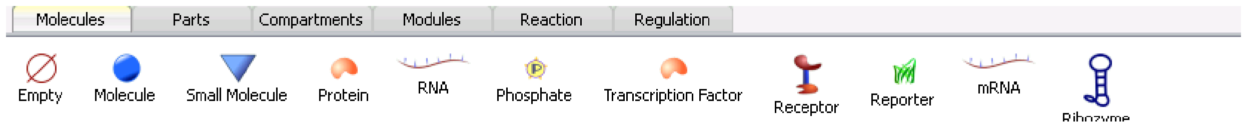
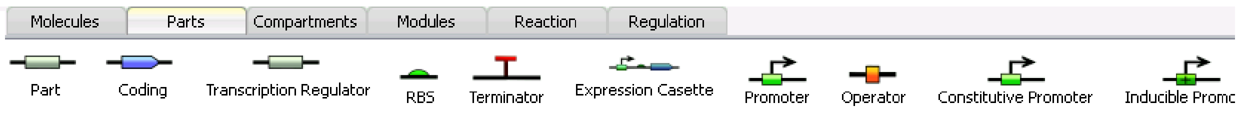
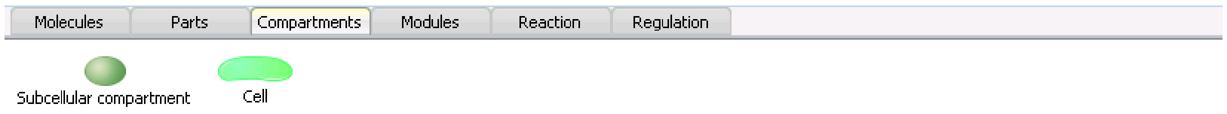
TinkerCell User Documentation

Biological components







The catalog of biological components (shown above) are the fundamental objects that are used to construct models in TinkerCell. Each component has a unique set of attributes and a designated visual representation. The attributes characterize that part. For example, the attributes for a "Coding" part are its DNA sequence (a text string) and the length of the sequence (a number). The default visual representation of each part can be changed by the user.

The catalog is organized as a tree, although not displayed that way. The tree of components is organized into
three major divisions: Nodes, Compartments, and Modules. The "empty"

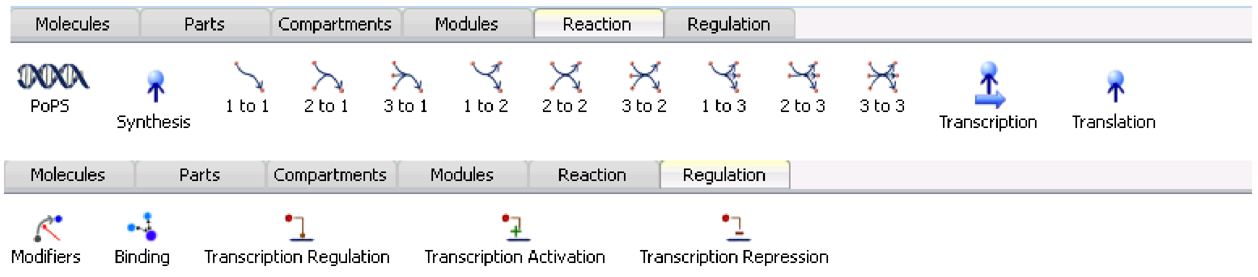
object is a special object that belongs in all categories. The "Nodes" division represents molecules such as proteins, RNA, and metabolites or fragments of DNA such as genes and promoters. The "Compartments" division represents things with volume that can contain other things. Compartments can include sub-cellular regions such as nucleus or mitochondrion. They may also represent general regions that the model wishes to separate. Cells are a specific kind of compartment. The third division, "Modules", represent sub-models that can be connected together to form bigger models. This is described more in a later section.

The philosophy of TinkerCell is to make models well characterized. Instead of defining all components of a model as generic "variables" and "reactions", TinkerCell models also contains information about the type of item, such as "Promoter", "Protein", "mRNA", and so on. The list of components and their hierarchy are stored in an XML file called NodesTree.xml. Editing this file will alter the tree of components. This file will eventually be loaded from a database of known biological parts. Users may click the right mouse button on components in the catalog to learn more about each item.

Units

Units are not defined in a model but in the catalog of parts and connections. TinkerCell models (particularly modules) are meant to be compatible with one another, and that is why units are kept away from the models themselves and instead shifted to the tree of parts and connections. It is assumed that the units provided in the XML file that defines the catalog of parts and connections will be "standard" units.

Connections

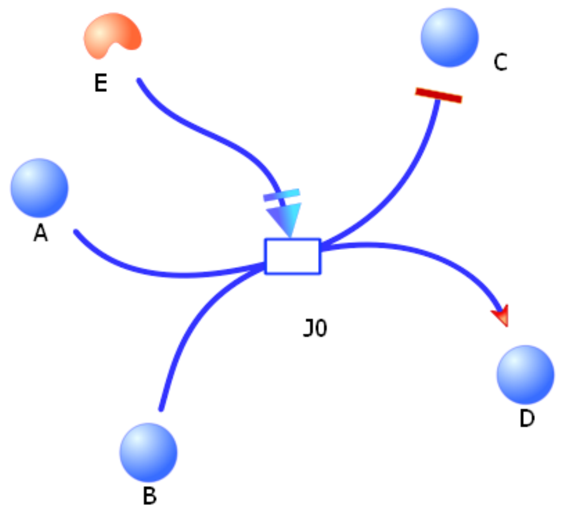


The catalog of connections (shown above) is the set of different types of connections that can be used to connect molecules and parts. Each type of connections can only be made between specific types of nodes. For example, a reaction where one small molecule such as glucose is converted to another small molecule is classified as a "biochemical" reaction in TinkerCell. Sub-classes of "biochemical" reactions include the "1 to 1", "2 to 1", "2 to 2", and similar connections listed in the catalog. When a transcription factor regulates a promoter, it is classified under "transcription regulation", which can then be further classified as activation or repression. A transcription regulation cannot be used to connect two molecules together (TinkerCell will display an error). Some of these connections only allow two parts to be connected, while some allow different number of parts to be connected. The type and number of parts allowed for each connection can be viewed by clicking the right mouse button on one of the connections in the connections catalog.

Once again, the philosophy in TinkerCell is to make models well characterized. If all the reactions were simply called "reactions", it would be very difficult for TinkerCell to interact with databases or perform other tasks which requires knowledge about the type of connections.

Visual details

Connections can be decorated in many ways: the arrow heads can be changed using the "replace graphics" option in the toolbar; the "middle region" can be made visible and replaced using the "replace graphics" option again. The lines can be dashed, dotted, or solid. Modifier connections (bottom-most option in the connections tree) are used to connect a part to a connection. Below is an illustration.



Inserting components

How to insert components:
Components are inserted by clicking on one of the buttons in the components catalog and then clicking on the screen. You can insert more of the same component by clicking again. Press Escape or the Spacebar to exit the insertion mode.

Extra details: what happens a component is inserted:
Various "plug-ins" in TinkerCell monitor the user's activity. When a component is inserted, the plug-ins carry out various operations. For example, one of the plug-ins called the "Numerical Attributes" retrieves information from the catalog of components and connections in order to define default *attributes* of each component. These See the parameters section for more detail.

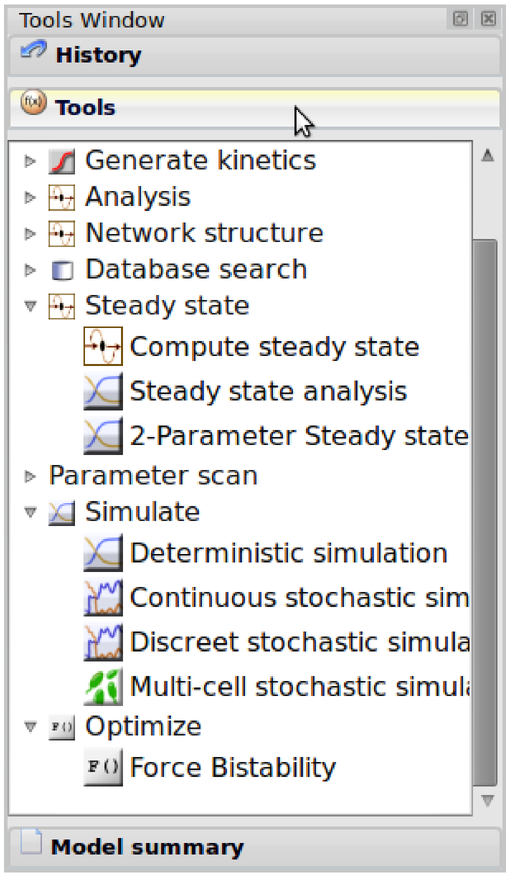
Inserting connections

How to insert connections:
Connections are inserted by first clicking on one of the connections in the connections tree and then selecting the parts to connect, one by one. The connections tree imposes some restrictions about what type of parts can be connected with the different types of connections, and those rules are enforced. For example, a user may not connect a promoter and a gene using a "biochemical" reaction, because the definition of a "biochemical" reaction in the connections tree states that it much between two or more "species" parts.

Extra details: what happens a connection is inserted
Various "plug-ins" in TinkerCell monitor the user's activity. When a connection is inserted, the plug-ins carry out various operations. One of the important plug-ins pertaining to connections is the "Stoichiometry and Rates" plug-in, which ensures that any new reaction has a default rate equation. Genetic networks are somewhat more complicated because TinkerCell offers different ways of modeling genetic networks. There is a special plug-in called "Auto Gene Regulatory Tool" that automatically generates rate equations for transcription and translation based on the nearby promoters and ribosomal binding sites. See the section on genetic networks for more details.

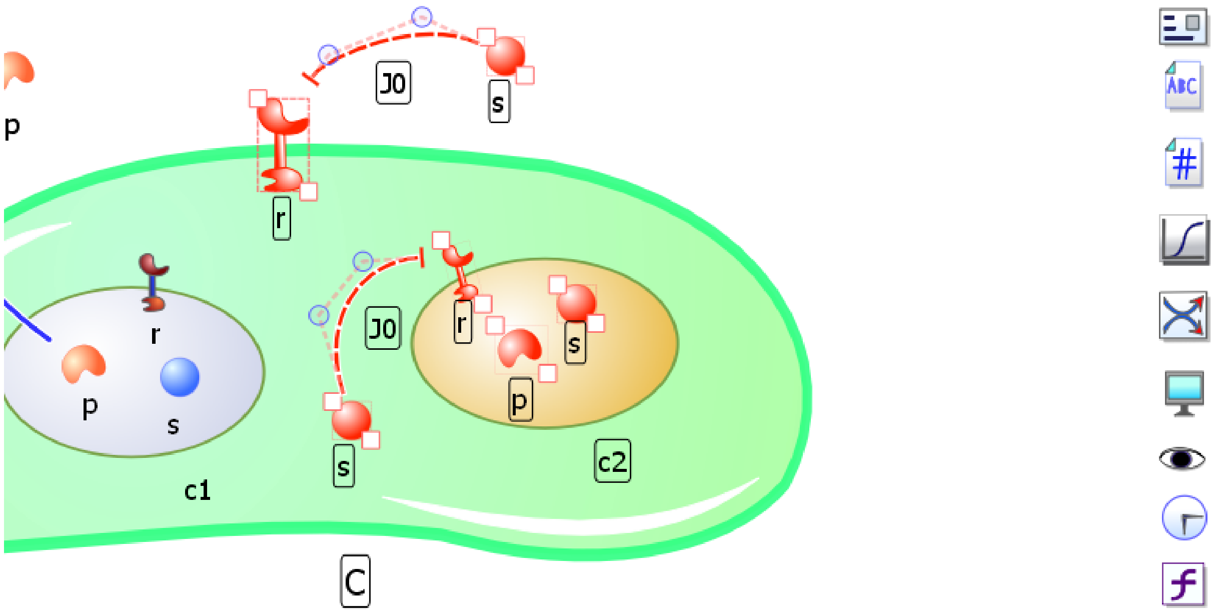
Simulating and analyzing a model

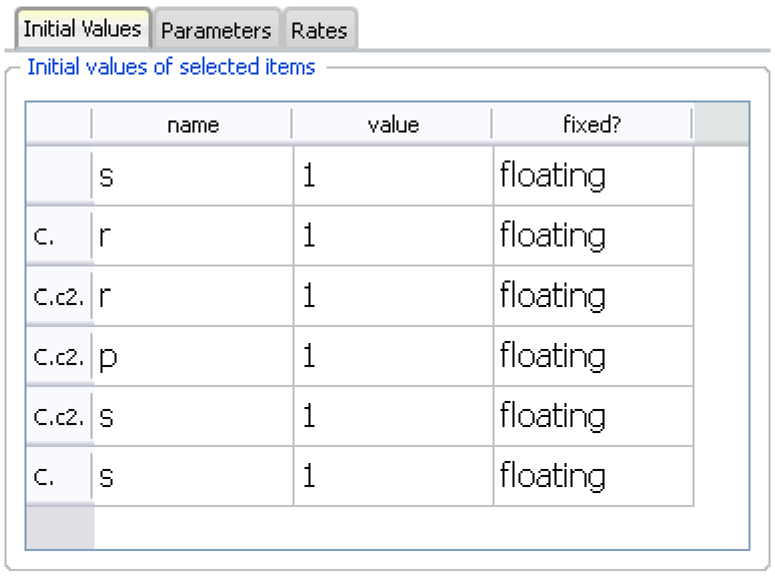
TinkerCell provides numerous simulation and analysis functions. All of these functions are listed in the "Tools" menu. Each function is listed inside a category. For example, the "Simulate" category contains deterministic simulation and stochastic simulation. The "Steady state" category contains 1D and 2D steady state analysis, which can be used to study the input-output response of a system.



View and edit model equations

There are several ways to examine and edit a model. The first method is to use the tools that are displayed when the user selects items on the screen (as shown in the diagram below). Pointing the on top of each tool will show a tool tip describing the tool.





	name	value	fixed?
	s	1	floating
C.	r	1	floating
C.c2.	r	1	floating
C.c2.	p	1	floating
C.c2.	s	1	floating
C.	s	1	floating

Each tool will allow the user to view and edit specific features of the model. For example, the "Reaction rates" tool allows users to view and edit the reaction rates and stoichiometry for selected reactions. The "Model summary" tool, shown to the right, uses several other tools to display a summary of the model.

Alternatively, the user may double-click or press enter on selected items to see the model summary window.

View and edit parameters

Parameters, also called "Numerical Attributes" in TinkerCell (only relevant for programmers), are constant values that may or may not be used in the model. The important fact is that parameters are local, meaning that they belong with an item. For example, reaction J0 might have parameter k0 and reaction J1 might also have a parameter called k0. The full name of each parameter is J0.k0 and J1.k0, respectively. When entering an equation, the full parameter name should be used. If the user enters "k0" by accident, TinkerCell will select one of the two: J0.k0 or J1.k0.

Naming convention

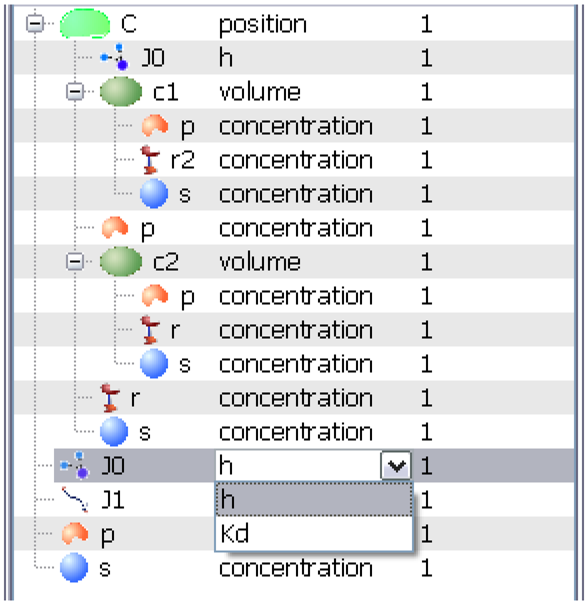
When an item has a parent item, such as a compartment, the parent's name is prepended to the name. For example, is part "A" is inside compartment or module "B", then the complete name of "A" is "B.A". If part A has a parameter named "k1", then the way to address this parameter in other parts of the model would be "B.A.k1".

Automatic correction

Every time the model is modified, TinkerCell generates an internal table with all the individual item names and parameter names as well as complete names. If a user enters an incomplete name, TinkerCell will search this table and use the closest matching complete name. If no such name exists, TinkerCell will create a new parameter with that name. The new parameter will belong with the item under consideration.

Viewing and changing parameters

The easiest way to view all the parameters is probably the "Model summary", shown to the right. This window lists all the items in the model and their respective parameters.



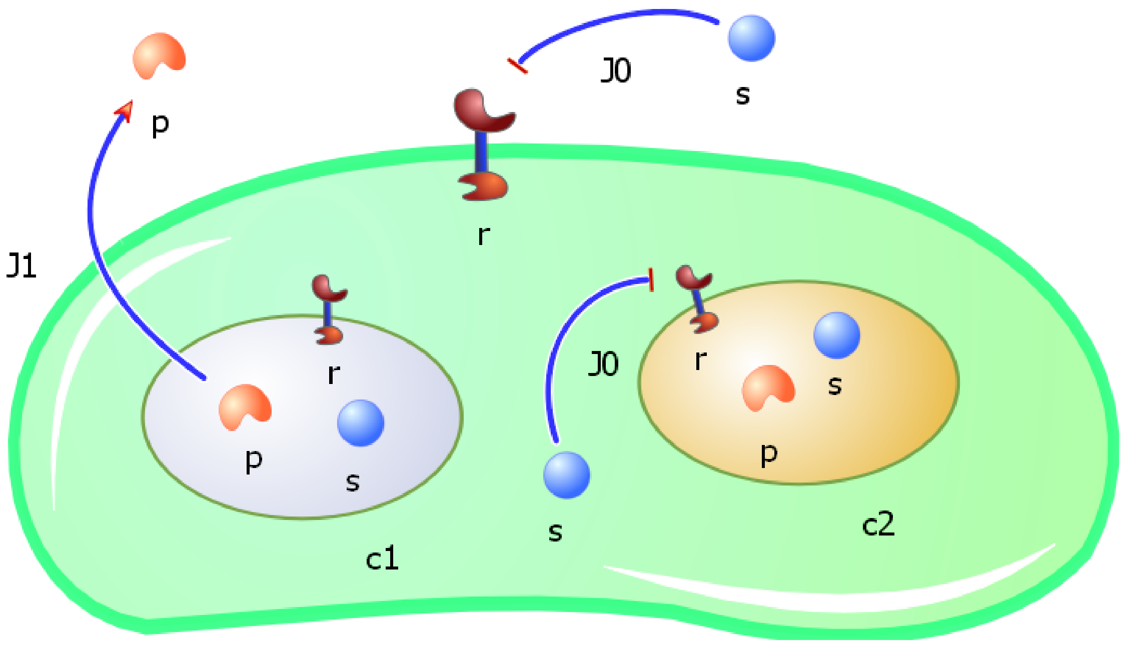
C	position	1
J0	h	1
c1	volume	1
p	concentration	1
r2	concentration	1
s	concentration	1
p	concentration	1
c2	volume	1
p	concentration	1
r	concentration	1
s	concentration	1
r	concentration	1
s	concentration	1
J0	h	1
J1	h	1
p	Kd	1
s	concentration	1

Another way to view and change parameters is by using the "Numerical Attributes" tool or the "Model Summary" tool.

The main reason for using local parameters is because the parameters can be loaded from a database or from some other source which might store information about a biological part. If parameters did not "belong" to a part, it would be nearly impossible to determine which parameters of the system to modify when a single part is loaded from a database.

Compartments

Compartments represent a space (volume) inside of which certain items are localized. In the screenshot below, two compartments, c1 and c2 belong inside a larger compartment, the cell. While the names appear to be redundant, the "complete" name is unique, because the complete name has the parent item's name as a prefix, which is explained in the next section. The parent of an item is the compartment containing that item.



Modules

<This section has been removed and needs to be re-written due to several recent changes>

Gene regulation networks

<This section has been removed and needs to be re-written due to several recent changes>

Events and forcing functions

There is an "Inputs" tab located next to the components and connections catalog. There are a list of forcing functions, such as sine wave, that are available here. There is also an icon for inserting events. Events are responses that are triggered under a specific condition. Insert an event on the screen and double-click in order to add or remove events.

Forcing functions can also be provided by selecting the "Functions" tool that is displayed when items are selected. This tool will list a set of functions defined inside the selected items. For example, if a user wants to assign a function to the molecule "s1", then the user should select this molecule, select the "Function" tool on the right, and then enter "s1 = s2 * (1 + sin(time))" or some other function of interest.

Text based model construction

TinkerCell supports graphical and text-based modeling. The text-based modeling is done via a third-party text parser (a plug-in). The text parser that is included with TinkerCell is called Antimony (antimony.sourceforge.net). Antimony is a modular model construction language. An example script is shown below.

```
Model M
  E + S -> ES;    kf * E*S
  ES -> E + S;    kb * E*S
  ES -> EP;   k1 * ES

  EP -> ES;   k2 * EP
  EP -> E + P;   kcat * EP

  kf = 1 + 0.6 *sin(time)
  kb = 1.2
  k1 = 0.1
  k2 = 0.9
  kcat = 0.3
end
```

User home folder

TinkerCell designated a folder as the user's home folder for temporary TinkerCell files. When TinkerCell loads plug-ins, it will also search this folder. So users without write-access to the main installation folder may use their TinkerCell home folder to place new plug-ins or Python scripts. By default, this folder is **<Documents folder>/TinkerCell**, where <Documents folder> is the default documents directory of the current user.

Command-line

<This section has been removed and needs to be re-written due to several recent changes>

PySCeS

All functions in the PySCeS module can be used inside TinkerCell's command prompt. PySCeS model is loaded by invoking **mod = tc2pysces.load()** (tc2pysces stands for TinkerCell-to-PySCeS). The returned variable **mod** will be a PySCeS model. Please see the PySCeS Documentation for all the functions available.

Programming

C, Python, and Octave functions can be written in TinkerCell, and these functions can be added to the list of programs in TinkerCell. See TinkerCell C API.