`meneco`
User Guide
version 1.5.0

Sven Thiele

1 What is `meneco`?

`meneco` is a tool for metabolic network completion. Within a qualitative approach that describes the biosynthetic capacities of metabolic networks, `meneco` uses qualitative constraints to express the producibility for a set of metabolites. `meneco` can be used to check whether a network provides the synthesis routes to comply with the required functionality described by the producibility constraints. In particular it tests whether it is possible to synthesize so called target metabolites from a set of seed metabolites. For networks that fail this test `meneco` can attempt to complete the network by importing reactions from a metabolic reference network such that the resulting network provides the required functionality. `meneco` can identify unproducible target metabolites and computes minimal extensions to the network that satisfy the producibility constraints. Additionally, it can compute the union and intersection of all minimal networks extensions without enumerating all minimal network extensions.

2 Prerequisites

`meneco` is a Python application that uses the power of answer set solving technology to compute its solution. Therefore it depends on the `PyASP` library, a python wrapper for the solvers from the Potassco Answer Set Solving Collection. All dependencies are automatically resolved and installed via `pip`, the recommended package installer for the Python Package index where the software is hosted. The `PyASP` package detects the running system and installs the matching binaries of the answer set grounder `gringo` and solver `clasp`. `meneco` runs on the Linux and Mac OS operating systems Windows is currently **not** supported.

3 Installation

3.1 Installation using `pip`

You can install the `meneco` package by running:

```
$ pip install --user meneco
```

On Linux the executable scripts can then be found in `/.local/bin`
and on Mac OS the scripts are under `/Users/YOURUSERNAME/Library/Python/3.2/bin`.

3.2 Installation of `pip`

If `pip` is not installed one can install `pip` without admin rights. Therefore one has to first download `getpip.py` via:

```
$ wget https://raw.github.com/pypa/pip/master/contrib/get-pip.py
```

and then install pip locally:

```
$ python get-pip.py --user
```

Now it is possible to use the local pip and proceed with the section *Installation of `pip`*.

3.3 Installation without `pip`

Although we do not recommend it, it is possible to install `meneco` without `pip`. But then one has to take care of the dependencies oneself. Therefore one has to first download `pyasp-1.4.2`.

```
$ wget https://pypi.python.org/packages/source/p/pyasp/pyasp-1.4.2.tar.gz
```

Next one has to extract and install `PyASP`:

```
$ gzip -d pyasp-1.4.2.tar.gz
$ tar -xvf pyasp-1.4.2.tar
$ cd pyasp-1.4.2
$ python setup.py install --user
```

Then one needs to download `meneco-1.5.0`:

```
$ wget https://pypi.python.org/packages/source/m/meneco/meneco-1.5.0.tar.gz
```

and to extract and install meneco:

```
$ gzip -d meneco-1.5.0.tar.gz
$ tar -xvf meneco-1.5.0.tar
$ cd meneco-1.5.0
$ python setup.py install --user
```

On Linux the executable scripts can then be found in `/.local/bin`
and on Mac OS the scripts are under `/Users/YOURUSERNAME/Library/Python/3.2/bin`.

4 Usage

Typical usage is:

```
$ meneco.py -d draftnet.sbml -s seeds.sbml -t targets.sbml -r repairdb.sbml
```

For more options you can ask for help as follows:

```
$ meneco.py --h
meneco.py [-h] -d DRAFTNET -s SEEDS -t TARGETS [-r REPAIRNET]
                 [--enumerate]

optional arguments:
  -h, --help            show this help message and exit
  -d DRAFTNET, --draftnet DRAFTNET
                        metabolic network in SBML format
  -s SEEDS, --seeds SEEDS
                        seeds in SBML format
  -t TARGETS, --targets TARGETS
                        targets in SBML format
  -r REPAIRNET, --repairnet REPAIRNET
                        perform network completion using REPAIRNET a metabolic
                        network in SBML format
  --enumerate           enumerate all minimal completions
```

5 Input

`meneco` works with two kinds of data. The first is metabolic reaction data representing metabolic reactions from the draft network and the reference database (i.e. the metacyc db). The second is the information about seed and target metabolites.

5.1 Metabolic reaction data

The metabolic reaction data must be presented in Systems Biology Markup Language format `SBML` as shown below.

Here is how one could represent a metabolic reaction network in SBML:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml level="2" version="3" xmlns="http://www.sbml.org/sbml/level2/version3">
    <model name="EnzymaticReaction">
        <listOfCompartments>
            <compartment id="cytosol" size="1e-14"/>
        </listOfCompartments>
        <listOfSpecies>
            <species compartment="cytosol" id="ES" initialAmount="0"     name="ES"/>
            <species compartment="cytosol" id="P"  initialAmount="0"     name="P"/>
            <species compartment="cytosol" id="S"  initialAmount="1e-20" name="S"/>
            <species compartment="cytosol" id="E"  initialAmount="5e-21" name="E"/>
        </listOfSpecies>
        <listOfReactions>
            <reaction id="veq">
                <listOfReactants>
                    <speciesReference species="E"/>
                    <speciesReference species="S"/>
                </listOfReactants>
                <listOfProducts>
                    <speciesReference species="ES"/>
                </listOfProducts>
            </reaction>
            <reaction id="vcat" reversible="true">
                <listOfReactants>
                    <speciesReference species="ES"/>
                </listOfReactants>
                <listOfProducts>
                    <speciesReference species="E"/>
                    <speciesReference species="P"/>
                </listOfProducts>
            </reaction>
        </listOfReactions>
    </model>
</sbml>
```

In this example, the model has the identifier `EnzymaticReaction`. The model contains one compartment (with identifier `cytosol`), four metabolites (with species identifiers `ES`, `P`, `S`, and `E`), and two reactions (`veq` and `vcat`). The elements in the `listOfReactants` and `listOfProducts` in each reaction refer to the names of elements listed in the `listOfSpecies`. The correspondences between the various elements is explicitly stated by the `speciesReference` elements. The reaction `vcat` has the attribute `reversible` set to `true`. Note that `meneco` will only treat a reaction as reversible if this attribute is set to `true`. If the attribute is not set, the default assumption is that a reaction is irreversible. Thus, the reaction `veq` is treated as irreversible. The SBML file may contain additional informations (like `initialAmount="1e-20"`) which will be ignored by `meneco`.

5.2 Seed and Target data

Also information about seed and target metabolites can be specified using SBML files. For convenience reasons `meneco` takes two files one which presents the specification of the seed metabolites and one file for the target metabolites. Here is how such a file could look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml level="2" version="3" xmlns="http://www.sbml.org/sbml/level2/version3">
<model id="enzreact_ntrexp_001" name="set of seed metabolites 1">
<listOfSpecies>
        <species id="E" name="This crazy metabolite E"/>
        <species id="S" name="This funky metabolite S"/>
</listOfSpecies>
</model>
</sbml>
```

In this example two metabolites (with identifiers E and S) are specified. It is important that these identifiers match with the identifiers in the description of the reaction network. Depending on whether this description is passed as 3rd or 4th argument `meneco` regards the specified metabolites as seeds resp. targets.

6 Output

`meneco` presents the results of its analysis as text output. The output of `meneco` can be redirected into a file using the > operator. For example to write the results shown below into the file `myfile.txt` type:

```
$ meneco.py -d draftnet.sbml -s seeds.sbml -t targets.sbml -r repairdb.sbml --enumerate > myfile.txt
```

In the following we will dissect the output generated by `meneco`. The first 3 lines document which files have been used as representation of the metabolic reaction network (line 1) and which files to specify the seed (line 2) and targets (line 3).

```
1  Reading draft network from  data/draftnet.sbml ... done.
2  Reading seeds from  data/seeds.sbml ... done.
3  Reading targets from  data/targets.sbml ... done.
```

In a next step `meneco` tests which target metabolite are producible given the seeds and the reactions of the draft network. In this case the result shows 7 target metabolites which are not producible.

```
4  Checking draftnet for unproducible targets ... done.
5    7 unproducible targets:
6    "M_pe161_c"
7    "M_so4_c"
8    "M_utp_c"
9    "M_thf_c"
10   "M_cu2_c"
11   "M_cl_c"
12   "M_ribflv_c"
```

Given that unproducible target metabolites exist the database with repair reactions is loaded. Line 13 documents that the reactions from `toy/repairdb.sbml` are used to find extensions to the original networks.

```
13  Reading repair network from  toy/repairdb.sbml ... done.
```

The next step, is to check which of the targets remain unproducible even with a the reactions from the repair database. In this example 5 metabolites remain unproducible (line 15), but for 2 targets `meneco` is able to repair the synthesis pathways (line 22-24).

```
14  Checking draftnet + repairnet for unproducible targets ... done.
15    still 5 unproducible targets:
16    "M_pe161_c"
17    "M_so4_c"
18    "M_utp_c"
19    "M_thf_c"
20    "M_ribflv_c"
21
22    2 targets to reconstruct:
23    "M_cu2_c"
24    "M_cl_c"
```

As a precomputation step `meneco` tries to reconstruct the production pathways for each reconstructable target metabolite, and computing the essential reactions. Essential reaction in this context are reactions which always must be added to allow a synthesis of the target from the seeds. This intermediate result can be reused in further computations as a solution that restores all targets must contain all reaction that are essential for each single target. In this example for each target metabolite exist one essential reaction.

```
25  Computing essential reactions for target("M_cu2_c") ... done.
26    1 essential reactions found:
27    "R_CU2tpp"
28
29  Computing essential reactions for target("M_cl_c") ... done.
30    1 essential reactions found:
31    "R_CLt3_2pp"
32
33  Overall 2 essential reactions found.
34    "R_CU2tpp"
35    "R_CLt3_2pp"
```

The next line states that for the following computation the essential reaction will be added to the network. The essential reactions are part of every solution and don't need to be recomputed.

```
36  Adding essential reactions to network.
```

The next step, is to compute one minimal completion to produce all targets. This is mainly done to get the size of a minimal completion. In this example the first minimal completion contains 3 reactions.

```
37  Computing one minimal completion to produce all targets ... done.
38    "R_CU2tpp"
39    "R_CLt3_2pp"
40    "R_O2tex"
```

Now that we know the size of a minimal completion, the intersection of all minimal completions is computed. This can be done without enumerating all completions. In our case we get 2 reactions in the intersection.

```
41  Computing common reactions in all completion with size 3 ... done.
42    "R_CU2tpp"
43    "R_CLt3_2pp"
```

Similarly, the union of all minimal completions is computed. Also this can be done without enumerating all completions. For this example the union contains 4 reactions, we can conclude that for our example has exactly two minimal completions.

```
44  Computing union of reactions from all completion with size 3 ... done.
45    "R_CU2tpp"
46    "R_CLt3_2pp"
47    "R_O2Stex"
48    "R_CU2tpp
```

Finally, if the command line option `--enumerate` has been given, all minimal completions are enumerated. Fortunately, our example has only two solutions. In general the number of solutions can be very high. Therfore, enumeration is only done if explicitly requested via command line option.

```
Computing all completions with size 3 ... done.
Completion 1:
  "R_CU2tpp"
  "R_CLt3_2pp"
  "R_O2Stex"
Completion 2:
  "R_CU2tpp"
  "R_CLt3_2pp"
  "R_O2tex"
```

References

[1] Guillaume Collet, Damien Eveillard, Martin Gebser, Sylvain Prigent, Torsten Schaub, Anne Siegel, and Sven Thiele. Extending the metabolic network of *ectocarpus siliculosus* using answer set programming. In Pedro Cabalar and Tran Cao Son, editors, *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 8148 of *Lecture Notes in Computer Science*, pages 245–256. Springer, 2013.

[2] Torsten Schaub and Sven Thiele. Metabolic network expansion with ASP. In Patricia M. Hill and David Scott Warren, editors, *Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09)*, volume 5649 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2009.