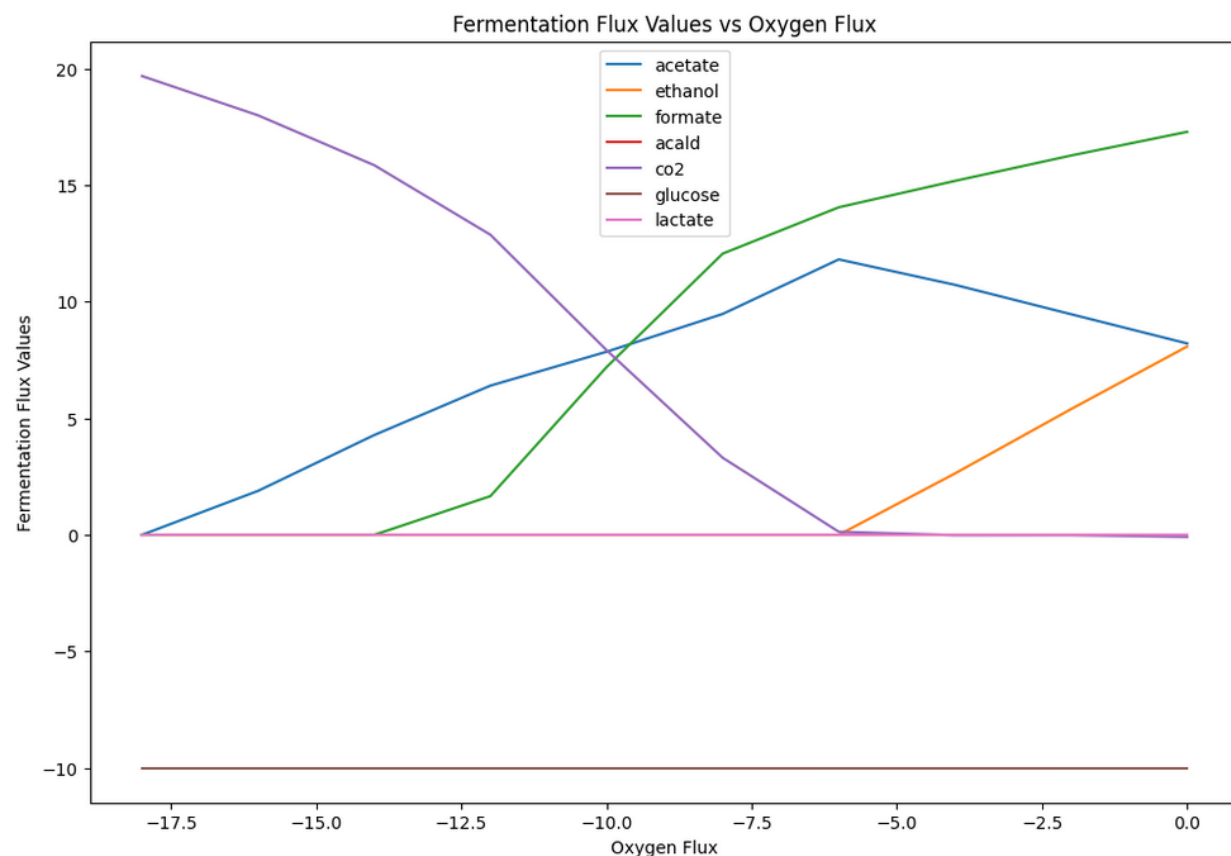


iJL1678b

[Oxygen_Impact_iJO13166.ipynb](#)[Oxygen_Impact_iJL1678b.ipynb](#)

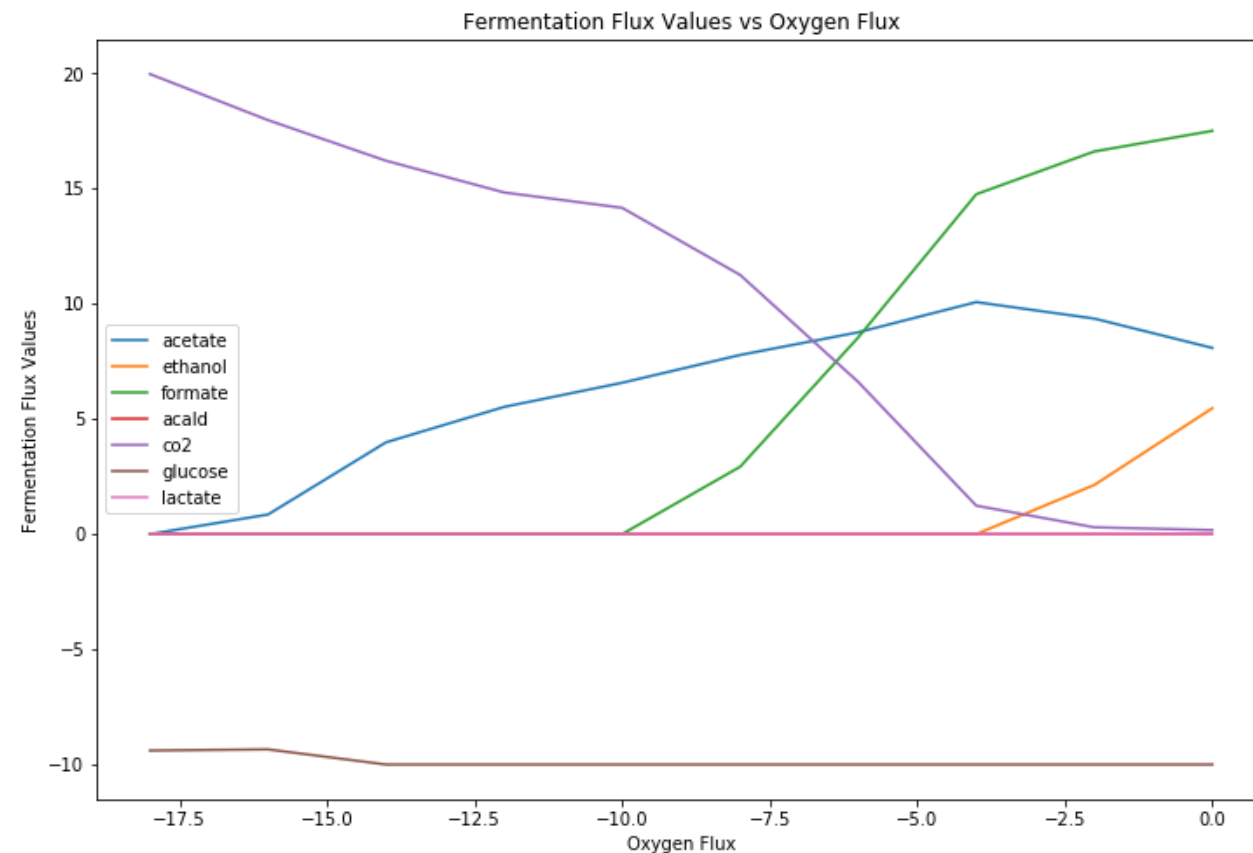


iJO1366 vs iJL1678b Oxygen Impact



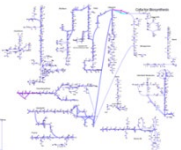
iJO1366

Oxygen_Impact_iJO13166.ipynb

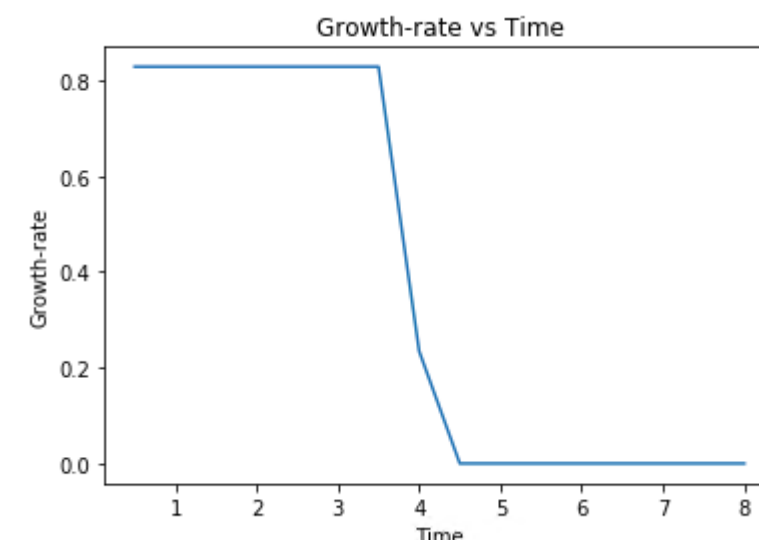
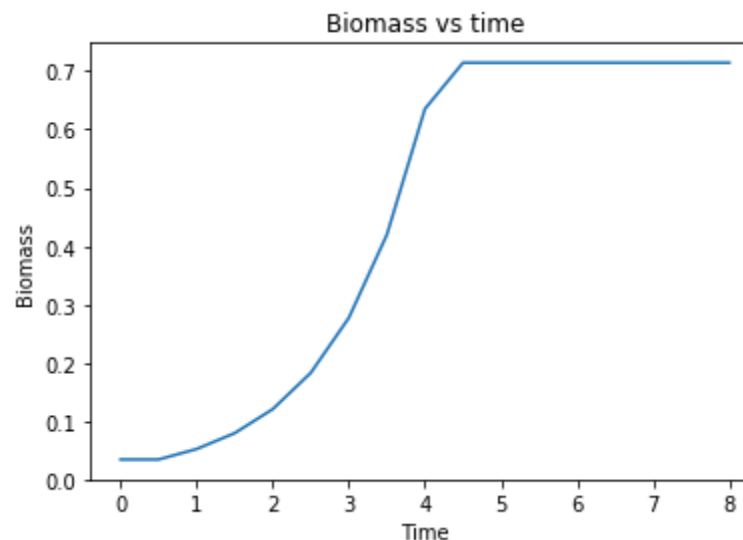
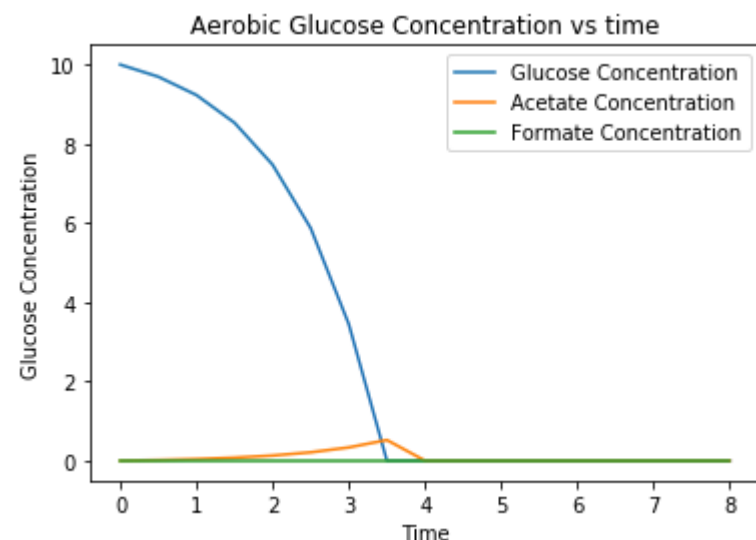


iJL1678b

Oxygen_Impact_iJL1678b.ipynb



iJL1678b Dynamic Flux Balance Analysis (dFBA) (Aerobic Growth)



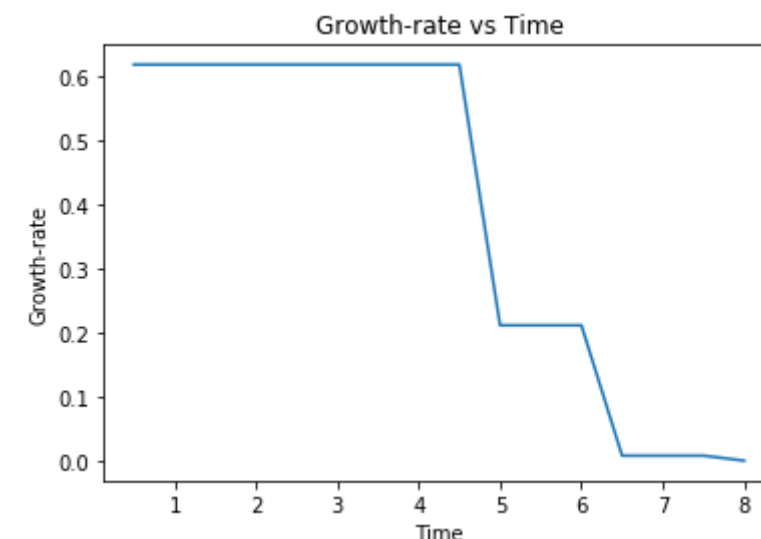
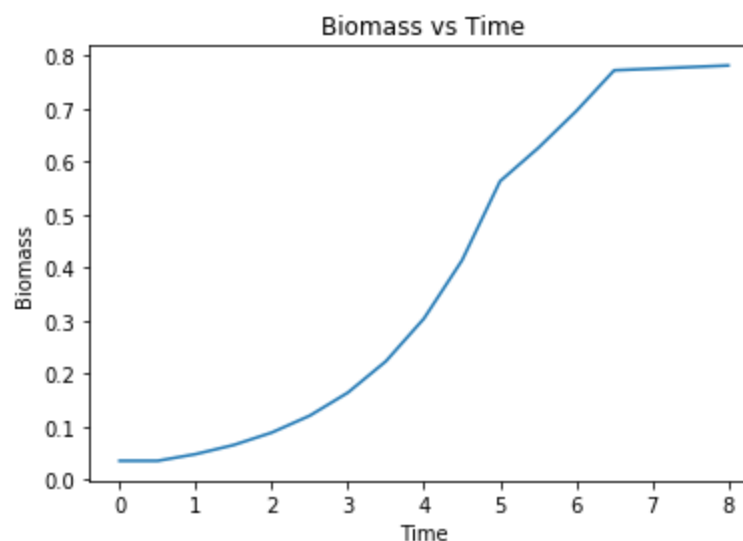
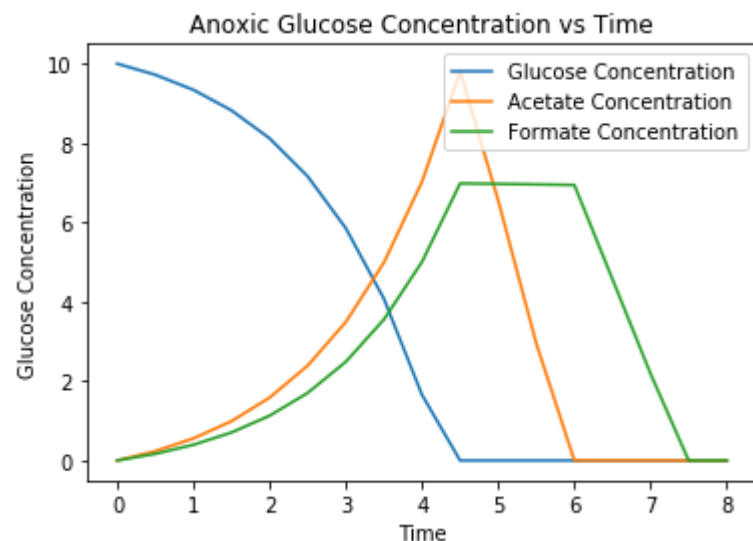
```
model.reactions.EX_o2_e.lower_bound = -20
model.reactions.EX_ac_e.lower_bound = -10
model.reactions.EX_for_e.lower_bound = -10
```

Initial Constants

```
biomass_0 = 0.035 # Initial biomass
biomass_t = biomass_0
time_total = 8 # Hours
samples = 16 # Total number of samples
```

dFBA_COBRame_Aerobic.ipynb

iJL1678b Dynamic Flux Balance Analysis (dFBA) (Anoxic Growth)

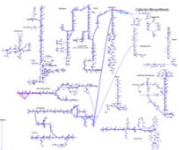


```
model.reactions.EX_glc__D_e.lower_bound = -10
model.reactions.EX_o2_e.lower_bound = -7
model.reactions.EX_ac_e.lower_bound = -10
model.reactions.EX_for_e.lower_bound = -10
```

Initial Constants

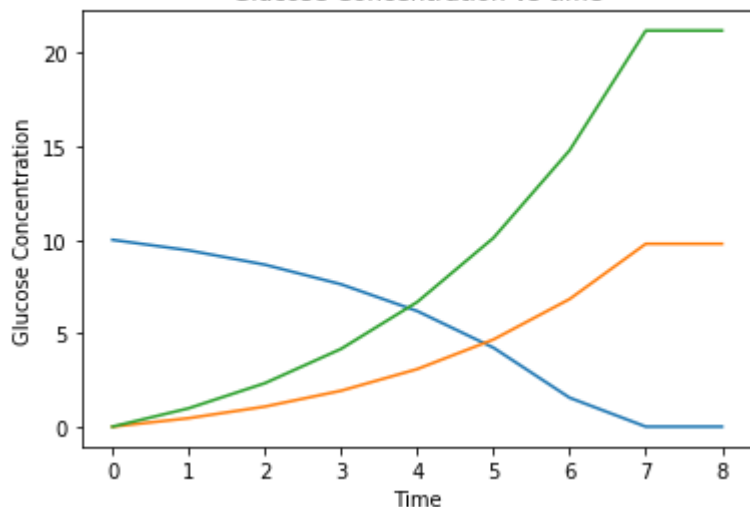
```
biomass_0 = 0.035 # Initial biomass
biomass_t = biomass_0
time_total = 8 # Hours
samples = 16 # Total number of samples
```

dFBA_COBRAME_Anoxic.ipynb

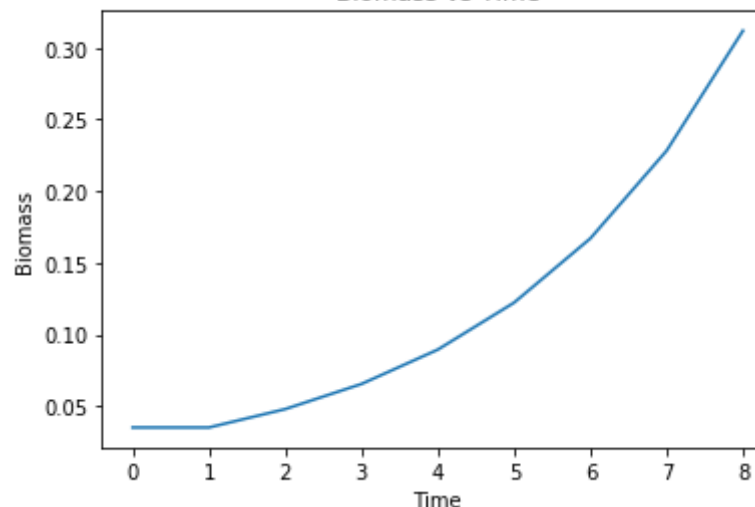


iJL1678b Dynamic Flux Balance Analysis (dFBA) (Anaerobic Growth)

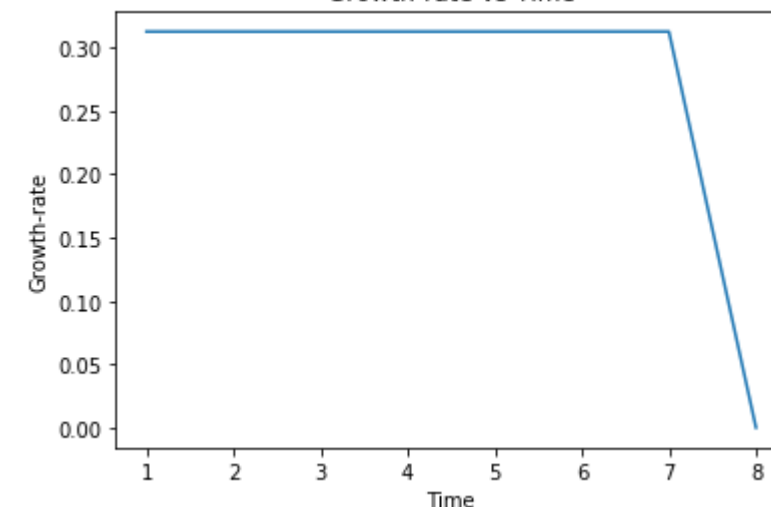
Glucose Concentration vs time



Biomass vs Time



Growth-rate vs Time



```
model.reactions.EX_glc__D_e.lower_bound = -10  
model.reactions.EX_o2_e.lower_bound = 0  
model.reactions.EX_ac_e.lower_bound = -10  
model.reactions.EX_for_e.lower_bound = -10
```

Initial Constants

```
biomass_0 = 0.035 # Initial biomass  
biomass_t = biomass_0  
time_total = 8 # Hours  
samples = 8 # Total number of samples
```



Lesson Outline

- Metabolism (M) vs. Metabolism & Expression (ME) Models
- COBRaME
 - Working Environment (Docker)
 - ME Model Architecture
 - Macromolecular Coupling
 - Reaction Lumping
- ECOLiME
 - Macromolecular Reactions
 - Transcription
 - Translation
 - COBRaME Execution
 - ECOLiME Operation