Notions

element The smallest, indivisible entity which may be drawn from an **urn**. Common elements are whole numbers like ‘1’, ‘2’; colors ‘red’, ‘green’ or dates like ’16.07.’

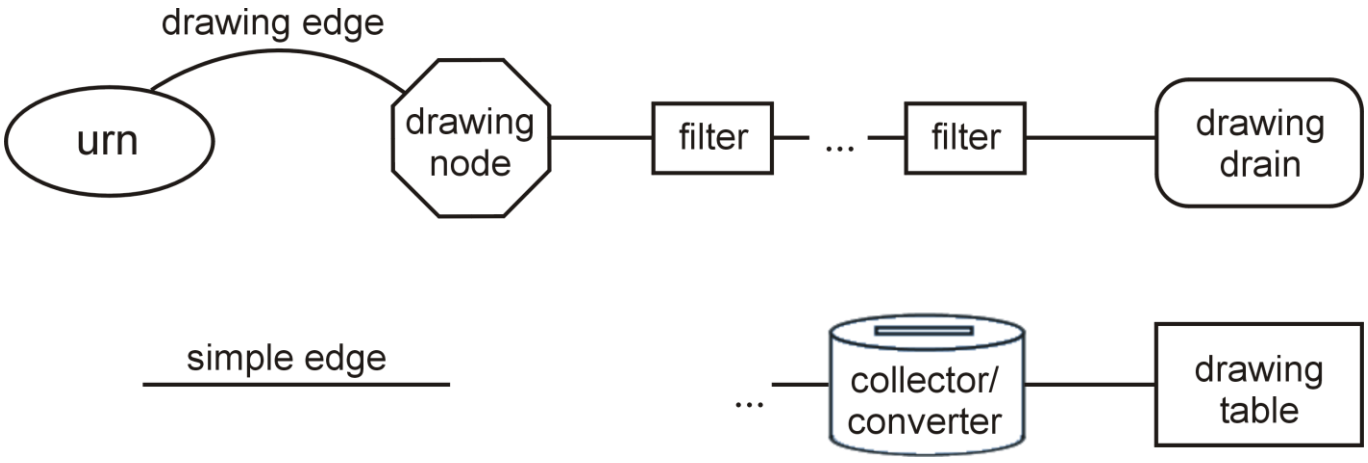
event One element or a combination of elements drawn from the urn(s) in *one* step; if only one urn is connected to a drawing node, the drawing events are single elements. Events consisting of more than one element may be generated by combined drawing from several urns or by using the converter.

NOTE: The number of elements in an event has nothing to do with the weight of the edge, because this tells how many elements should be drawn from the urn one *after* the other.

experiment When engaging the ‘execute’ button of the simple drawing node in your model (or multiple drawing node with value ‘1’) you will perform *one* experiment. The result of this experiment will be a table produced by the drawing node. Each *row* of this **drawing table** describes one event. Each *column* of the table (except the first) is assigned to a certain urn. Supposed each urn contains sufficiently much elements, the number of rows is given by the largest weight of all edges and the number of columns is given by the number of urns.

Syntax

Any stochastic model is a graph in the original mathematical sense consisting of nodes and edges connecting two nodes. Therefore the graphical syntax of a correctly modelled stochastic szenario can be defined by a BNF-like formalism describing the structure of *any trace* through the whole graph beginning at a certain class of nodes. In our case these are the urn nodes, and a correct trace is shown in the following picture:



The number of filters between a drawing node and a drawing drain is unlimited and may also be zero. All the nodes or classes of nodes correspond to the categorization used in the plug-in palette and depicted in the following. One exception is the following class:

drawing drain = collector | converter | drawing table | estimation node | description node

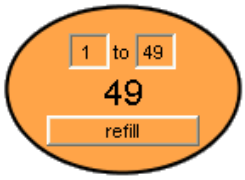
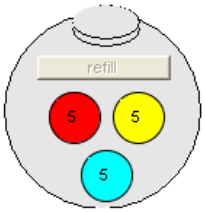
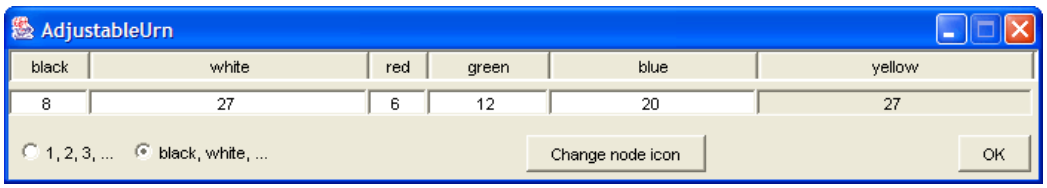
Although it is generally senseless to connect two drawing drains, there is again one exception: You may connect a drawing table to a collector or to a converter as shown in the picture.

Finally the drawing edge class is defined as follows:

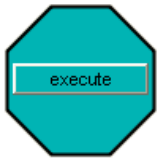
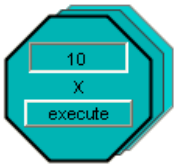
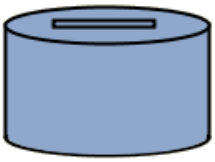
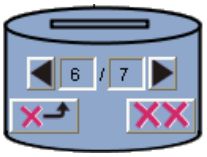
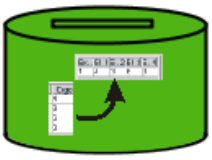
drawing edge = pick edge | pick-and-put edge

Nodes

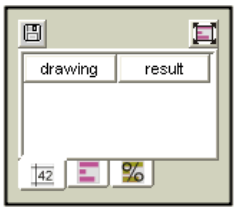
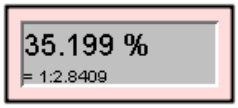
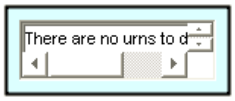
Urn nodes

	**Lottery bowl** Contains subsequent whole numbers (also negative values) in a given interval. The number in the center of the node's shape tells you how many elements actually are in it. By pressing 'refill' the initial state is been produced.
	**Calendar** Contains all days of a (leap-)year, i.e. 366 different date values. A 'new' calendar looks like the image on the left. As soon as some leafs are removed, the shape looks like shown on the right. Again, 'refill' will produce a completely new calendar. 
	**Globe urn** Contains balls of three different colors. The initial number of balls may be specified individually for each color by editing the numbers in the balls. Whether the actual state of the urn is the initial one can be seen by the 'refill' button: When it is inactive, the initial state is present.
	**Dice** Contains the whole numbers '1' to '6'. In contrast to the urns described above, the elements of this and the two following urns may *not* be removed. Therefore you always draw *with put-back* – no matter which edge is used.
	**Tetrahedral** Contains the whole numbers '1' to '4'. As to the dice, these elements are not removable.
	**Coin** Contains just the two elements 'face' and 'verso', which of course are not removable.
	**Adjustable Urn** This urn *always* contains exactly 100 elements, which are not removable. By engaging the ‘adjust’ button you will see the following dialog that allows you to modify the composition of these 100 elements:  As you can see, the number of *different* elements is restricted to six, and the elements themselves may be numbers or colors. The portion of each type of element may either be adjusted by moving the seperators between their labels or by typing in a value into the field beneath.The latter also allows you to set a certain value to ‘0’.

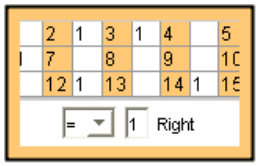
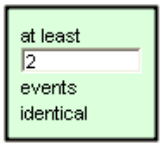
Drawing & Collecting

	**Simple drawing node** This central element of every stochastic model performs the experiment itself by collecting the wanted number of elements from all the connected urns and passing them through to all the connected drawing sources.
	**Multiple drawing node** Same function as the simple drawing node, but with the ability to repeat the given experiment a desired number of times. Setting the node's value to '10' e.g. is equivalent to using the simple drawing node and pressing 'execute' *and* afterwards all the urns' 'refill' buttons for 10 times. During the execution of the series of experiments the procedure may be interrupted at any time by pressing 'STOP'. If done so, the currently running experiment will just be finished and the number of experiments left out will be filled into the node's value field.
	**Collector** This node collects all the drawing results given to it in several tables. Each table contains the elements drawn within *one* experiment. Connecting the collector to a multiple drawing node with value '10' will produce 10 tables. The picture on the left shows an empty collector. When there are tables stored in it, the node provides a control panel (picture on the right) which helps to display the collector's content using a *simple drawing table node*. The collector's capacity is limited to 1000 tables. 
	**Converter** The converter node fulfills two jobs: Rearanging incoming drawing results and storing them alltogether within a single drawing table. In order to achieve this the drawing results passed to this node are converted in the following way: All the elements in the result of one experiment are arranged in a single table row (as if they were all drawn from different urns at one time). This row is then simply added to the table already stored in the converter. The table's size is limited to 10000 rows.

Display nodes

	**Drawing table** This node is the universal display component for the experiments' results. It is able to show the drawn elements either chronologically in a table or graphically using bars that represent the frequency in which all different elements (or combinations of them) have occurred. For the second case you can further choose between an absolut and a relative view (giving percentage values). Within your model this node can be used in two different ways: Firstly to directly display drawn elements and secondly to display the content of a collector. The whole table actually displayed in this node may be exported to a *csv*-file (comma seperated values) which e.g. can be imported into an *MS Excel*-worksheet. You may enlarge the node in order to provide more space to the data view. But because this space gets lost for other objects in your workspace one can use the button on the top rigth of the node to open a separate window in which the node's content may be displayed in any size. This window additionally allows to enlarge the font size for usage in a presentation.
	**Evaluation table** In order to summarize the main results of your experiments this simple node is useful. It is designed as a stand-alone component into which you type in the information you are interested in by yourself. In order to corporately evaluate a certain problem by different groups this component may be used in synchronized workspaces.
	**Estimation node** In connection with some specific stochastic scenarios like the lotto-game or the birthday problem, this node is able to exactly *calculate* probabilities. In order to do so you have to specify which of all the *possible* results of your experiments you would call '*successful*'. This may be done using filters. If possible the estimation node will finally give the expected portion of successful experiments. A ratio of 20% would e.g. mean that a collector at the same place as this node is expected to contain 20 (positive!) results after performing 100 experiments!
	**Description node** As also to the estimation node this component is used at a place in your model where usually a collector would make sense. The node's purpose is to give a linguistic description of the modelled experiment in a 'headword' style. Mainly the used urns, the properties of the corresponding drawing edges and the filters in between are listed.

Filters

	**Lottery coupon** This is the most poweful filter because you can make issues concerning every single type of element used in your experiment. This is realized as follows: The filter generates a table containing all elements that exist in the *first* urn been drawn from exactly one time. A tetrahedral in your model would e.g. lead to the set {‘1’, ‘2’, ‘3’, ‘4’} whereas in case of the globe urn the table of choice would be {‘cyan’, ‘red’, ‘yellow’}. Into the white table cells next right to the elements you may type in a number which defines your bet of how often this specific element will finally appear in the experiment’s result. Leaving white cells empty means that you don’t care about these elements. If you guess that an element will not appear in the result at all, you have to fill in a ‘0’. After having placed all your bets you have to specify how *many* of them must conform to a single experiment’s result so that it will be accepted by the filter. This is done by selecting a relation (‘<’, ‘=’, ‘>’, ..) and giving a number, which of course may not exceed the total number of bets you have placed.
	**Equalization node** Think of being interested in results in which elements appear repeatedly, regardless of which elements these are. In generall this may also be expressed using the lottery coupon filter. In case of the birthday problem it would however mean to place the same bet (e.g. ‘2’) to all of the 366 possible days of a year! To avoid this work, the equalization node was invented. You just define how often any element must at least eppear within a single experiment so that this one is accepted by the filter. The filter does not care about how *often* this condition is fulfilled within one experiment. More precisely the semantics of this filter can be described as “Accept results in which at least one time at least n events are identical”. The expression ‘event’ denotes either a single element or a set of elements (produced by combined drawing or the converter node).
	**Sorter** The sorter is not a filter in the sense of the two components described above, because it does not take away any of the produced results. Its purpose is to just sort the elements given to it alphanumerically. The reason why this node is assigned to the filter category is that it may be located exactly at the same positions within your graph as it is to the ‘real’ filters. The way in which a given drawing table is actually sorted depends on the number of columns: When the tables consist of just two columns (i.e. drawing events are single elements) the whole result *column* will be sorted. As soon as the table has more than two columns (i.e. drawing events consist of two or more elements) each *row* will be sorted independently. Therefore this node may be used to distinguish between experiments in which the order of elements is relevant (combinations of elements) and those where the order does not matter (sets of elements).

The Drawing Table Node

The drawing table node is the universal display component for the experiments' results. Figure 1 shows a simple dice game model in which a drawing table node is used in order to display the content of a converter node. Because the die is thrown twice per experiment, the converter generates a result table with three columns. The only 'filter' used here is a sorter node so that, corresponding to the 100 experiments, the table will contain exactly 100 rows.

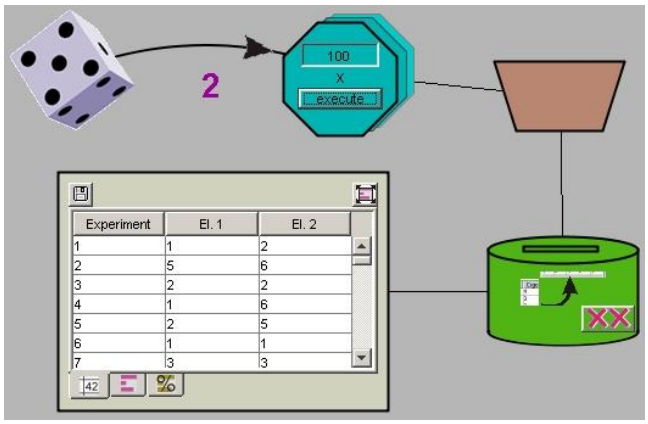


fig.1: A dice game model

The drawing table node provides three different views of the data obtained: The current 'table view' that shows the experiments' results in the chronological order they have occurred and two 'bar chart views', which show how often all the different *events* have occurred totally. These views differ in whether the total number or the percentage ratio of a certain event is of interest. You may switch between the three views easily using the tab buttons at the bottom of the node. These and all the other components of the node's GUI are explained in figure 2. As you can see from the pictures, neither the table view nor the absolute bar graph fits into the current display area of the node. Therefore a scroll bar is given in both cases. If you like to display a greater amount of data at the same time without resizing the node, make use of the enlarge button. This will open the current view in an own window (figures 3, 4) that may be resized up to the whole size of your desktop. The view in fig. 3 shows a total number of 21 different events, which is a significant result due to the fact, that all pairs of numbers thrown are sorted.

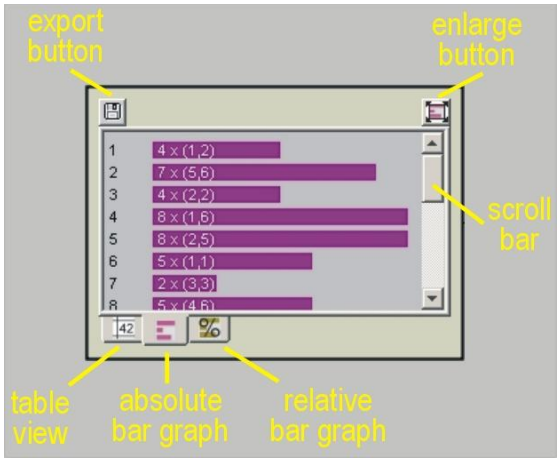


fig. 2: The drawing table node's UI

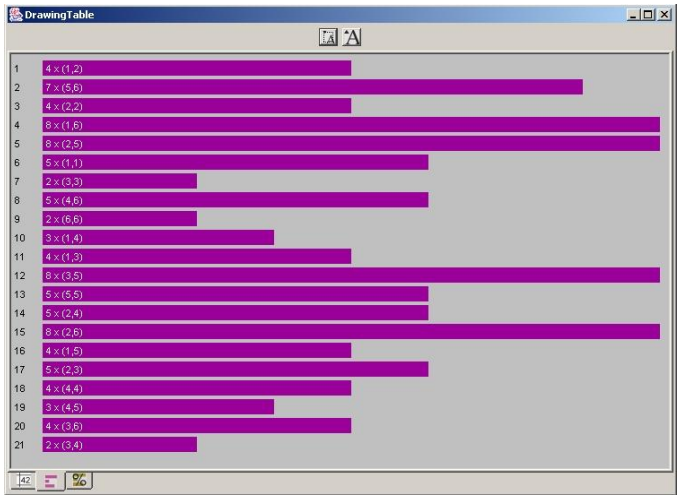
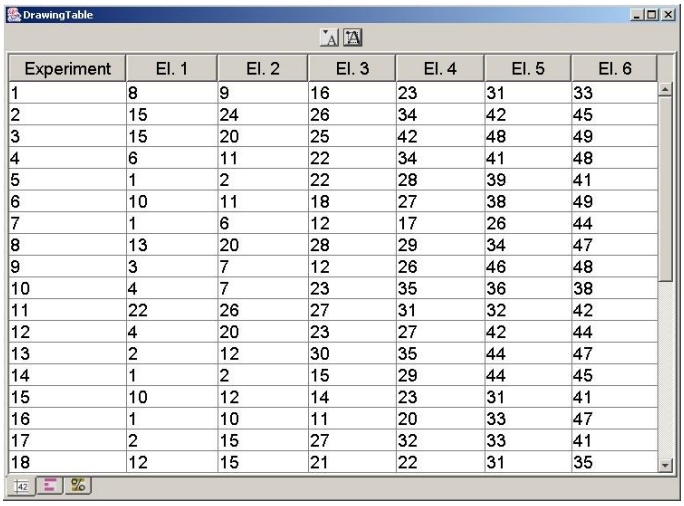


fig. 3: Enlarged absolute bar graph view



Experiment	El. 1	El. 2	El. 3	El. 4	El. 5	El. 6
1	8	9	16	23	31	33
2	15	24	26	34	42	45
3	15	20	25	42	48	49
4	6	11	22	34	41	48
5	1	2	22	28	39	41
6	10	11	18	27	38	49
7	1	6	12	17	26	44
8	13	20	28	29	34	47
9	3	7	12	26	46	48
10	4	7	23	35	36	38
11	22	26	27	31	32	42
12	4	20	23	27	42	44
13	2	12	30	35	44	47
14	1	2	15	29	44	45
15	10	12	14	23	31	41
16	1	10	11	20	33	47
17	2	15	27	32	33	41
18	12	15	21	22	31	35

fig. 4: Enlarged table view

Another useful feature of the drawing table node is the export function. Pressing the export button will open a file save dialog in which you can specify where to save a file containing the whole table. This is a '.csv'-file (comma separated values), which may e.g. be imported by Microsoft Excel in order to apply further calculations on the data, to generate other types of charts, to print the table, etc.