# modelMaGe Documentation

Jörg Schaber and Max Flöttmann

Humboldt University

Invaliden Str. 42

10115 Berlin

Germany

18 February 2010

Email: admin@modelmage.org

## Contents

*modelMaGe* Documentation ........ 1
Contents ........ 2
1 Introduction ........ 3
2 Installation and running *modelMaGe* ........ 4
2.1 Installation ........ 4
2.1.1 Linux/Mac OS X ........ 5
2.1.2 Windows ........ 5
2.2 Running *modelMaGe* ........ 6
3 Generating Model Alternatives ........ 6
3.1 Preliminaries: defining and inspecting the master model ........ 6
3.2 Removing Reactions ........ 12
3.3 Removing Species ........ 15
3.4 Inserting alternative kinetics ........ 18
3.5 Removing modifiers ........ 24
3.6 Using model alias and init-file ........ 25
4 Automatic parameter estimation and model discrimination ........ 27
5 Documentation of candidate models ........ 32
6 Web Interface ........ 33
6.1 Documentation of Code ........ 34
7 References ........ 35

# 1 Introduction

*modelMaGe* (Flöttmann, et al., 2008) automatically generates, fits and discriminates candidate models based on a master model and certain directives that specify how candidate models are to be derived from the master model. Such candidate models are created by two basic processes, first, by removing species, modifiers or reactions from the master model and, second, by introducing user-specified alternative kinetics for certain reactions.

The generated candidate models are automatically documented in a way that it is always comprehensible how they were derived from the master model, thereby keeping track of model alternatives.

The principle of generating candidate models by leaving out components of a master model implies that the master model must comprise all components of the candidate models.

The user has to provide only two things:

1) the structure of the master model in SBML or *Copasi* (Hoops, et al., 2006) format, with or without kinetics, and
2) directives for *modelMaGe* how to generate the candidate models.

When no initial kinetics are assigned, *modelMaGe* assumes mass action kinetics of appropriate order by default.

There is no need to edit the candidate models at any time. Thereby common parameters, modifications or directives are changed in one place only and are automatically updated in each candidate model. This is supposed to remove the errors introduced by modifying each candidate model individually. Finally, all models are simulated, fitted to data and compared automatically, if data is available and a corresponding parameter estimation task is specified in the *Copasi* master model. At the end, the user is provided with a ranking of the model fits and statistical measures that help to discriminate between model alternatives (Figure 1).

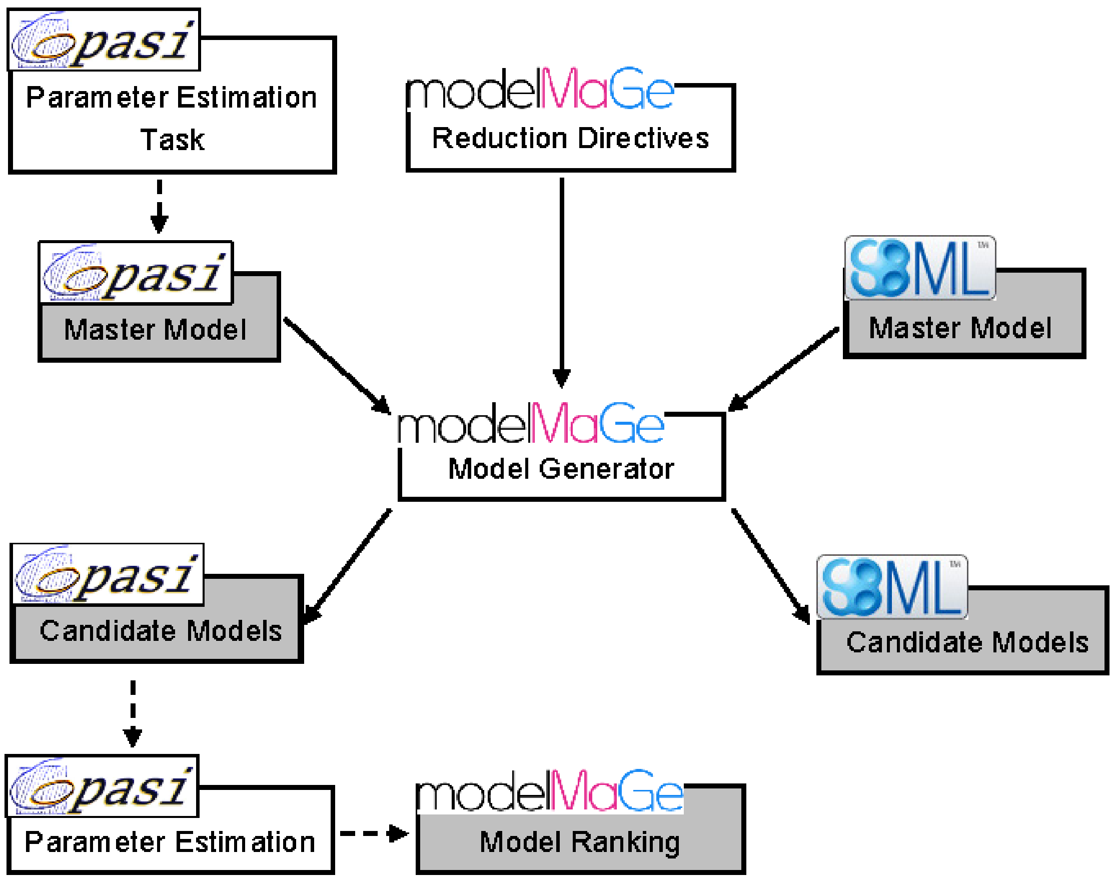


Figure 1: Workflow of *modelMaGe*. Light boxes represent program tasks that are executed by the programs indicated in the attached balloons. Grey boxes represent results files, with the respective formats indicated in the attached boxes. The user provides an SBML or *Copasi* file, which can also be provided with parameter estimation information, and directives for model generation. *modelMaGe* creates a set of models that can be passed to *Copasi* for parameter estimation. After that the files are returned to *modelMaGe* that generates a ranking of the candidate models.

## 2 Installation and running *modelMaGe*

### 2.1 Installation

*ModelMaGe* is a python package and a script that can be run from the command line. There are two other Python packages that are required to run *modelMaGe*. One is libSBML, that is used to handle SBML files, and the other one is networkx, which is a package for graph manipulation and visualization. To install modelMaGe and all its dependencies just follow the steps for your system. We recommend to use Python 2.5, libSBML 3.4.1 and networkx 0.36, because with those *modelMaGe* has been extensively tested. libSBML versions ≥ 4.0 are not suppoted at the moment, and libSBML 3.4.1 has no bindings for python 2.6 (for Windows at least).

### 2.1.1 Linux/Mac OS X

1. Install *Python 2.5* (www.python.org). The python version depends on the python bindings supported by *libSBML. Python 2.6* also works, but features of the networkx package 0.36 are deprecated.
2. download and install *libSBML 3.4*
    a. Go to http://sbml.org/Software/libSBML and download version 3.4 for your system. libSBML Versions ≥ 4.0 are currently not supported.
    b. Follow the instructions of the installer and make sure to install the python bindings.
3. download and install the python package *networkx*:
    a. download the networkx package (version 0.36) from http://networkx.lanl.gov/download/networkx/networkx-0.36.tar.gz
    b. Unpack the file and change to the directory networkx-0.36
    c. Run: python setup.py install
4. download and install Copasi 4.5:
    a. download installer from www.copasi.org
5. download and install *modelMaGe* from source:
    a. download the newest version as a tarball from www.modelmage.org
    b. Run tar xvfz modelmage-x.x.tar.gz
    c. cd modelmage-x.x
    d. python setup.py install

### 2.1.2 Windows

Download the Windows installer from www.modelmage.org and double-klick on it. The installer will check if *Python*, *libSBML* and *networkx* are installed and download and install them if needed. When all requirements are fullfilled *modelMaGe* will also be downloaded and installed.

If the installer for some reason does not work for you, you can download and install the individual programs and packages required for *modelMaGe* and then install *modelMaGe* from the *modelMaGe*-only Windows installer from www.modelmage.org.

For Windows we recommend to install the console program (>=v.1.5) from

Marko Bozikovic as the shell to run *modelMaGe* in. This console is much

more convenient than the native Windows console .

### 2.2 Running *modelMaGe*

To run *modelMaGe* add the folder containing Python scripts (e.g. C:\Programs\Python\Scripts\) to your PATH and PYTHONPATH environment variables. Then you can start the program in command-shell by just typing the command `modelmage.py`. That should work for all platforms.

## 3 Generating Model Alternatives

### 3.1 Preliminaries: defining and inspecting the master model

Generation of candidate models in *modelMaGe* is a two step process. The first step is to create a master model in *Copasi* (Hoops, et al., 2006) or in any other SBML (Finney and Hucka, 2003; Hucka, et al., 2003) compliant editor like *CellDesigner* (Funahashi, et al., 2007) or *SemanticSBML* (Schulz, et al., 2006). This model must include all possible species and reactions that shall be included in at least one of the candidate models. In the second step, the set of candidate models is built by removing reactions, species or modifiers from the master model (Figure 1). In addition, there is a possibility to automatically use simple alternative kinetics. This does not limited the use of more complicated kinetics, however, these have to be specified in dedicated reactions (see section 3.4).

In the following, we will consider a simple model `M1` that is schematically represented in Figure 1.

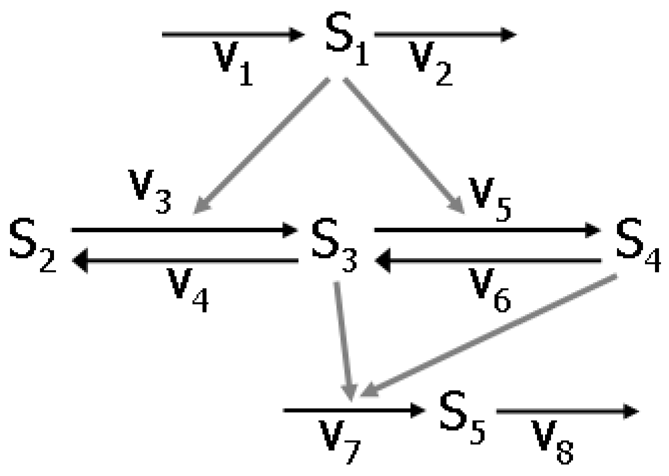


Figure 1: Scheme of the example model `M1`

The model `M1` consists of five species *S*$_i$, i=1,…,5, and eight reactions *v*$_j$, j=1,…,8. Reactions *v*$_1$ and *v*$_2$ produce and degrade species *S*$_1$, respectively. Reactions *v*$_3$ – *v*$_6$ (black arrows) convert species *S*$_2$, *S*$_3$ and *S*$_4$ into each other, and reactions *v*$_7$ and *v*$_8$ produce and degrade species *S*$_5$, respectively. Species *S*$_1$ influences reactions *v*$_3$ and *v*$_5$ without being consumed and is therefore called a modifier, indicated by a gray arrow on the black reaction arrows. Species *S*$_3$ and *S*$_4$ are modifiers to reaction *v*$_7$. For the moment, we assume simple mass action kinetics for all reactions.

We can implement the wiring diagram in any SBML (Finney and Hucka, 2003) compliant tool, e.g. *Copasi* (Hoops, et al., 2006) or *CellDesigner* (Funahashi, et al., 2007) or *SemanticSBML* (Schulz, et al., 2006).

**Note**: *modelMaGe* operates mainly on SBML files. In case a *Copasi* file is used as a master model, *modelMaGe* converts it to an SBML file version 2.3 on which it operates and then converts it back into a *Copasi* file.

A wiring scheme of `M1` in *CellDesigner* could look like the following (Figure 2):

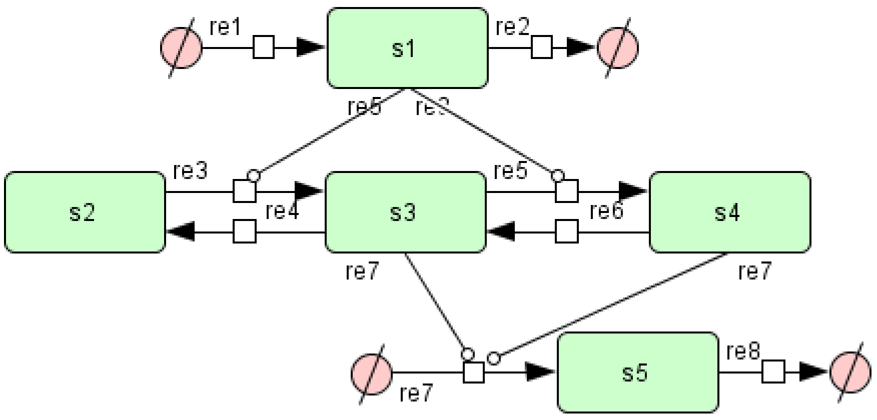
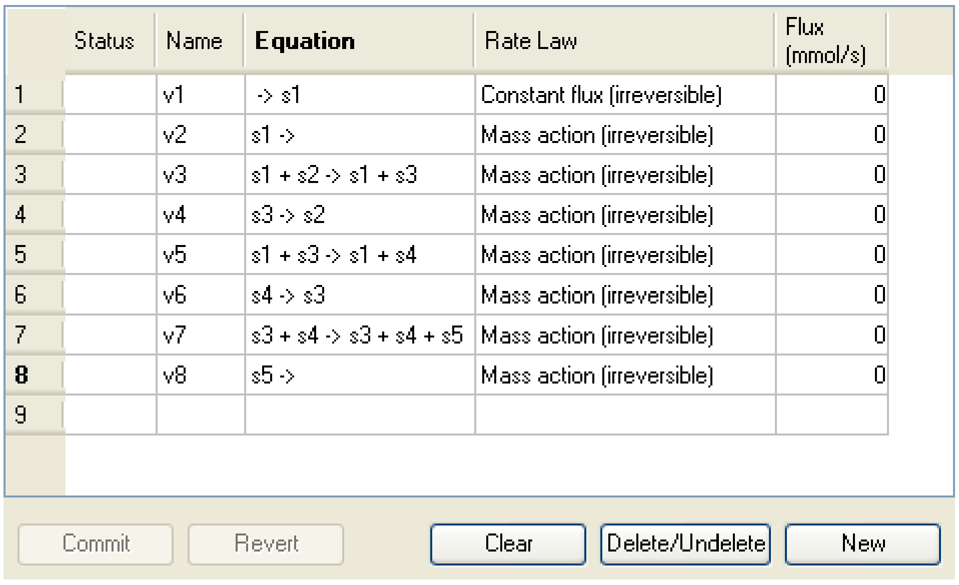


Figure 2: Wiring scheme of `M1` in *CellDesigner.*

**Note**: In CellDesigner producing reaction with no substrate (reaction `re1` and reaction `re7`)and degrading reactions with no product (reaction `re2` and reaction `re8`) introduce an additional species indicated by ∅.

**Note**: It is possible to provide *modelMaGe* with an SBML file where only the structure and no kinetics are specified. In that case, *modelMaGe* automatically inserts the appropriate mass-action kinetics with standard values for the parameters (usually 0.1).

In *Copasi*, one possible specification of model `M1` is displayed in Figure 3.

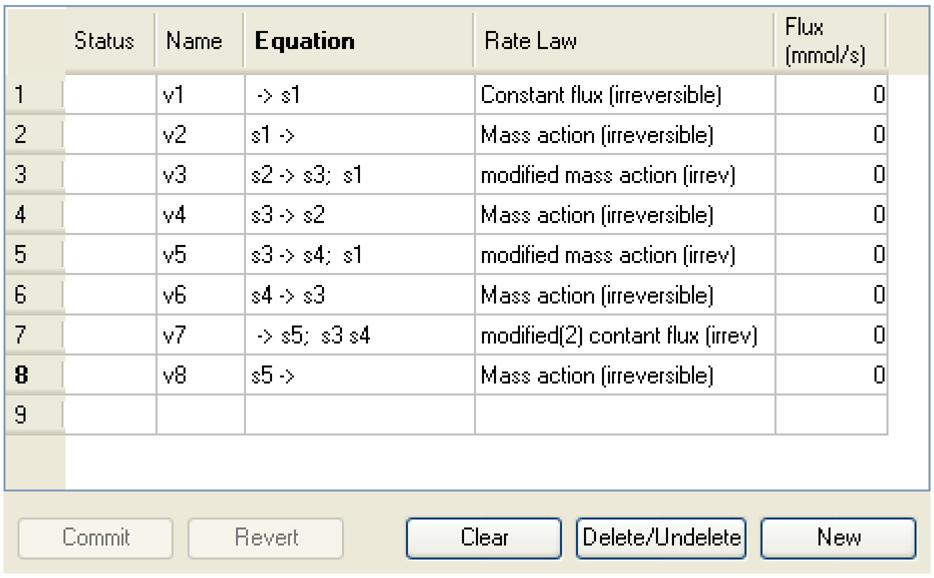


| | Status | Name | Equation | Rate Law | Flux (mmol/s) |
|---|---|---|---|---|---|
| 1 | | v1 | -> s1 | Constant flux (irreversible) | 0 |
| 2 | | v2 | s1 -> | Mass action (irreversible) | 0 |
| 3 | | v3 | s1 + s2 -> s1 + s3 | Mass action (irreversible) | 0 |
| 4 | | v4 | s3 -> s2 | Mass action (irreversible) | 0 |
| 5 | | v5 | s1 + s3 -> s1 + s4 | Mass action (irreversible) | 0 |
| 6 | | v6 | s4 -> s3 | Mass action (irreversible) | 0 |
| 7 | | v7 | s3 + s4 -> s3 + s4 + s5 | Mass action (irreversible) | 0 |
| **8** | | v8 | s5 -> | Mass action (irreversible) | 0 |
| 9 | | | | | |

Figure 3: *Copasi* screenshot of one possible specification of model `M1` using standard mass action kinetics.

In the specification of `M1` as displayed in Figure 3, the modifiers $S_1$, $S_3$ and $S_4$ are specified as substrates as well as products. Therefore, the standard mass action kinetics can be used here for the kinetic law.

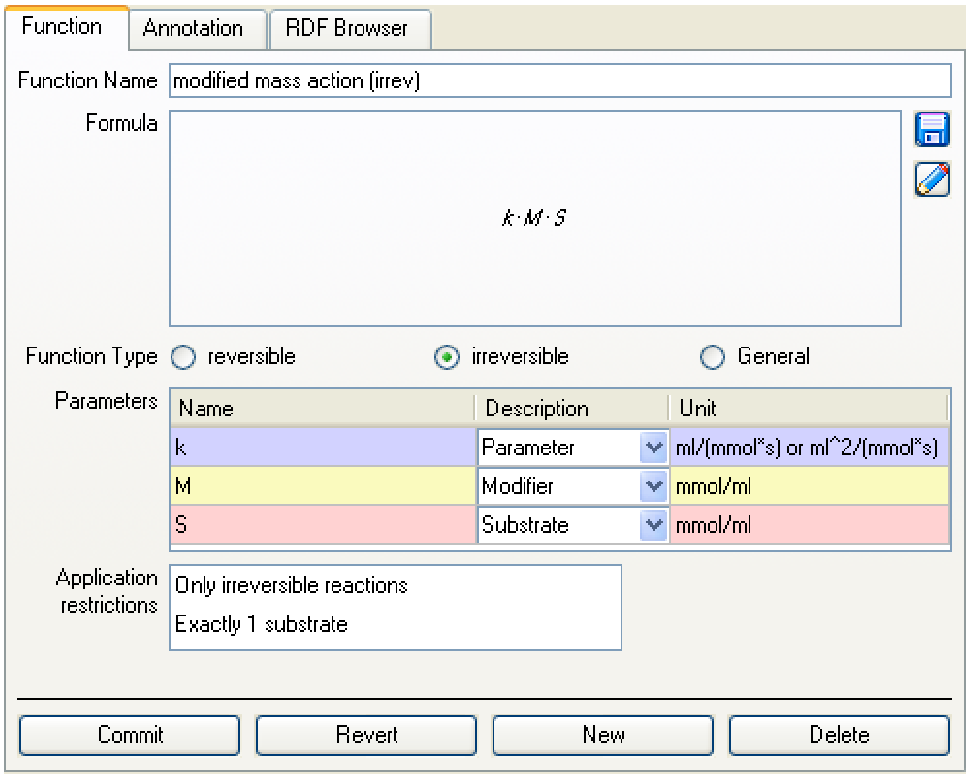
However, we could also choose a different description in *Copasi*, which is displayed in Figure 4:



|  | Status | Name | **Equation** | Rate Law | Flux (mmol/s) |
|---|---|---|---|---|---|
| 1 |  | v1 | -> s1 | Constant flux (irreversible) | 0 |
| 2 |  | v2 | s1 -> | Mass action (irreversible) | 0 |
| 3 |  | v3 | s2 -> s3; s1 | modified mass action (irrev) | 0 |
| 4 |  | v4 | s3 -> s2 | Mass action (irreversible) | 0 |
| 5 |  | v5 | s3 -> s4; s1 | modified mass action (irrev) | 0 |
| 6 |  | v6 | s4 -> s3 | Mass action (irreversible) | 0 |
| 7 |  | v7 | -> s5; s3 s4 | modified(2) contant flux (irrev) | 0 |
| **8** |  | v8 | s5 -> | Mass action (irreversible) | 0 |
| 9 |  |  |  |  |  |

Figure 4: *Copasi* screenshot of one possible specification of model `M1mod` using a user-defined kinetic (modified mass action).

In this specification, the modifiers $S_1$, $S_3$ and $S_4$ are defined neither as substrates nor a products, but as true modifiers in a user-defined kinetic, called 'modified mass action (irrev)' and 'modified(2) constant flux (irrev)'. A screenshot of the former is displayed in Figure 5:



Function Name: modified mass action (irrev)

Formula: $k \cdot M \cdot S$

Function Type: reversible [✓ irreversible] General

Parameters:

| Name | Description | Unit |
|---|---|---|
| k | Parameter | ml/(mmol*s) or ml^2/(mmol*s) |
| M | Modifier | mmol/ml |
| S | Substrate | mmol/ml |

Application restrictions:
Only irreversible reactions
Exactly 1 substrate

Figure 5: Screenshot of a user-defined kinetic in *Copasi*.

Mathematically, both formulations (Figure 3 and Figure 4) are equivalent.

In SBML models components (like species, reactions and modifiers) are addressed by names and identifiers. The latter are unique.

**Note**: *modelMaGe* addresses model components by their identifiers.

Once a model is implemented we can use *modelMaGe* to display a list of component names and identifiers with the –s option.

---------------------------------------------------------------------------------------------------

**Example 1:** The -s option shows us a listing of species, reactions and modifiers with their corresponding identifiers and names in the format [‘identifier’,’name’].

```
$ python modelmage.py –s M1.cps

modelMaGe v1.0beta


Model species:
['species_1','s1'],  ['species_2','s2'],  ['species_3','s3'],
['species_4','s4'],  ['species_5', 's5']

Model reactions:
['reaction_1', 'v1']
      products  : ['species_1']
['reaction_2', 'v2']
```

```
     substrates: ['species_1']
['reaction_3', 'v3']
     substrates: ['species_1', 'species_2']
     products  : ['species_1', 'species_3']
['reaction_4', 'v4']
     substrates: ['species_3']
     products  : ['species_2']
['reaction_5', 'v5']
     substrates: ['species_1', 'species_3']
     products  : ['species_1', 'species_4']
['reaction_6', 'v6']
     substrates: ['species_4']
     products  : ['species_3']
['reaction_7', 'v7']
     substrates: ['species_3', 'species_4']
     products  : ['species_3', 'species_4', 'species_5']
['reaction_8', 'v8']
     substrates: ['species_5']
```

----------------------------------------------------------------------------------------------------

**Note**: Modifiers that are declared as substrate and product are not especially indicated in the respective reactions.

----------------------------------------------------------------------------------------------------

**Example 2:** No option shows us a listing of options.

```
$ python modelmage.py
Usage: modelmage.py [options] [file]

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -c, --cleanresult     delete the result folder
  -d, --discriminate    parses result files of the parameter
                       estimation and displays a model ranking
                       according to the AIC.
  -i INIT, --init=INIT  path to an init file
  -k KINETICS, --kinetics=KINETICS
                        specifies reactions with alternative kinetics
  -o OUTPUT, --output=OUTPUT
                        name of the result directory, 'result' by
                       default.
  -p, --parameterestimation
                        uses CopasiSE to do a parameter estimation as
                        specified in  .cps
  -r REMOVE, --remove=REMOVE
                        SBML ids separated by logical operators
  -s, --show            print species and reaction in the .cps file
                       to the shell
  -v, --verbose         Verbose mode shows more information about
                       removed nodes.
```

----------------------------------------------------------------------------------------------------

## 3.2 Removing Reactions

If a reaction is chosen for removal, *modelMaGe* removes the specified reaction from the model and leaves the connected species untouched. Removing reactions is fairly simple, because they can simply be deleted from the model. The worst effect a deletion can have is that two or more unconnected graphs are present in the resulting model. If the removal of a reaction results in unconnected subgraphs, *modelMaGe* by default prints out a warning in the verbose mode, but nevertheless removes the reaction. If a reaction is removed, all the connected edges are automatically removed, too.

**Note**: The `-r` option allows us to remove reactions, given as a semicolon-separated list of logical expressions.

----------------------------------------------------------------------------------------------------------

**Example 3:** The -r option allows us to remove reactions, given as a string of logical expressions and/or semicolon -separated list.

```
$ python modelmage.py -r  "reaction_7 | reaction_1; reaction_8" 
M1.cps

modelMaGe 1.0beta

Model species:
['species_1','s1'],  ['species_2','s2'],  ['species_3','s3'],
['species_4','s4'],  ['species_5', 's5']

Model reactions:
['reaction_1', 'v1'](remove)
     products  : ['species_1']
['reaction_2', 'v2']
     substrates: ['species_1']
['reaction_3', 'v3']
     substrates: ['species_1', 'species_2']
     products  : ['species_1', 'species_3']
['reaction_4', 'v4']
     substrates: ['species_3']
     products  : ['species_2']
['reaction_5', 'v5']
     substrates: ['species_1', 'species_3']
     products  : ['species_1', 'species_4']
['reaction_6', 'v6']
     substrates: ['species_4']
     products  : ['species_3']
['reaction_7', 'v7'](remove)
     substrates: ['species_3', 'species_4']
     products  : ['species_3', 'species_4', 'species_5']
['reaction_8', 'v8'](remove)
          substrates: ['species_5']
```

```
            writing reaction_8.xml
            writing reaction_1.xml
            writing reaction_7reaction_1.xml
            writing reaction_7.xml

    4 SBML files created.

            writing reaction_8.cps
            writing reaction_1.cps
            writing reaction_7reaction_1.cps
            writing reaction_7.cps

    4 Copasi files created.
```
---------------------------------------------------------------------------------------------------------

The above example demonstrates a typical model generation task with *modelMaGe*. First, we see that two types of files are generated, SBML files and *Copasi*-files. Both files types are written to a specific directories, which are generated below the directory where the program `modelmage.py` lives and that are named

`result/ResultSBMLFiles` and

`result/CopasiFiles`,

respectively, by default. Morever, the master model is converted into a SBML file, in case it was given as a Copasi file, or standard mass action kinetics are introduced, in case a mere SBML structure was provided.

**Note**: The generated files are named after the components that have been removed from the master model.

Second, the above directive “`-r "reaction_7 | reaction_1; reaction_8"`” generates four different models. Directives that are separated by semicolons are treated as separate calls to *modelMaGe*. Therefore, the first model ‘`reaction_8`’ is generated by the directive “`-r "reaction_8"`”, telling *modelMaGe* just to remove reaction $v_8$. The last three models are generated by the directive “`-r "reaction_7 | reaction_1"`” , which tells *modelMaGe* to remove reaction $v_7$ or $v_1$ or both. The generated models are named accordingly.

**Note**: Logical operators that can be used by *modelMaGe* are | (OR), & (AND) and ^ (XOR). Moreover, logical statements can be grouped by parentheses.

**Note**: Logical directives very quickly become very confusing and hard to understand (combinatorial explosion), especially, when you try to implement a truth table with more than two inputs. We strongly recommend to only use the ‘&’-operator in connection with a semicolon-separated list.

It is possible to specify an empty list or an empty list element. In that case, the generated candidate is the master file itself. That becomes important when we want to introduce alternative kinetics in the master model (see section 3.4).

**Note**: The output directory can alternatively be specified with the `–o` option.

---

**Example 4a:** The `-r` option allows for an empty list or empty list elements.

```
$ python modelmage.py -r  "" M1.cps

…(snip)

        writing master.xml

SBML files created.

        writing master.cps

Copasi files created.
```

**Example 4b:**

```
$ python modelmage.py -r  ";reaction_7 | reaction_1;
reaction_8" M1.cps

…(snip)

        writing master.xml
        writing reaction_8.xml
        writing reaction_1.xml
        writing reaction_7reaction_1.xml
        writing reaction_7.xml

5 SBML files created.

        writing master.cps
        writing reaction_8.cps
        writing reaction_1.cps
        writing reaction_7reaction_1.cps
        writing reaction_7.cps

5 Copasi files created.
```

---

## 3.3 Removing Species

When the user specifies one or more species for removal, *modelMaGe* analyzes the neighbourhood of these species and rewires the model in an intelligent manner. The rewiring heuristic follows the principle of reachability and it works the following way:

First, *modelMaGe* detects the incoming and outgoing reactions of the species that shall be removed and analyzes the species involved in these reactions, i.e. the substrates of which the removed species was the product and the products of which the removed species was the substrate. Reversible reactions are decomposed into a forward and a backward reaction.

Then, *modelMaGe* tries to combine all pairs of incoming and outgoing reactions into one single new reaction, respectively. The new reaction inherits all substrates and products from the combined reactions and assigns a kinetic for the new reaction. The inserted kinetic is a suitable mass action kinetic by default, but other kinetics can be assigned, too (see section 3.4). There are three possible cases for each combination of incoming and outgoing reactions that have to be considered and treated separately.

1. If the combination of a pair of incoming and outgoing reactions would result in a reaction which has the same species, both as substrates and as products, i.e. a self loop to a list of species, the respective reactions are not combined.
2. If there is a subset of species equal in both substrates and products, then these species are regarded as enzymes and are set as modifiers for the resulting reaction (Figure 4 B).
3. If the sets are disjunct, the reactions are combined in the above described way.

Figure 6 depicts some possible scenarios how *modelMaGe* rewires the structures upon removal of a species. In the left panels the original models are displayed and in the right panels the resulting models upon removal of the species coloured in red.

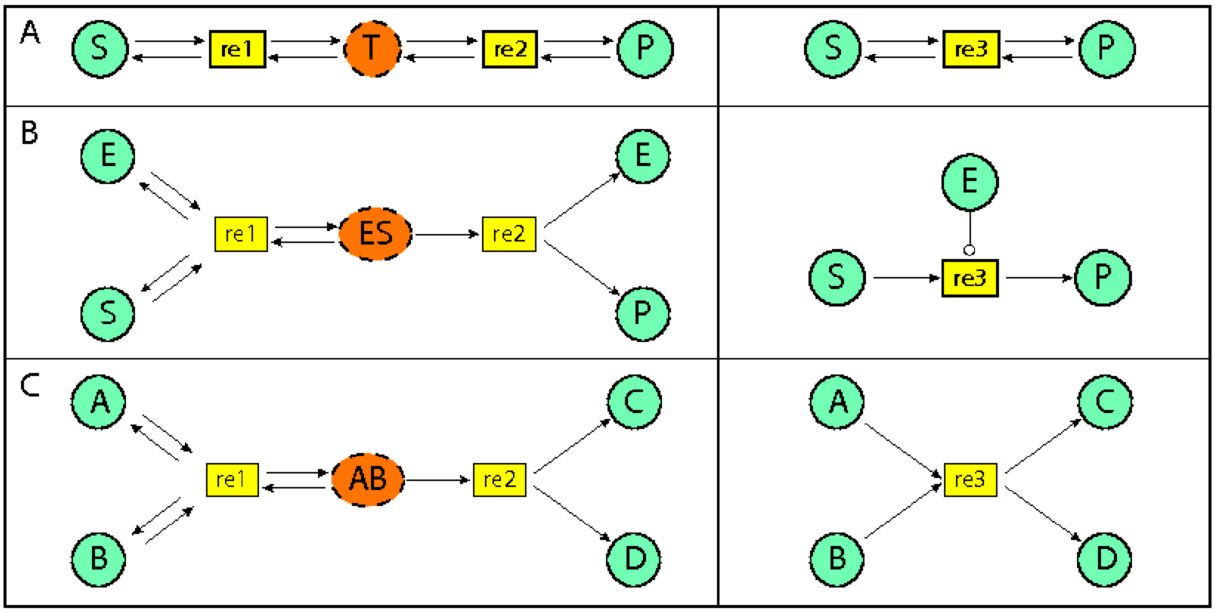


Figure 6: Examples of removed species. A: Removal of an intermediate species. Species T is removed and reactions `re1` and `re2` are combined. B: When an enzyme-substrate-complex is removed from a reaction *modelMaGe* creates a new reaction with the enzyme set as a modifier. C: Removal of species AB leads to new a reaction that is a combination of all incoming and outgoing reactions of the removed species.

The following example removes species $S_3$ from model `M1` and corresponds to the scenario in Figure 6A.

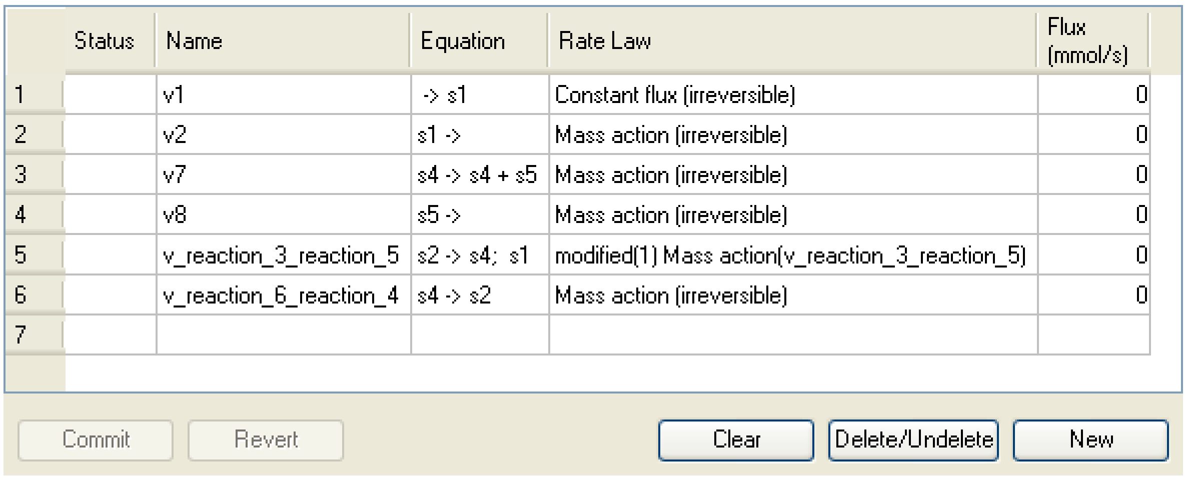
---

**Example 5:** Removing species $S_3$ from model `M1`.

```
$ python modelmage.py -r "species_3" M1.cps

modelMaGe 1.0beta

…(snip)

        writing species_3.xml

1 SBML files created.

        writing species_3.cps

1 Copasi files created.
```

---

The resulting model is displayed in Figure 7 in a **Copasi**-screenshot.



| | Status | Name | Equation | Rate Law | Flux (mmol/s) |
|---|---|---|---|---|---|
| 1 | | v1 | -> s1 | Constant flux (irreversible) | 0 |
| 2 | | v2 | s1 -> | Mass action (irreversible) | 0 |
| 3 | | v7 | s4 -> s4 + s5 | Mass action (irreversible) | 0 |
| 4 | | v8 | s5 -> | Mass action (irreversible) | 0 |
| 5 | | v_reaction_3_reaction_5 | s2 -> s4; s1 | modified(1) Mass action(v_reaction_3_reaction_5) | 0 |
| 6 | | v_reaction_6_reaction_4 | s4 -> s2 | Mass action (irreversible) | 0 |
| 7 | | | | | |

Figure 7: The resulting candidate model when $S_3$ from model `M1` is removed.

Apparently, in the candidate model, the original reactions $v_3$ – $v_6$ were removed along with $S_3$. Two new reactions were created, `v_reaction_3_reaction_5` and `v_reaction_6_reaction_4`, named after the original reaction pairs they combined, now connecting species $S_2$ and $S_4$ as in the scenario in Figure 6A. In this case, the new reactions are assigned a suitable mass action kinetic, because nothing else was specified.

**Note**: Any reaction that is changed or newly created gets a suitable mass action kinetic with standard parameters unless specified otherwise (see section 3.4). Moreover, modifiers that are kept in the new reaction, but were declared as substrate and product in the parent reaction, are re-declared as true modifiers as in Figure 5.

**Note**: When a species is removed that acts as a modifier its modifying influence is removed from the respective reaction(s), of course. In the verbose-mode a warning is displayed.

In case, the user wants a specific kinetic for a reaction that is changed in a candidate model, he can either tell *modelMaGe* to automatically insert a certain kinetic law (see Examples 6-9 in the next section), or define this reaction in the master model and just remove it from the candidate models, where it is not desired. The former mechanism is limited to a certain set of kinetics, whereas the latter mechanism applies in general.

**Note**: Model structures should be unique as they are the identifiers for the generated models. For generating a structure with kinetic alternatives, see below.

## 3.4 Inserting alternative kinetics

As mentioned above, one way to specify alternative kinetics for a reaction is to simply explicitly define these reactions in the master model and generate the candidate models by removing the unwanted reactions. This mechanism is independent from the chosen kinetics.

*modelMaGe* offers a much easier and elegant way to exchange kinetics for reactions. This mechanism, however, only applies to a limited set of kinetics. The alternative kinetics is specified by a string that can be described by the regular expression: `[m,i,ih]`*`[MA,MM,CF]`. This allows for three general types of kinetics, i.e `MA` for mass action, `MM` for Michaelis-Menten and `CF` for constant flux. These general kinetics can be modified by any number of modifiers, which can be of three types

a) `m` for multiplicative, i.e. the modifier is just multiplied by the respective kinetic law,

b) `i` for inhibitory, i.e. the kinetic law is multiplied by a the term $1/(1+M/K_i)$, where $M$ is the modifier,

c) `ih` for hill-inhibition, i.e. the kinetic law is multiplied by a the term $1/(1+(M/K_i)^h)$, where $M$ is the modifier, $K_i$ the half-inhibition constant, and $h$ is the Hill-parameter. $K_i$ is set to 1 by default and h is set to 2 by default.

**Note**: The `-k` option allows us to assign alternative kinetics to reactions. For each element of the semicolon-separated `–r` option list a corresponding list element in the semicolon-separated `–k` option has to be specified, which may be empty. Each element of the semicolon-separated kinetics list is a space-separated list the reaction-ids with the desired kinetic in parentheses: `reaction_id([m,i,ih]*[MA,MM,CF])`.

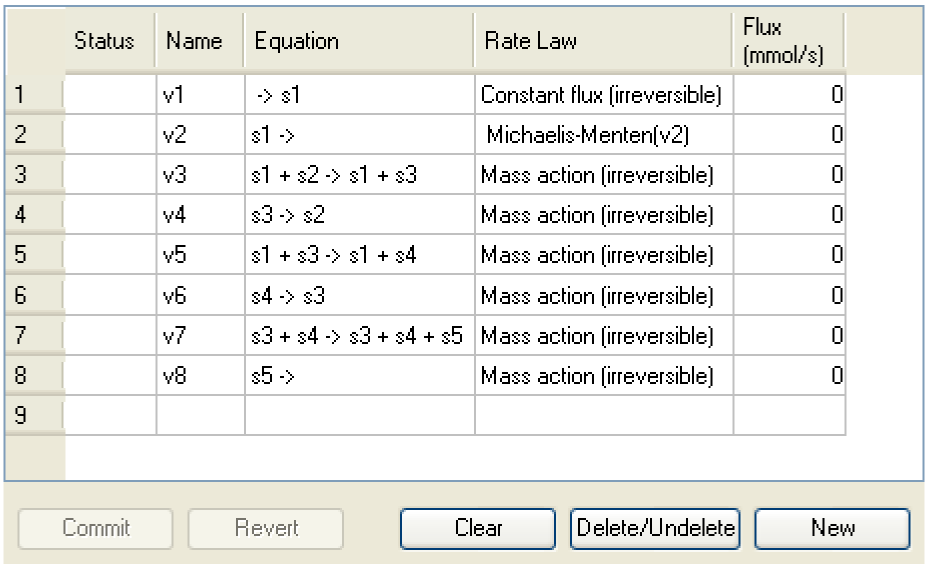
-------------------------------------------------------------------------------------------------------

**Example 6:** Assigning a Michaelis-Menten kinetic to $v_2$ of model `M1`.

```
$ python modelmage.py -k "reaction_2(MM)" M1.cps
…(snip)
     generating master.xml
1 SBML files generated.
      generating master.cps
```

```
        1 Copasi files generated.
```
---

This generates one model with a Michaelis-Menten kinetic for reaction $v_2$. When no remove-option is given, the master model is modified by default (Fig. 8)



| | Status | Name | Equation | Rate Law | Flux (mmol/s) |
|---|---|---|---|---|---|
| 1 | | v1 | -> s1 | Constant flux (irreversible) | 0 |
| 2 | | v2 | s1 -> | Michaelis-Menten(v2) | 0 |
| 3 | | v3 | s1 + s2 -> s1 + s3 | Mass action (irreversible) | 0 |
| 4 | | v4 | s3 -> s2 | Mass action (irreversible) | 0 |
| 5 | | v5 | s1 + s3 -> s1 + s4 | Mass action (irreversible) | 0 |
| 6 | | v6 | s4 -> s3 | Mass action (irreversible) | 0 |
| 7 | | v7 | s3 + s4 -> s3 + s4 + s5 | Mass action (irreversible) | 0 |
| 8 | | v8 | s5 -> | Mass action (irreversible) | 0 |
| 9 | | | | | |

Figure 8: The master model with Michaelis-Menten kinetic for reaction $v_2$.

**Note**: In the `-k` option string no logical expressions are allowed. The number kinetically different models per model structure is the number of unique combinations of kinetic alternatives.

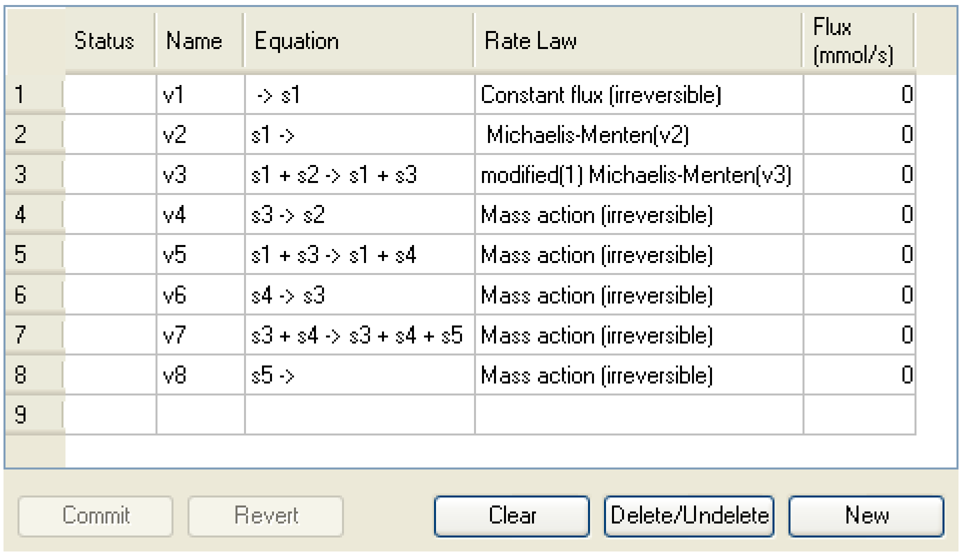
---

**Example 7:** More then one kinetic alternative per model structure.

```
        $ python modelmage.py  -k "reaction_3(mMM) reaction_3(mMA)
        reaction_2(MM) reaction_2(MA)" M1.cps

        …(snip)

                generating master.xml
                generating masterK1.xml
                generating masterK2.xml
                generating masterK3.xml

        4 SBML files generated.

                generating master.cps
                generating masterK1.cps
                generating masterK2.cps
                generating masterK3.cps

        4 Copasi files generated.
```
---

In Example 7 four models are generated, because there are two kinetic possibilities for two reactions, respectively, resulting in four possible combinations.

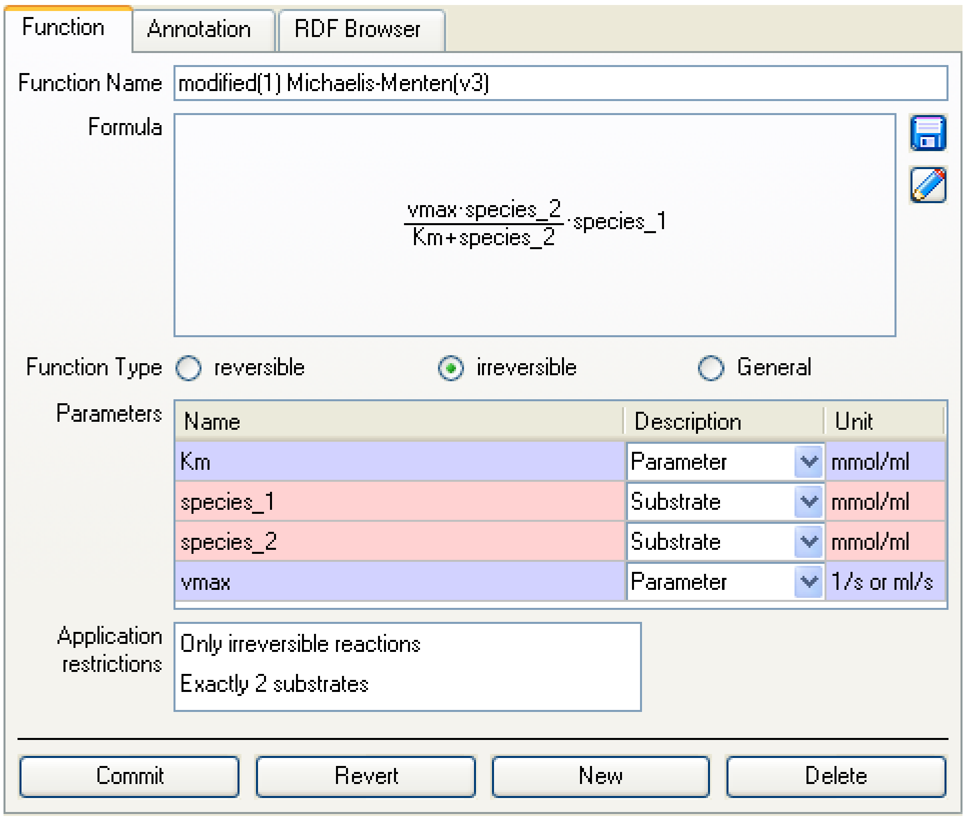
In case a reaction is assigned a new kinetic rate law and it is not newly created, the original structure of the reaction is not changed, however, in the reaction name possible modifiers are indicated. E.g. for `reaction_3` in Example 7, the modifier $S_1$, was declared as substrate and product, which is not changed, e.g., in the model `masterK3`, however, in the new reaction name the modifier is indicated (Figure 9).



| | Status | Name | Equation | Rate Law | Flux (mmol/s) |
|---|---|---|---|---|---|
| 1 | | v1 | -> s1 | Constant flux (irreversible) | 0 |
| 2 | | v2 | s1 -> | Michaelis-Menten(v2) | 0 |
| 3 | | v3 | s1 + s2 -> s1 + s3 | modified(1) Michaelis-Menten(v3) | 0 |
| 4 | | v4 | s3 -> s2 | Mass action (irreversible) | 0 |
| 5 | | v5 | s1 + s3 -> s1 + s4 | Mass action (irreversible) | 0 |
| 6 | | v6 | s4 -> s3 | Mass action (irreversible) | 0 |
| 7 | | v7 | s3 + s4 -> s3 + s4 + s5 | Mass action (irreversible) | 0 |
| 8 | | v8 | s5 -> | Mass action (irreversible) | 0 |
| 9 | | | | | |

Figure 9: The model `masterK3` from Example 7. The structure of `reaction_3` is kept as in the master model, with the modifier $S_1$ declared as substrate and product. However, in the new rate law the modifier is indicated.

The rate law ‘modified(1) Michaelis-Menten(v3)’ of the model `masterK3` is defined in *Copasi* as depicted in Figure 10:



Function Name: modified(1) Michaelis-Menten(v3)

Formula: $\frac{vmax \cdot species\_2}{Km+species\_2} \cdot species\_1$

Function Type: reversible [✓ irreversible] General

Parameters

| Name | Description | Unit |
|---|---|---|
| Km | Parameter | mmol/ml |
| species_1 | Substrate | mmol/ml |
| species_2 | Substrate | mmol/ml |
| vmax | Parameter | 1/s or ml/s |

Application restrictions:
Only irreversible reactions
Exactly 2 substrates

Figure 10: Rate law ‘modified(1) Michaelis-Menten(v3)’ with a multiplicative modifier that is declared as substrate and product of a reaction.

**Note**: In case a -r option is given with an -k option, the length of the semicolon-separated list has to be the same.

---

**Example 8:** Kinetic alternatives together with structural alternatives.

```
$ python modelmage.py  -r "species_5; reaction_1" -k ";
reaction_3(mMM) reaction_2(MM) reaction_2(MA)" M1.cps

…(snip)

        generating reaction_1.xml
        generating reaction_1K1.xml
        generating species_5.xml

3 SBML files generated.

        generating reaction_1.cps
        generating reaction_1K1.cps
        generating species_5.cps

3 Copasi files generated.
```

---

In Example 8 two structural alternatives are specified, one with `species_5` removed and the other with `reaction_1` removed. For the first structural alternative, without `species_5`, no kinetic alternative is given, indicated by the first empty list element of the `–k` option. For the second structural alternative, without `reaction_1`, two models are generated, because there are two possible combinations of alternative reaction kinetics.

In case more than one modifier is specified in an alternative kinetic, the order of the modifiers and their corresponding kinetic laws is determined by the order of species given in the species list that is displayed at the start of each program (see Example 3 and 9).

-----------------------------------------------------------------------------------------------------------

**Example 9a:** Order of modifiers in kinetic alternatives alternatives.

```
$ python modelmage.py  -k "reaction_7(mihCF)" M1.cps


Model species:
['species_1','s1'],  ['species_2','s2'],  ['species_3','s3'],
['species_4', 's4'], ['species_5', 's5']

Model reactions:
['reaction_1', 'v1']
      products  : ['species_1']
['reaction_2', 'v2']
      substrates: ['species_1']
['reaction_3', 'v3']
      substrates: ['species_1', 'species_2']
      products  : ['species_1', 'species_3']
['reaction_4', 'v4']
      substrates: ['species_3']
      products  : ['species_2']
['reaction_5', 'v5']
      substrates: ['species_1', 'species_3']
      products  : ['species_1', 'species_4']
['reaction_6', 'v6']
      substrates: ['species_4']
      products  : ['species_3']
['reaction_7', 'v7']
      substrates: ['species_3', 'species_4']
      products  : ['species_3', 'species_4', 'species_5']
['reaction_8', 'v8']
            substrates: ['species_5']

        generating master.xml

1 SBML files generated.

        generating master.cps

1 Copasi files generated.
```

**Example 9b:**

```
$ python modelmage.py  -k "reaction_7(ihmCF)" M1.cps
```

-----------------------------------------------------------------------------------------------------------

In Example 9a the kinetic law of `reaction_7` is `k*species_3*1/(1+(species_4/Ki2)^h2)`, whereas in Example 9b it is `k*1/(1+(species_3/Ki1)^h1)*species_4`, according to the order of modifiers.

As indicated above, within the described framework of alternative kinetics, there is also the possibility to assign an alternative kinetic to newly created reactions. This is simply achieved by assigning an alternative kinetic to a predecessor of the new reaction.

**Note**: An alternative kinetic for newly created reaction can be specified by assigning an alternative kinetic to one of the parent reactions.

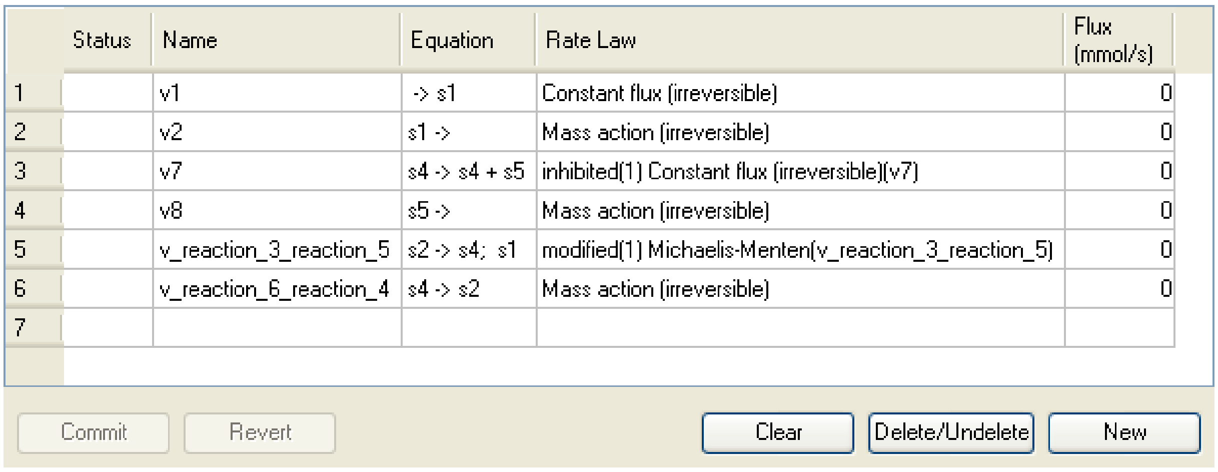
Taking the structure that is created by Example 5 we can assign a kinetic to the newly created reaction `v_reaction_3_reaction_5` by either specifying an alternative kinetic for `reaction_3` or `reaction_5` or both. In the latter case two models would be generated.

-----------------------------------------------------------------------------------------------------------

**Example 10:** Kinetic alternatives for newly created reactions.

```
$ python modelmage.py  -r "species_3" -k "reaction_3(mMM)
reaction_7(iCF)" M1.cps

…(snip)

        generating species_3.xml

1 SBML files generated.

        generating species_3.cps

1 Copasi files generated.
```
-----------------------------------------------------------------------------------------------------------

The *Copasi* reaction scheme of `species_3.cps` generated by Example 10 is displayed in Figure 11.



| | Status | Name | Equation | Rate Law | Flux (mmol/s) |
|---|---|---|---|---|---|
| 1 | | v1 | -> s1 | Constant flux (irreversible) | 0 |
| 2 | | v2 | s1 -> | Mass action (irreversible) | 0 |
| 3 | | v7 | s4 -> s4 + s5 | inhibited(1) Constant flux (irreversible)(v7) | 0 |
| 4 | | v8 | s5 -> | Mass action (irreversible) | 0 |
| 5 | | v_reaction_3_reaction_5 | s2 -> s4; s1 | modified(1) Michaelis-Menten(v_reaction_3_reaction_5) | 0 |
| 6 | | v_reaction_6_reaction_4 | s4 -> s2 | Mass action (irreversible) | 0 |
| 7 | | | | | |

Figure 11: The *Copasi* reaction scheme of `species_3.cps` generated by Example 10.

The alternative kinetic (iCF) that was specified for `reaction_7` includes only one modifier, whereas the original reaction has two modifiers that, in addition, where declared as substrate and product. No error is generated, because the alternative kinetic is only applied to the new model structure, in which `reaction_7` has indeed only one modifier, which, however, is still declared as substrate and product, because the inhibitors where declared this way in the original reaction.

**Note**: Alternative kinetics for new reactions must match their structure rather than the structure of the parent reaction to which it is actually assigned.

## 3.5 Removing modifiers

There is also the possibility to just remove the modifying influence of a species on a reaction. This does not change the number of species and reactions, but changes the wiring scheme and the kinetic of affected reaction.

**Note**: In case a modifier is removed from a reaction, this reaction is assigned the appropriate mass action kinetic.

**Note**: Modifiers of a certain reaction in the remove-option are indicated by ‘`reaction:modifier`’.

-----------------------------------------------------------------------------------------------------------

**Example 11:** Removing the modifying influence of $S_3$ on $v_7$ in model `M1`.

```
$ python modelmage.py  -r "reaction_7:species_3" M1.cps
```

-----------------------------------------------------------------------------------------------------------

Of course, the new reaction kinetic must not necessarily be mass action by default, but we can assign a new kinetic, too.

---

**Example 12:** Removing the modifying influence of $S_3$ on $v_7$ in model `M1` and assigning an inhibited constant flux kinetic to the new reaction $v_7$.

```
$ python modelmage.py  -r " reaction_7:species_3" -k
"reaction_7(iCF)" M1.cps
```

---

## 3.6 Using model alias and init-file

When a model structure with many removed components is generated, the resulting file name can become rather long. This can be avoided by assigning an alias of the form ‘`alias=`’ to a model structure. For the models generated in Example 8 this looks the following:

---

**Example 13:** Aliases for generated structures.

```
$ python modelmage.py  -r "c1=species_5; c2=reaction_1" -k
"c1=; c2=reaction_3(mMM) reaction_2(MM) reaction_2(MA)" M1.cps

…(snip)

        generating c2.xml
        generating c2K1.xml
        generating c1.xml

3 SBML files generated.

        generating c2.cps
        generating c2K1.cps
        generating c1.cps

3 Copasi files generated.
```

---

**Note**: The aliases of the –r option must match the ones of the –k option.

For convenience the remove-option list and the kinetics-option list can be given as a file. In the file the remove-option list and the kinetics-option list are defined by appropriate opening and closing tags. Comments can be added to the file by a leading '#' in the comment line. An example of an init file is given in listing 2.

---

**Example 14:** File definition of remove-option list and the kinetics-option of Example 13.

```
#Model option file M1.ini

[Begin Remove]

c1=species_5;
c2=reaction_1

[End Remove]

[Begin Kinetics]

c1=;
c2=reaction_3(mMM) reaction_2(MM) reaction_2(MA)

[End Kinetics]
```

---

**Note**: The list element must be separated by a semi-colon, as in the command-line. For clarity, however, in the options file the single list elements can additionally be on separate lines.

There are additional parameters that can be set in the options file. These are, at the moment, the upper boundary, lower boundary and initial value of newly created reaction parameters that are defined in the parameter estimation task. These setting are also set within enclosing tags.

---

**Example 15:** Setting upper boundary, lower boundary and initial value of newly created reaction parameters.

```
#Model M1.cps

[Begin FitItems]

LowerBound = 1e-6
UpperBound = 10
StartValue = 1

[End FitItems]
```

---

The options file is passed to modelMaGe by the –i option.

----------------------------------------------------------------------------------------------------------

**Example 16:** Example 13 by passing the options file from Example 14.

```
$ python modelmage.py  -i M1.ini M1.cps

…(snip)

        generating c2.xml
        generating c2K1.xml
        generating c1.xml

3 SBML files generated.

        generating c2.cps
        generating c2K1.cps
        generating c1.cps

3 Copasi files generated.
```

----------------------------------------------------------------------------------------------------------

## 4 Automatic parameter estimation and model discrimination

In case a parameter estimation task is defined in the master model and the `–p` flag is given on the command line, *modelMaGe* uses the *Copasi* simulation engine *CopasiSE* for parameter estimation. The engine is started for every candidate file that is created by *modelMaGe* and executes the parameter estimation task that is specified in the *Copasi* file for the master model. *modelMaGe* automatically updates the task according to the changes in the model. Parameters that were estimated in the master model but no longer exist in that particular candidate model are deleted from the list of parameters for estimation. If the combination of reactions or exchange of kinetics produced new parameters in the candidate model these are added to the list, if the parameters of the reactions were estimated in the master model. Metabolite concentrations and particle numbers that are involved in the parameter estimation task, e.g. as constraints, are also removed from the parameter estimation task in case they are no longer present in the respective candidate model.

**Note**: At the moment only local reaction parameters are deleted, inserted or updated. All modified parameters are assigned an initial value of 0.1 and a lower and upper boundary of $10^{-6}$ and 100, respectively, unless otherwise defined in the options files (Example 15).

The parameter estimation in *Copasi* creates output files for all of the models that contain details about the results of the estimation. These files are named like the corresponding candidate model with an “`_est.txt`”-extension. *modelMaGe* parses all files with this extension in the specified output folder and extracts the objective value, which is the weighted residual sum of squares (*SSR*) of the fitted model, the number of parameters *k* and the number of data points *n*. From this values it calculates the Aikaike’s Information Criterion AIC to rank and, thus, discriminate the models according to their *SSR* and number of parameters (Burnham and Anderson, 2002):

$$AIC = 2k + n\left(\ln\left(\frac{2\pi SSR}{n}\right) + 1\right).$$

If n>k-1, the AIC is corrected for small sample size:

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$

**Note**: The parameter estimation task for the candidate models is evoked with the –p option.

**Note**: *modelMaGe* parses all files with the “`_est.txt`”-extension in the given output-directory, not only the newly created.

**Note**: The model ranking without running the parameter estimation again can be displayed with the `-d` option. The ranking list is also written to a file `ModelRanking.txt.`

The parameter estimation task in the candidate models is set such that the estimation process

a) Uses the algorithm that is specified in the master model, and in case it is one of the Evolutionary Strategies, *`'Evolutionary Programming'`*, *`'Evolutionary Strategy (SRES)'`*, *`'Genetic Algorithm'`*, *`'Genetic Algorithm SR'`*,

b) population size is set to 10*k

c) number of generation is set to 10*n

d) seed of random number generator is 1, i.e. the estimation process always starts from the same random number and therefore should results always in the same result, at least on the same machine.

---

**Example 17:** Automatic model generation, parameter estimation and candidate discrimination with *modelMaGe*.

An artificial data set is displayed in Figure 11.

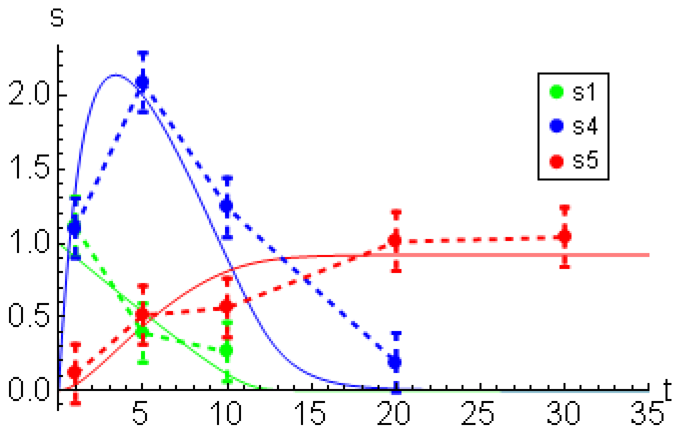


Figure 11: An artificial data set. The lines indicate simulations from the parameterized candidate model c4K3 (`reaction_2(MM) reaction_7(mCF)`(see below). The dots are generated by adding samples from a normal distribution with $\sigma^2$=0.2 to the simulations

The option file: Four candidate model structures are created each having four kinetic alternatives. For candidates c3 and c4, the kinetic alternatives for reaction $v_7$ has only one modifier, because from the corresponding structure one modifier is removed. In the master model the parameter estimation task is defined for all reaction parameters.

```
#Model M1.cps

[Begin FitItems]

LowerBound = 1e-6
UpperBound = 10
StartValue = 1

[End FitItems]
[Begin Remove]

c1=;
c2=reaction_1;
c3=reaction_1 & species_3;
```

```
c4=reaction_1 & species_3 & reaction_8

[End Remove]

[Begin Kinetics]

c1=reaction_2(MA) reaction_2(MM) reaction_7(mmCF)
reaction_7(miCF);
c2=reaction_2(MA) reaction_2(MM) reaction_7(mmCF)
reaction_7(miCF);
c3=reaction_2(MA) reaction_2(MM) reaction_7(mCF)
reaction_7(iCF);
c4=reaction_2(MA) reaction_2(MM) reaction_7(mCF)
reaction_7(iCF)

[End Kinetics]
```

Model generation, parameter estimation and candidate discrimination:

```
$ python modelmage.py -p -i M1fit.ini M1.cps

modelMaGe 1.0beta

Model species:
['species_1','s1'],  ['species_2','s2'],  ['species_3','s3'](remove),
['species_4', 's4'],
['species_5', 's5']

Model reactions:
['reaction_1', 'v1'](remove)
      products  : ['species_1']
['reaction_2', 'v2']
      substrates: ['species_1']
['reaction_3', 'v3']
      substrates: ['species_1', 'species_2']
      products  : ['species_1', 'species_3']
['reaction_4', 'v4']
      substrates: ['species_3']
      products  : ['species_2']
['reaction_5', 'v5']
      substrates: ['species_1', 'species_3']
      products  : ['species_1', 'species_4']
['reaction_6', 'v6']
      substrates: ['species_4']
      products  : ['species_3']
['reaction_7', 'v7']
      substrates: ['species_3', 'species_4']
      products  : ['species_3', 'species_4', 'species_5']
['reaction_8', 'v8'](remove)
            substrates: ['species_5']

      generating c1.xml
      generating c1K1.xml
      generating c1K2.xml
      generating c1K3.xml
      generating c3.xml
      generating c3K1.xml
      generating c3K2.xml
```

```
     generating c3K3.xml
     generating c2.xml
     generating c2K1.xml
     generating c2K2.xml
     generating c2K3.xml
     generating c4.xml
     generating c4K1.xml
     generating c4K2.xml
     generating c4K3.xml

16 SBML files generated.

     generating c1.cps
     generating c1K1.cps
     generating c1K2.cps
     generating c1K3.cps
     generating c3.cps
     generating c3K1.cps
     generating c3K2.cps
     generating c3K3.cps
     generating c2.cps
     generating c2K1.cps
     generating c2K2.cps
     generating c2K3.cps
     generating c4.cps
     generating c4K1.cps
     generating c4K2.cps
     generating c4K3.cps

16 Copasi files generated.

Update tasks for new created cpsfiles ...

Estimating parameters for result\ResultCopasiFiles\c4K2.cps ...
Done.
     Results written to result\ResultCopasiFiles\c4K2_est.txt


Estimating parameters for result\ResultCopasiFiles\c2K2.cps ...
Done.
     Results written to result\ResultCopasiFiles\c2K2_est.txt


…(snip)


Ranking of models by AIC(c):
  #            Model     n     k     Objective Value   AIC(c)
  1.     c4K2_est.txt   12     4              0.1605   -5.7191
  2.     c4K3_est.txt   12     5              0.1294   -0.3037
  3.     c3K3_est.txt   12     6              0.1294    7.6963
  4.       c4_est.txt   12     4              0.4936    7.7630
  5.       c3_est.txt   12     5              0.3299   10.9278
  6.     c4K1_est.txt   12     5              0.4770   15.3535
  7.     c3K1_est.txt   12     6              0.2781   16.8795
  8.       c2_est.txt   12     7              0.0896   17.2805
  9.     c2K2_est.txt   12     7              0.1545   23.8279
 10.     c3K2_est.txt   12     5              1.3050   27.4304
```

```
11.     c2K1_est.txt   12    8              0.1116    41.9209
12.     c2K3_est.txt   12    8              0.1770    47.4538
13.       c1_est.txt   12    8              2.2582    78.0102
14.     c1K2_est.txt   12    8              2.2624    78.0330
15.     c1K1_est.txt   12    9              0.1108    85.8343
16.     c1K3_est.txt   12    9              0.7789   109.2370
```

Among all sixteen generated candidate models, structure no. 4 (with $S_3$ and $v_1$ and $v_8$ removed (Figure 1)) and with all reactions having mass action kinetics was ranked best. In this case, the model structure from which the data were generated was recovered. However, the data was not sufficient or too noisy to recover the Michaelis-Menten kinetics of reaction $v_2$ that was used to produce the data. The simulation of the fitted model are displayed in Figure 12.

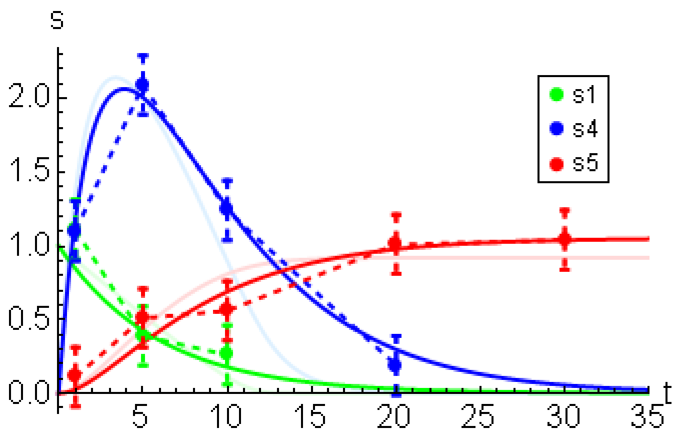


Figure 12: Model simulation of candidate c4K2 (thick lines). The light lines are the simulation from the original model c4K3 (Figure 10).

---

**Note**: The kinetic alternative per model structure is, at the moment, just indicated by numbers (K1, K2, etc). In order to know which specific kinetic alternative was implemented in a certain candidate structure, the user has to look into the *XML*- or *Copasi*-file.

## 5 Documentation of candidate models

The candidate models are all automatically annotated by *modelMaGe* in the annotation tag of the SBML model. This annotation includes the list of removed components from the master model and can always show how this special model is derived from the master model, even if the filename was changed.

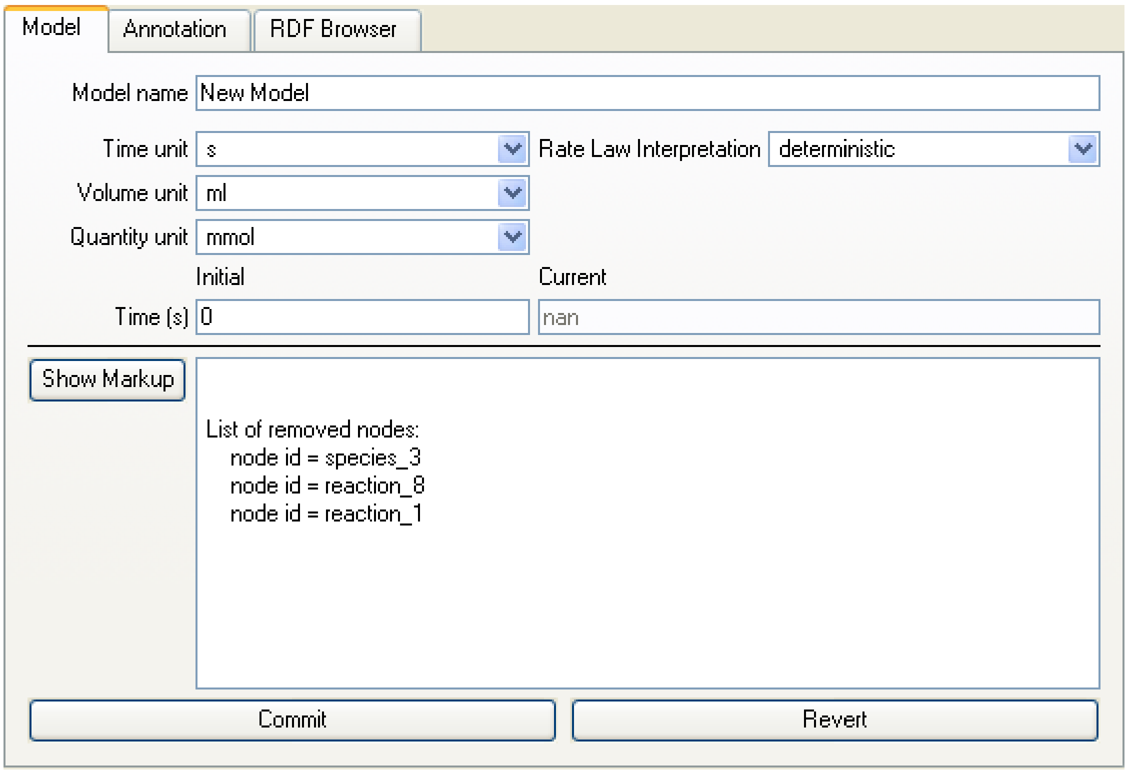
---

**Example 18:** Annnotation tag of `c4K3.xml` from example 14.

```
<annotation>
  <modelMage xmlns="http://www.modelmage.org/">
    <listOfRemovedNodes>
      <node id="species_3"/>
```

```
        <node id="reaction_8"/>
        <node id="reaction_1"/>
      </listOfRemovedNodes>
      <kineticNumber id="3"/>
    </modelMage>
  </annotation>
```
-----------------------------------------------------------------------------------------------------------

The same information is also display in the Copasi file (Figure 13).



Model name: New Model
Time unit: s
Rate Law Interpretation: deterministic
Volume unit: ml
Quantity unit: mmol

| | Initial | Current |
|---|---|---|
| Time (s) | 0 | nan |

```
List of removed nodes:
   node id = species_3
   node id = reaction_8
   node id = reaction_1
```

Figure 13: Screenshot from model `c4K2.cps` displaying the list of removed nodes.

## 6 Web Interface

To simplify the usage of the software and to give users the chance to test it before an installation on their hard disk, we developed a web-interface. The web-interface can be accessed freely at the *modelMaGe* website (www.modelmage.org).

The web interface is a very much simplified version of the complete program. Users can upload their model to the server and get an overview of the species and reactions that can be removed. They have the possibility to insert commands for the removal of the components and can generate the models they like. The generated models can be downloaded from the server afterwards.

In case a Copasi model with a specified parameter estimation task is uploaded, the cooresponding data file can also be uploaded and the candidate model can be fitted and ranked as described above.

## 6.1 Documentation of Code

The documentation of the written Python code is done by the *epydoc* package. *Epydoc* reads the doc strings of all modules, classes and methods and creates an HTML documentation from this. The documentation is downloadable via the *modelMaGe* website http://modelmage.org and is included in the source code package. It can also be viewed directly online.

# 7 References

Burnham, K.P. and Anderson, D.R. (2002) *Model Selection and Multi-Model Inference: A Practical Information-theoretic Approach*. Springer.

Finney, A. and Hucka, M. (2003) Systems biology markup language: Level 2 and beyond, *Biochem Soc Trans*, **31**, 1472-1473.

Flöttmann, M., Schaber, J., Hoops, S., Klipp, E. and Mendes, P. (2008) ModelMage: A Tool for Automatic Model Generation, Selection and Management, *Genome Informatics* **20**.

Funahashi, A., Jouraku, A., Matsuoka, Y. and Kitano, H. (2007) Integration of CellDesigner and SABIO-RK, *In Silico Biol*, **7**, S81-90.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P. and Kummer, U. (2006) COPASI--a COmplex PAthway SImulator, *Bioinformatics*, **22**, 3067-3074.

Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., Cuellar, A.A., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, II, Hedley, W.J., Hodgman, T.C., Hofmeyr, J.H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, A., Kummer, U., Le Novere, N., Loew, L.M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, T.S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J. and Wang, J. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, *Bioinformatics*, **19**, 524-531.

Schulz, M., Uhlendorf, J., Klipp, E. and Liebermeister, W. (2006) SBMLmerge, a system for combining biochemical network models, *Genome Inform*, **17**, 62-71.