ModelExplorer 2.1 User Manual

The package

ModelExplorer is shipped as an archive containing:

1) Linux or Windows binary file - “ModelExplorer”
2) A folder containing the source code of ModelExplorer - “source_code”
3) A folder containing the source code of ErrorTracer - “source_code_ErrorTracer”
4) Folders with libraries (COIN-OR Clp and Graphviz) - “modelEpxlorerLibs” and “extraLibs” (for Linux) or just a set of libraries (for Windows).
5) An installation script “install.sh” (on Linux only).
6) Font file - “arial.ttf”, necessary for running ModelEplorer
7) A test model (iTO977) - “test.xml”
8) This manual
9) A license notice - “LICENSE.txt”

Intended use

ModelExplorer is a metabolic model visualization package that can assist the user in finding blocked parts of the metabolic network as well as finding out why they are blocked. For this, the user is provided with three different definitions of inactive reactions: *FBA* and *Bidirectional* for correcting finished models, and *Dynamic* for building new ones. ModelExplorer assists the user in finding the source of network blocking with two tracking tools – one which shows the node’s nearest neighbours, and one that shows a minimal reaction path necessary to produce a metabolite. The user can also find and separately visualize connected sets of blocked reactions and metabolites – Blocked Modules. Blocked reactions and non-produced metabolites can often be the result of faulty transport between compartments. The network layout algorithm used in ModelExplorer therefore visually separates and highlights different cellular compartments, in order to make troubleshooting easier and more intuitive. Using the inbuilt ErrorTracer algorithm (also provided as a command line tool) the user can explicitly trace the origins of all reaction inconsistencies. Finally ModelExplorer has all the necessary tools required to edit existing models and build up new ones by hand.

Installation

The program is provided as two different executables, one being the graphical software ModelExplorer, and the other being a command line tool running the ErrorTracer algorithm. The programs come in two versions: Windows and Linux. The Windows versions have been tested to run on Windows 10, while the Linux verision has been tested to run on Ubuntu 17.04, 17.10, 16.04 LTS and 18.04 LTS, as well as on Manjaro 17.1.1. We recommend using a true installation of the system and not a virtual machine, as the latter will likely not be able to take advantage of the GPU.

On Windows no installation is required. The only premise is that the ModelExplorer (or ErrorTracer) executable is kept in the same folder as the libraries and the font file. The user is advised to place the content of the archive into a single folder and make a shortcut to the ModelExplorer executable. Alternatively the program can be run from a terminal (cmd) window, which is advisable in case troubleshooting will be required.

The Linux installation procedure works as follows:
Extract the archive to a folder of your choice. Run the installation script, agreeing to everything that the user will be prompted with (it will preform a system update and install the required libraries, copying the libraries shipped with ModelExplorer into a system folder).

```
$ sudo ./install.sh
```

After the installation script has been run, ModelExplorer can be launched from within a terminal or directly. Running it from terminal is advisable in case troubleshooting will be required.

Compiling

The program is shipped together with its source code located in the “source_code” folder, which can be compiled on Linux using Cmake. *In order to compile the program, the user first needs to perform the installation steps in the previous section.* Next, the user needs to enter the build folder inside the source folder:

```
$ cd <your-ModelExplorer-folder>/source_code/build
```

The code can then be compiled using the following two commands:

```
$ cmake ..
$ make
```

The executable “ModelExplorer” will then be located in the “build” folder and will require the “arial.ttf” font file to be located in the same folder with the executable.

The same compilation procedure as above applies to the ErorTracer executable, except its source code is located in the “source_code_ErrorTracer” folder. The font file is not required to run ErrorTracer standalone executable.

Launching

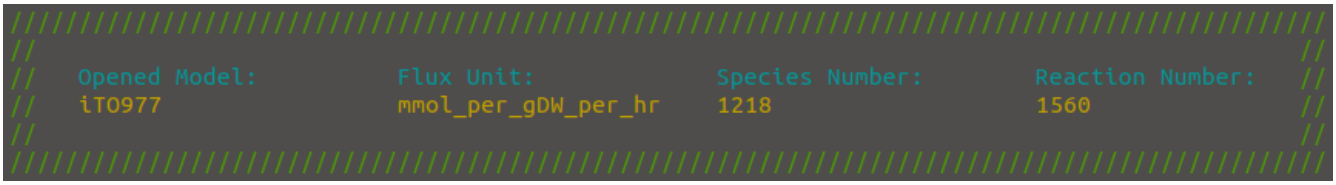
The program should be run from the terminal and the command could be followed by a path to a model file that is to be opened. For example on Linux:

```
$ ./ModelExplorer test.xml
```

Or on Windows:

```
$ ModelExplorer.exe test.xml
```

The program accepts only **SBML2** as the model format. A model could be loaded either with the command above or from within the GUI. If reactions refer to undefined metabolites, undefined compartments or lack stoichiometries, the user will be presented with self-explanatory warnings. If species, reactions or compartments lack the “id” field, an appropriate error message will be presented, and the model will not be opened. When the model gets loaded the user is presented (in the terminal) with model name, flux unit and the number of chemical species and reactions in the model:



When a model is loaded, the program will immediately proceed to calculating which metabolites and reactions are blocked in the three available blocking modes. This and the following graphical layout procedure may take some time, especially for models with more than 3000 reactions / metabolites.

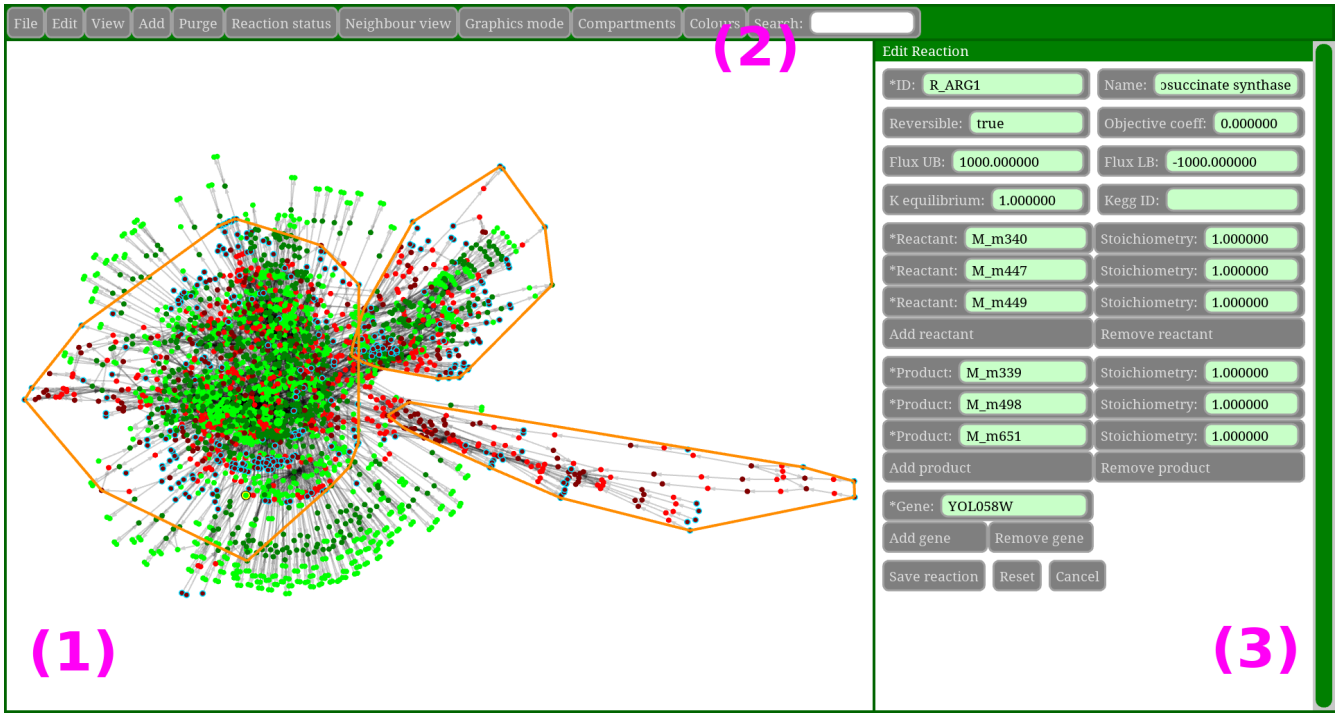
For certain models with a lot of metabolites that are not utilized in any reaction (e.g. Recon2), it is recommended to use the *"-ps"* flag before the path to the model in order to remove these species and decrease the time it takes to make a layout of the model. In case of very large models (with connected parts $>10^4$ reactions) it may take a long time to run all the three algorithms, so the user can use *"-FBAonly"* flag to save time and only do the FBA (and error) calculations, omitting the Bidirectional and Dynamic ones.

If the user wants to test the execution time of the different algorithms on his or her computer, the flag *"-time"* should be added. In order to test the graphical performance of ModelExplorer the flag *"-FPS"* will output the frames per second of the visualization. Note that the latter command will only show output if the user is actively navigating, as otherwise the screen is not refreshed.

All of the above commands also apply to the ErrorTracer executable, with the addition of the *"-o"* flag, which should be followed by the folder of file name to which the error tracking output of the ErrorTracer should be saved. If this flag is not specified, the output will be saved to the same folder as the executable itself, and it will get the name "ErroTracer_results_test.xml".

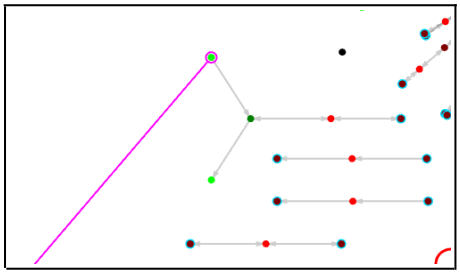
Graphical User Interface:

When the model is loaded the user is presented with the GUI of ModelExplorer. On top is the ***command panel*** with all available commands (1), to the left is the ***network view*** in which the model is visualized (2), and to the right is a ***text panel*** (3), which is used to output information from the neighbour viewing tools and input/output information from model editing tools.



Command Panel:

The panel contains a set of function menus and a ***Search*** tool. The search tool can be used to search reactions and species by their id or name, by typing a part of that id or name (not shorter than 2 letters) and selecting one of the suggestions that appear. The search tool is ***case sensitive***. When an entry is selected **purple** circle is drawn around the target, and a line of similar colour is drawn from the lower left corner to the circle:



The set of menus is as follows:

1) **File** – make ***new***, ***open*** and ***save*** models, ***quit*** the program. ***Save as SVG*** saves the current network view to a file in SVG format (vector graphics). ***Save blocked modules*** saves the blocked reactions and species found in the three modes (*FBA, Bidirectional and Dynamic*) into CSV files, grouped into columns by their blocked module, sorted by module size (descending). ***Save error tracing*** saves the results of the execution of the ErrorTracer algorithm to a specified file using the XML format. A description of the general structure of this file is presented in the section "Error tracing file structure".
2) **Edit** – ***undo*** and ***redo*** the last change to the model (adding, removing or changing species, reactions and compartments), ***replot, recalc***ulate blocked reactions/species by all three available methods (recalc all), or only using the FBA mode (recalc FBA).
3) **View** – ***Blocked module*** and ***Whole model*** plots only the selected blocked module or the whole model respectively. The former mode is to reduce complexity when resolving inconsistencies.
4) **Add** – ***species***, ***reaction***, ***compartment*** to the model (for model editing and building). The user gets to fill out a set of input boxes in order to create a new species, reaction or node. The user can click the "Save species/reaction/compartment" of "Cancel" buttons in order to add a new entry or close the adding menu. Entries marked with an asterisk (*) are obligatory to fill out. The rest are not, but rules can apply to the content. The sets of entries are:
   Add ***species:***
   - ID – required, must be unique
   - Name – not required, can by anything
   - Compartment – not required, must select one of existing compartments or none
   - Boundary condition – required, must be either true or false
   - Formula – not required, can be anything
   - Kegg ID – not required, can by anything

   Add ***reaction***:
   - ID – required, must be unique
   - Name – not required, can by anything
   - Reversible – required, can be either true or false
   - Objective coeff (coefficient) – not required, must be a number
   - Flux UB (upper bound) – not required, must be a number
   - Flux LB (lower bound) – not required, must be a number
   - K equilibrium (equilibrium constant) – not required, must be a number
   - Kegg ID – not required, can by anything
   - Add Reactant – button adds reactant to the end of the list
     - Reactant – required, must select one of existing species
     - Stoichiometry – required, must be a number (set to 1 by default)
   - Remove Reactant – button removes last reactant in list
   - Add Product – button adds product to the end of the list
     - Product – required, must select one of existing species
     - Stoichiometry – required, must be a number (set to 1 by default)
   - Remove Product – button removes last product in list
   - Add Gene – button adds gene to the list of genes
     - Gene – required, specifies an identifier of the generate
   - Remove Gene – button removes last gene in list

   Add ***compartment***:
   - ID – required, must be unique
   - Name – not required, can by anything
   - Outside – not required, (compartment outside the current compartment), must select one of existing compartments or none

An input box gets red if the input is rejected and green if it is accepted. If all entries are valid and the user clicks the “Save” button, a new species gets placed at a random position on the screen or a new reaction gets placed at the barycenter of its reactants and products or a compartment gets created (but does not appear until species are added to it).

5) **Purge** – remove the selected option:
   - ***Selection*** – removes a species, reaction or compartment selected with a RMB click.
   - ***Boundary species*** (names ending in “_b” or “_boundary”).
   - ***Extracellular species*** (names ending in “_e” or “_extracellular”). This and the previous options were added in order to correct some models in which cellular import was initiated in a boundary or extracellular species instead of a reaction which only has products.
   - ***Disconnected species*** (not participating in any reaction).
   - ***Disconnected reactions*** (lack both reactants and products).
   - ***Disconnected clusters*** (remove any clusters (as well as singlets) of reactions and species not connected to the largest cluster).
6) **Reaction status** – visualize blocked reactions and species according to:
   - ***Always on*** – no reactions or species are shown as blocked.
   - ***FBA*** – a reaction is shown blocked if it cannot carry a steady state flux. A species is shown as blocked if all reactions that can generate it are blocked.
   - ***Bi-directional*** – setting all reactions to be bidirectional, a reaction is shown as blocked if it cannot carry a steady state flux. A species is shown as blocked if all reactions that can generate it are blocked.
   - ***Dynamic*** – a species is shown as blocked if it will block the biomass (growth) reaction when added to it. Useful in the process of building models, as it shows if the cell can produce that species dynamically or not. A reaction is blocked if any of its reactants is blocked.
7) **Neighbour view** – highlight the species/reactions linked to the currently active one. A reaction/species is made active by holding the cursor over its node in the network:
   - ***None*** – show only the current species/reaction and an ***Edit menu*** for it (in the Text Panel).
   - ***Ego-centric*** – if the active node is a reaction, show its reactants and products. If the active node is a species, show producing and consuming reactions.
   - ***Node ancestry*** – track the shortest path (all the way to import reactions) necessary to produce a species or to make a reaction active. Visualized in **purple** colour. Possible only if the pathway is non-cyclic (is not a part of a strongly connected component). If cyclic, the cycle (strongly connected component) is visualized in **black**. Strongly connected components may constitute large parts of a model.
   - ***Blocked module*** – if a species or reaction is blocked, shows the connected blocked module (the largest possible set of connected blocked reactions and species). This module can be viewed for itself using ***View → Blocked module*** in order to make model correcting easier.
   - ***Error Tracer*** – traces the cause of reaction inconsistency. For every blocked reaction it shows a set of species around which the model is inconsistent, which can cause blockage of that specific reaction. Resolving ***all*** of the inconsistencies may not be necessary to unblock the specific reaction, and there may be many alternative combinations (subsets) of the shown inconsistencies that may do the work. When a blocked reaction is selected, hovering over or selecting an inconsistency-causing species will show the set of reactions that the species can affect. If the inconsistency is caused by a set of consistent reactions being stoichiometrically locked together in a cycle or set of cycles , this set is also highlighted, as well as the the species at the interface of this set and the group of blocked reactions.
8) **Graphics mode** – determine the resolution of the arrows linking the nodes (the most graphically intense part of drawing):

- ***High resolution*** – full resolution.
- ***Low resolution*** – half (in each direction) of the full resolution.

9) **Compartments** – ***Show*** or ***Hide*** the visualizations of cellular compartments.
10) **Colours** – set the colours of:
    - ***Active reactions'*** nodes
    - ***Active species'*** nodes
    - ***Blocked reactions'*** nodes
    - ***Blocked species'*** nodes
    - Contours around ***Endpoint species*** (species that lack either producing or consuming reactions).
    - ***Disjoint species'*** nodes (species that are not associated with any reactions).
    - Contours around cellular ***Compartments*** (as well as the text colour that tells which compartment the cursor is currently inside).
    - ***Constraining reactions*** and ***species*** (see the Error Tracer neighbour view).

Network view:

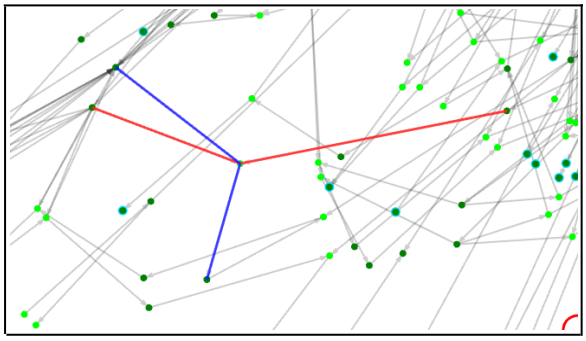
The network view shows a bipartite graph representation of the metabolic model. The nodes are connected by ***gray*** semitransparent arrows. If a reaction is unidirectional, the arrows have only one arrowhead (indicating the reaction direction). If bidirectional, the arrow has two arrowheads. Reactions and species are represented with different shades of the same colour (reactions are ***bright*** and species are ***dark***). If the reaction/species is active the base colour is ***green***, and ***red*** if blocked. Endpoint species (those which are either not produced or not consumed) have a ***light blue*** outline, while the biomass (growth) reaction has thick ***yellow*** outline.

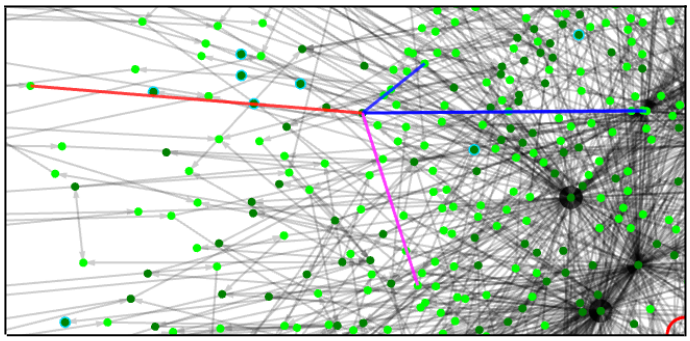
When the user hovers the cursor over a species or reaction, information about it is displayed in **purple** in the upper left corner of the view. This information contains the ***name*** of the reaction/species, its ***id*** in [square] brackets and if species, its parent ***compartment*** in (round) brackets. When Neighbour View is set to ***None***, an edit menu appears in the text panel to the right, showing model-related information about the species, reaction or compartment (the same as when you add a new one). If the user right clicks a species, reaction or compartment, the ***edit menu*** gets fixed on it, so the user can make changes to it. Colours can be changed using the panel *Colour* menu.

Cellular compartments are represented with ***orange*** outlines. Each outline contains all the species in that compartment, but may also contain species from other compartment (if compartments intersect). The drawing of compartment outlines can be turned off using the *Compartments* menu in the panel. Even when turned off, the name of the compartment in which the cursor currently is will be displayed in ***orange*** in the upper left corner of the view below the species/reaction information. This colour can also be changed.

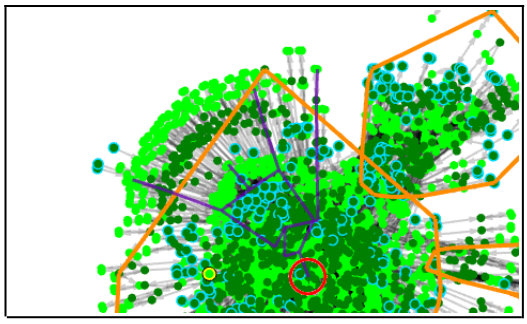
Network view – neighbour view:

The ***Neighbour view*** option in the panel allows the user to choose between *Ego-centric*, *Node ancestry* and *Blocked module* modes. In the ego-centric mode, when the user holds the cursor over a species or reaction the links to its direct neighbours are highlighted. For reactions these are ***red*** for reactant and ***blue*** for products (left picture). For species these are ***red*** for producing, ***blue*** for consuming and ***purple*** for bidirectional reactions (right picture).



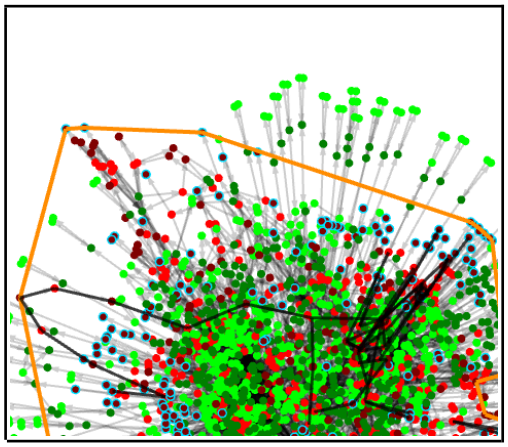


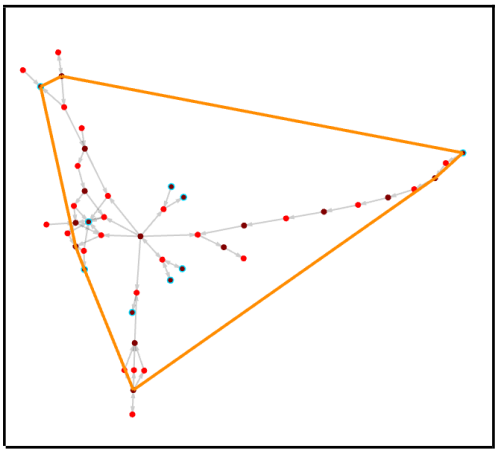
One can freeze the current selection by right-clicking the node (and continue viewing neighbours of other nodes while the ones of the frozen are still shown), and unfreeze by clicking elsewhere.



In the node ancestry mode, the one can view the shortest pathway (all the way to import reactions) necessary to produce a species or to make a reaction active (see picture below). The pathway is shown in **purple** colour. It is possible to draw this pathway only if the pathway is non-cyclic (is not a part of a strongly connected component). If cyclic, the cycle (strongly connected component) is visualized in **black**.

In the ***Blocked module*** mode, ModelExplorer highlights in **black** the set of connected blocked reactions and metabolites to which the target species or reaction belongs (i.e. the blocked module). The module can be viewed on its own using ***View → Blocked module***, in order to make the search for a possible cause of the inconsistency in this blocked module less complicated. Below to the left one can see a Blocked Module as selected in the full model, and to the right the blocked module on its own.





The ***Error Tracer*** mode traces the cause of reaction inconsistency. When hovering the mouse or selecting a blocked reaction it shows a set of species around which the model is inconsistent, which can cause blockage of that specific reaction. These inconsitency-causing species are visualized as ***orange***

crosses (see picture below). Resolving ***all*** of the inconsistencies may not be necessary to unblock the specific reaction, and there may be many alternative combinations (subsets) of the shown inconsistencies that may do the work. Each species-centered inconsistency has a certain set of blocked reactions which it can influence. This set of reactions and species is visualized as ***black lines***, which get thin at species that are consumed or produced by reactions outside the set. The set is shown when the user freezes a blocked reaction, and then hovers or freezes one of the inconsistency-causing species.

The inconsistencies can be of 4 types: ***source***, ***reversibility***, ***stoichiometry*** and ***cycle***.

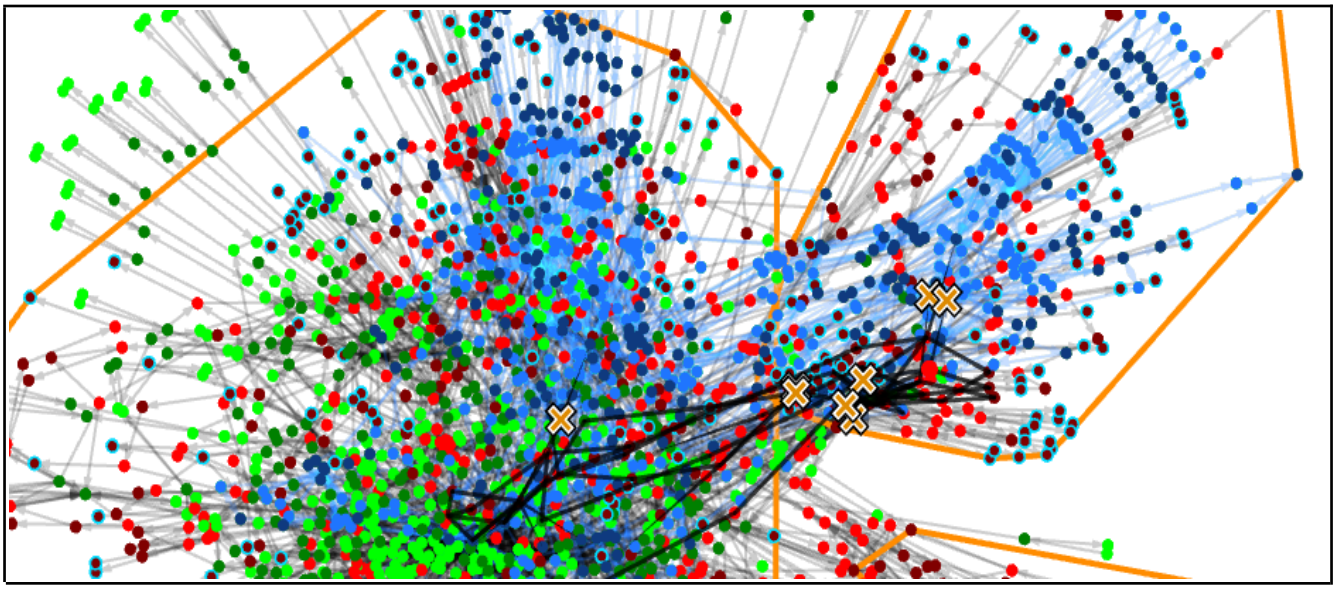
Source – inconsistencies are caused by a species being consumed or produced by only one reaction or by a group of reactions whose fluxes are directly coupled. These species are usually easy to spot visually as they look like dead ends and have a light blue rings around. Solving this kind of inconsistencies can be done dirtily by adding an import or export reaction or pathway to the species in question, or by adding a pathway that would connect it to a related but consistent species in the network. Adding the species to the biomass-production reaction may solve the problem if the species is produced but not consumed.

Reversibility – inconsistencies are caused by species being only consumed or only produced by all reactions around. This may not be easy to spot by just inspecting the network, as this kind of error may often be masked by false reversibility of the surrounding reactions (false in the sense that the reversible reaction is in reality forced into one specific direction by neighbouring irreversible reactions). Solving this kind of inconsistency can be similar to solving source inconsistencies, unless the error is caused by the user setting reaction reversibility wrong. Therefore it is advisable that the user carefully considers the nearby reactions to find out if any of them have wrong reversibility or direction.

Stoichiometry – inconsistencies are caused by cycles of directly coupled reactions, which try to net-produce or net-consume a species within the cycle. The user has to consider if this is caused by faulty reaction stoichiometry, which creates a false “perpetuum mobile”, or if the stoichiometry is valid, and thus the species in question needs a producing or consuming pathway, and this is thus another source-inconsistency.

Cycle – inconsistencies are similar to stoichiometry-inconsistencies in the sense that they are caused by cycles of reactions. However, in cycle-inconsitencies the reactions are not coupled, and the reactions in the cycle are not blocked. The cycle in question, however, does not support consumption (***sink***) or production (***source***) (or both) of species from the cycle. Thus such a cycle blocks anything that tries to add or consume its species without returning them back to the cycle (and thus becoming a part of the cycle itself). When this kind of cycle is the cause of the inconsistency, ModelExplorer shows the cycle reactions in ***blue*** and species in ***dark blue*** (see the picture on the next page). The orange crosses in this case are species which the blocked group of reactions tries to consume or produce that belong to the constraining (blue) cycle.

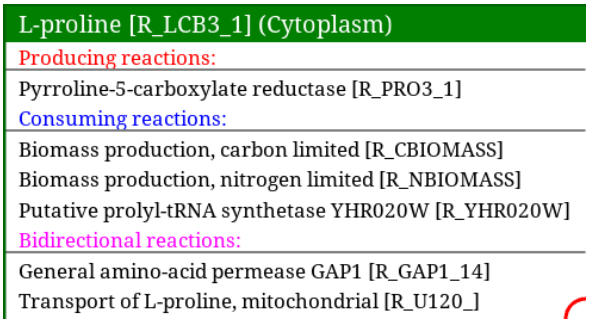
There are several ways of resolving cycle-inconsistencies. One is to extend the group of blocked reactions so that their net influx into the cycle is zero (i.e. they return what they consume). Another is to add an importing or exporting reaction pathway to the species that the blocked reaction group is trying to consume, or to any related species within the cycle. Adding imports/exports to non-related species within the cycle will not solve the inconsistency.

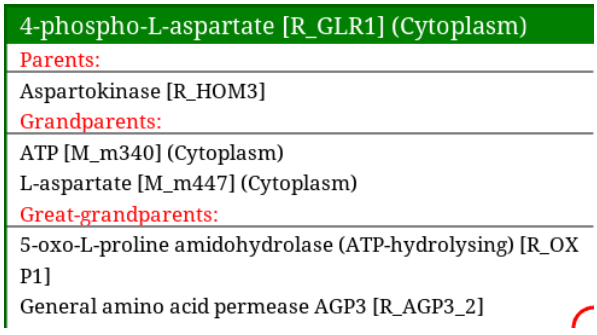


Text panel:

When Neighbour View is set to ***None*** and the user hovers the cursor over or clicks a species, reaction or compartment, the text panel shows an ***Edit menu*** for this object. When hovering, the edit menu can only be used for viewing information about the object. When an object is selected (with the RMB), the edit menu can be used to change its properties.

When using the Neighbour View menu as discussed above, the name, [id] and (compartment) is diplayed for all the plotted species and reaction in the text panel. In the ego-centric mode, one sees a list of producing, consuming and bidirectional reactions (left picture below). In the node ancestry mode one can see list of the node's ancestor nodes by "generation" - the shortest distance to the node (see right picture below).





In Blocked Module view, the text panel shows the members of the blocked module. In Error Tracer view the Text panel shows either the species whose inconsistency can block the current reaction (if one hovers over a blocked reaction), or, when hovering over an inconsistency-causing species, it shows reactions that can be blocked by its inconsistency as well as the members of the constraining cycle when the error type is "Cycle".

Error tracing file structure:

The error tracing results can be saved to a file from within ModelExplorer, using the "File → Save error tracing" command. The same file is produced when running the ErrorTracer from the command line. The command line tool is intended to be executed by external software when repeated error tracing or automated analysis of the results is required.

The tracing results are presented in a tree-like fashion using the XML format. The results have the following structure:

```
algorithm:
    @name: ErrorTracer
    @version: <algorithm_version>

    blocked_reactions:
        reaction:
            @id: <reaction_1>

        error_sources:
            species:
                @id: <species_1>
                @error_type: <source, reversibility, stoichiometry>
            species:
                @id: <species_2>
                @error_type: <cycle>
                @cycle_id: <cycle_1>
            …
        …

    blocking_cycles:
        cycle:
            @id: <cycle_1>

            species:
                @id: <species_2>
            …
```

In this structure, blocked reactions are listed together with their ids, which link back to ids in the original model. Every blocked reaction has a list of error sources (species), every such source carrying the id of the respective species and the type of error it induces (source, reversibility or stoichiometry). If the error type is "cycle", a cycle id is also listed. After the list of blocked reactions, we have a list of blocking cycles, every cycle having an id that links to the cycle ids of the error sources. Every blocking cycle has a corresponding list of all member species of the cycle.

License