

# JDecisiontable Manual

## Table of Contents

Copyright of this document.....	3
Conventions.....	3
Overview.....	4
A first look.....	4
Left hand table.....	8
Right hand table.....	9
Short cuts.....	12
Cut, copy and paste.....	13
Menu DecisionTable.....	14
New, Open, Save and so on.....	14
New from Node Descriptions.....	14
Export to Csv.....	17
Copy to Clipboard.....	18
Menu Node.....	19
New Node.....	19
Remove Node.....	21
Copy Node.....	23
Menu Rule.....	27
New Rule.....	27
New Empty Rule.....	33
Remove Rule.....	35
Copy Rule.....	38
Menu Verify D.table.....	40
Run All Checks.....	41
Check Nodes_Dontcare.....	45
Check Rules_Disjunct.....	46
Check Rules_Number.....	48
Show actual num. of th. rules.....	50
Show expected num. of th. rules.....	50
Menu Tools.....	51
Clear Check Results.....	51
Sum Probabilities.....	52
Show Nodes without Y in Valid Rule.....	53
Undo/Redo.....	54
Screenshot.....	54
Menu TestSpecification.....	55
Menu Options.....	56

Make Rows Higher.....	56
Use Icon Set 2.....	57
Use English.....	57
Menu Help.....	57
About, Help and License.....	57
Command Line.....	57
Option File.....	58
Linux.....	58
Windows®.....	58
File Formats.....	58
For translators.....	61
Please do not rely on the undo/redo feature – use version control.....	64

## Copyright of this document

Copyright 2012 - 2016 Michael Groß, [mgmechanics@mgmechanics.de](mailto:mgmechanics@mgmechanics.de),  
<http://sourceforge.net/users/mgmechanics>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Conventions

Usually there are two screen-shots for one action. The first screen-shot shows the state before action and the later ones shows the result of the action.

First-class headlines starting with "Menu" represent the main menus, second-class headlines in this chapters the menu items.

"Disk" represents any volume build-in in your machine. In most machines this is the hard disk. It does not mean a removable volume e.g. an USB stick.

Due to the large amount of screenshots in this book we can not update each screenshot when the use interface of the software changes. Therefore we update only those screenshots we need to show new features or when they became confusing. If the changes doesn't matter for a certain screenshot we will keep it.

## Overview

### A first look

The screen-shots are made with “Tools” > “Screenshot”. This function can not capture the window title.

DecisionTable Node Rule Verify D.Table Tools TestSpecification Options Help

● dyeingSilk.5dt x

● rangeOfNumbers.5dt x

ID	Cond?	Description	Probab...	Co...
1	x	use red color	0%	=
2	x	use blue color	0%	=
3		dyeing silk purple	0%	=
4		dyeing silk red	0%	=
5		dyeing silk blue	0%	=

1	2	3	4	5	6	7	8	9	10	11	12	13
Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
Y	Y	N	N	N	Y	Y	Y	Y	N	N	N	N
Y	N	Y	N	N	Y	N	N	N	Y	N	N	N
-	-	-	Y	N	-	Y	N	N	-	Y	N	N
-	-	-	-	-	-	-	Y	N	-	-	Y	N
x			x				x					x
												whiteSilk.5dt

DecisionTable Node Rule Verify D.Table Tools TestSpecification Options Help

● dyeingSilk.5dt x

● rangeOfNumbers.5dt x

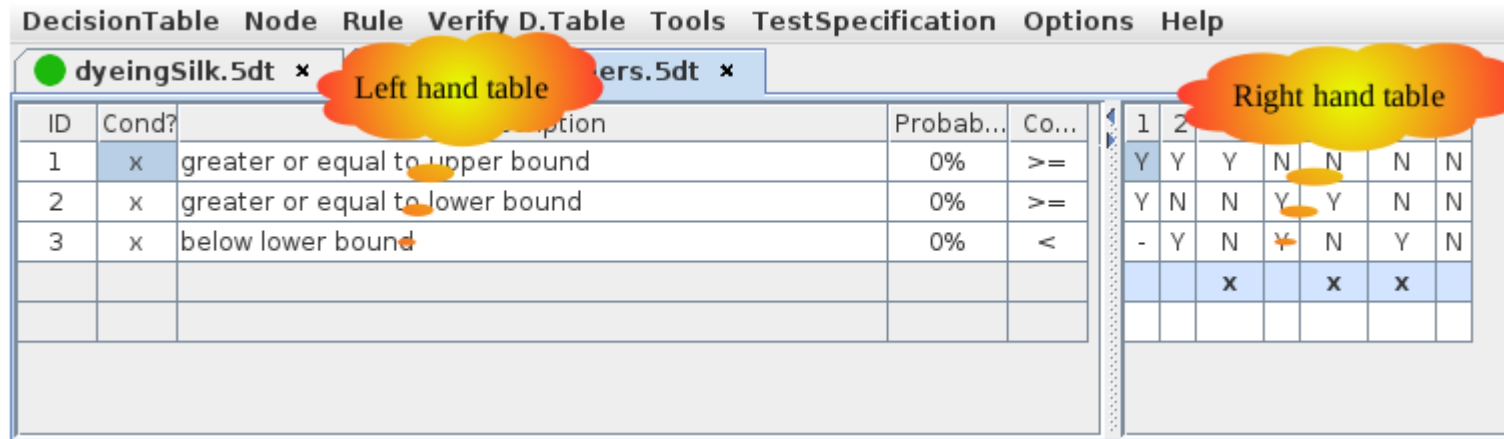
ID	Cond?	Description	Probab...	Co...
1	x	greater or equal to upper bound	0%	>=
2	x	greater or equal to lower bound	0%	>=
3	x	below lower bound	0%	<

1	2	3	4	5	6	7
Y	Y	Y	N	N	N	N
Y	N	N	Y	Y	N	N
-	Y	N	Y	N	Y	N
		x		x	x	

The green dot means that this decision table passed all checks. It changes to red if one check fails and to yellow if the decision table was changed. To execute the checks run “Verify D. Table” > “Run All Checks”.

This application can handle multiple tabs. Each tab contains one decision table. Each tab contains two tables.

- The left hand table contains the nodes
- The right hand table handles the rules



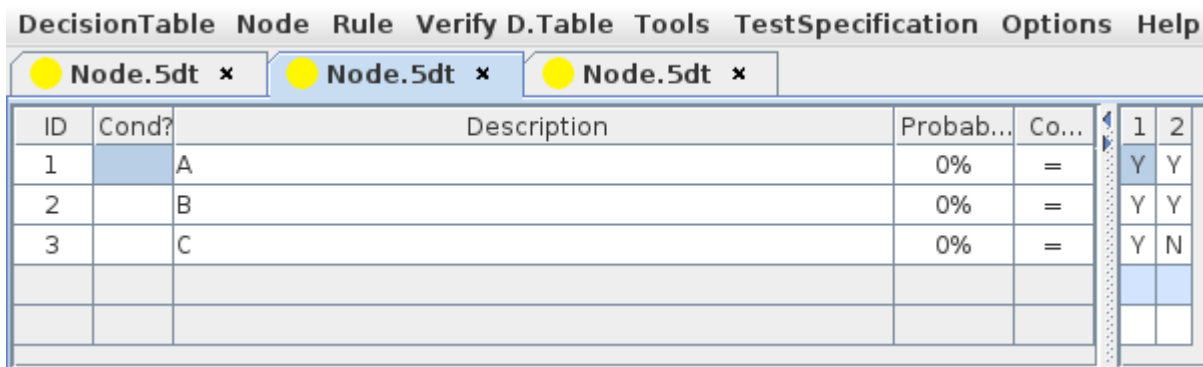
The screenshot shows the 'DecisionTable' application window with a menu bar (DecisionTable, Node, Rule, Verify D.Table, Tools, TestSpecification, Options, Help) and two tabs: 'dyeingSilk.5dt' (active) and 'ers.5dt'. The left table, labeled 'Left hand table', has columns: ID, Cond?, Description, Probab..., and Co... It contains three rows of data. The right table, labeled 'Right hand table', has columns 1 and 2, and contains a 3x6 grid of Y/N values.

ID	Cond?	Description	Probab...	Co...
1	x	greater or equal to upper bound	0%	>=
2	x	greater or equal to lower bound	0%	>=
3	x	below lower bound	0%	<

1	2					
Y	Y	Y	N	N	N	N
Y	N	N	Y	Y	N	N
-	Y	N	Y	N	Y	N
x	x	x	x	x	x	x

Left hand + right hand table = decision table.

Each decision table is independent from other decision tables and saved into one file.



The screenshot shows the 'DecisionTable' application window with the same menu bar. There are three tabs, all named 'Node.5dt'. The left table, labeled 'Left hand table', has columns: ID, Cond?, Description, Probab..., and Co... It contains three rows of data. The right table, labeled 'Right hand table', has columns 1 and 2, and contains a 3x2 grid of Y/N values.

ID	Cond?	Description	Probab...	Co...
1		A	0%	=
2		B	0%	=
3		C	0%	=

1	2
Y	Y
Y	Y
Y	N

Each row in the left hand table is one node preceded by a number, the node number. Each column in the right hand table is a rule topped by a

DecisionTable Node Rule Verify D.Table Tools TestSpecification Options Help

dyeingSilk.5dt x rangeOfNumbers.5dt x

ID	Cond?	Description	Probab...	Co...
1	x	use red color	0%	=
2	x	use blue color	0%	=
3		dyeing silk purple	0%	=
4		dyeing silk red	0%	=
5		dyeing silk blue	0%	=

Node number

Rule number

Both tables are divided by a bar which you can move to left or right. It comes with two arrows (red circle). Click on one of them to make the left or right hand table disappear.

DecisionTable Node Rule Verify D.Table Tools TestSpecification Options Help

dyeingSilk.5dt x

rangeOfNumbers.5dt x

ID	Cond?	Description	Probab...	Co...	1	2	3	4	5	6	7
1	x	greater or equal to upper bound	0%	>=	Y	Y	Y	N	N	N	N
2	x	greater or equal to lower bound	0%	>=	Y	N	N	Y	Y	N	N
3	x	below lower bound	0%	<	-	Y	N	Y	N	Y	N
							x		x	x	

You may also change the width of the columns in the left hand table. The width of columns in the right hand table is handled by the application.

Not all columns are editable as in a spreadsheet. These columns are limited to certain values. They change their value when you focus a cell and then type a certain key on your key board. This is pointed out in the next chapter.

## Left hand table

The left hand table handles only the nodes of the decision table. Nodes are organized in rows. Each row shows the parts (fields) of one node.

Field	Purpose	Editable?	Limited?	Limited to
ID	The node number	No	Yes	Generated by the application.
Cond?	[x] = this node is a condition [ ] = this node is an action “Cond?” means “is condition”	No	Yes	“x”, DEL key, BACKSPACE key [ ← ]
Description	Contains either what the condition is or which action should performed.	Yes	No	I would avoid to include tab [ ⇥ ] and linefeed chars here. There is really no need to format text here. You may use the option to make the rows higher to distinct nodes visibly.
Probability	Contains the probability that this condition is true or that this action is performed.	Yes	Yes	Numbers, decimal delimiter and “%” char e.g. “2” → 2.0%, “2.1” → 2.1%, “3%” → 3.0 % If you leave the field empty it will be filled with default value again. Same if you type some chars outside the limits (e.g. a letter) and leave the field.
Comparison	Contains the comparison. It is mainly used when dealing with ranges - see screenshots .	No	Yes	Type a number from range 1..5. 5 → “>”, 4 → “>=”, 3 → “=”, 2 → “<=”, 1 --> “<” <sup>1</sup>

---

<sup>1</sup> Mnemonic: 5 is the biggest number, it is greater than the other numbers; 1 is the least number, it is less than the other numbers.



**There is an easy trick to delete existing values in editable fields:**

- select the cell
- type Backspace key [ ← ] several times until the selected cell is empty
- type new values

### ***Right hand table***

The right hand table handles the rules of the decision table. Rules are organized in columns. Each column contains the parts (fields) of one rule.

The table header contains the number of each rule. If you insert or remove a rule they get new numbers.

Below the numbers there are **decisions** (hatched area). Depending on internationalization each letter or char means:

Letter / char	Meaning / decision
Y	Yes
N	No
-	Don't Care

Each rule can be valid or not. The row before the very last row (coloured with light blue background) contains the **isValid** flag.

DecisionTable Node Rule Verify D.Table Tools TestSpecification Options Help

dyeingSilk.5dt x

rangeOfNumbers.5dt x

ID	Cond?	Description	Probab...	Co...	1	2	3	4	5	6	7	8	9	10	11	12	13
1	x	use red color	0%	=	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
2	x	use blue color	0%	=	Y	Y	N	N	N	Y							N
3		dyeing silk purple	0%	=	Y	N	Y	N	N	Y							N
4		dyeing silk red			-	-	-	Y	N	-							N
5		dyeing silk blue			-	-	-	-	-	-		Y	N	-	-	Y	N
					x			x				x					x
																	whiteSilk.5dt

This rule is valid

This rule is not valid

The very last row contains the field **Successor** telling that after this rule you may succeed with another decision table (a file named “whiteSilk.5dt” in rule 13 in the example).

DecisionTable Node Rule Verify D.Table Tools TestSpecification Options Help

dyeingSilk.5dt x

rangeOfNumbers.5dt x

ID	Cond?	Description	Probab...	Co...	1	2	3	4	5	6	7	8	9	10	11	12	13
1	x	use red color	0%	=	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
2	x	use blue color	0%	=	Y	Y	N	N	N	Y	Y	Y	Y	N	N	N	N
3		dyeing silk purple	0%	=	Y	N	Y	N	N	Y	N	N	N	Y	N	N	N
4		dyeing silk red	0%	=	-	-	-	Y						-	Y	N	N
5		dyeing silk blue	0%	=	-	-	-	-	-	-				-	Y		N
					x			x				x					x
																	whiteSilk.5dt

Successor

Some fields are not editable but change their value when you type certain keys on your keyboard.

Field	Purpose	Editable ?	Limited?	Limited to
decision	Contains the decisions of each rule. “-” and “d” toggle a “Don't Care”-decision	No	Yes	“y”, “n”, “d”, “-”
isValid	[x] = this rule is valid [ ] = this rule is not valid	No	Yes	“x”, DEL key, BACKSPACE key [ ← ]
Successor	Contains the successor. This another decision table to continue after finishing with this rule.	Yes	No	

## ***Short cuts***

All menu items used for day-by-day work may also be used by using a short cut.

“Ctrl-Q” means “press down the Control key, type “w” and release Control key.

“Ctrl+Alt-W” means “press down the Control key, press down Alt key, type “w”, release Alt key and release Control key.

“Ctrl+Shift-W” means “press down the Control key, press down Shift key, type “s”, release Shift key and release Control key.

Please note that “Ctrl-Q” does not mean to type an upper case Q – just type the key labeled with “Q”.

You may

- switch between left hand and right hand table using Ctrl-# or Ctrl-T (Mnemonic: # = table)
- navigate from cell to cell using Arrow keys, Tab, Shift-Tab, Enter and Shift-Enter
- start editing editable cells by going to this cell and just start typing, typing F2 or double-click

Currently there is no short cut to go from tab to tab. It would be easy to implement but it seems not possible to avoid any short cut used to switch from desktop to desktop.

Some short cuts are different than in well-known Windows applications e.g. Ctrl-Q (q stands for quit) rather than Alt-F4. The author was free to keep along common Linux desktop applications.

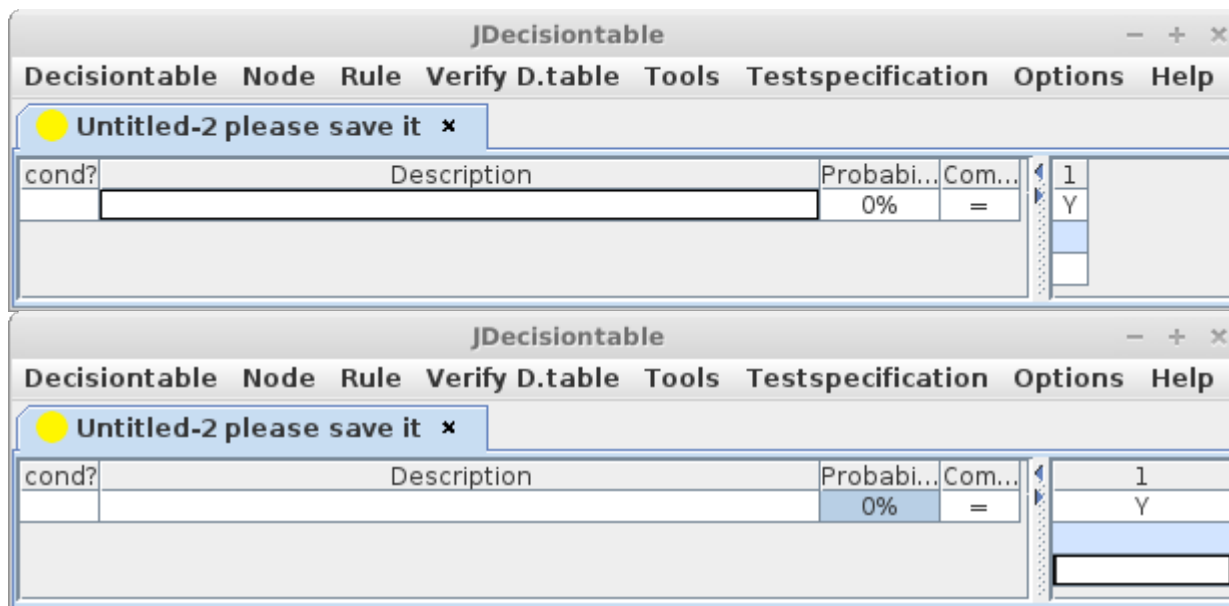
## Cut, copy and paste

Cut, copy and paste works fine with JDecisiontable in Linux and Windows® (both tested with JDecisiontable version 1.1.1) for the fields **Description** and **Successor** (only) if you bring these fields to edit mode e.g. by

- double click
- type F2
- type any letter, number, ... and place the cursor into the cell by clicking somewhere in the cell

Then use Ctrl+X to cut, Ctrl+C to copy and Ctrl+V to paste some text from/into the selected cell as usual.

The screenshots below show JDecisiontable with fields Description and Successor being in the edit mode.



## Menu DecisionTable

### ***New, Open, Save and so on***

This menus should work as you expect it from other applications with one exception: There is intentionally no decision table open when JDecisiontable just started. Please use Open or New to get one.

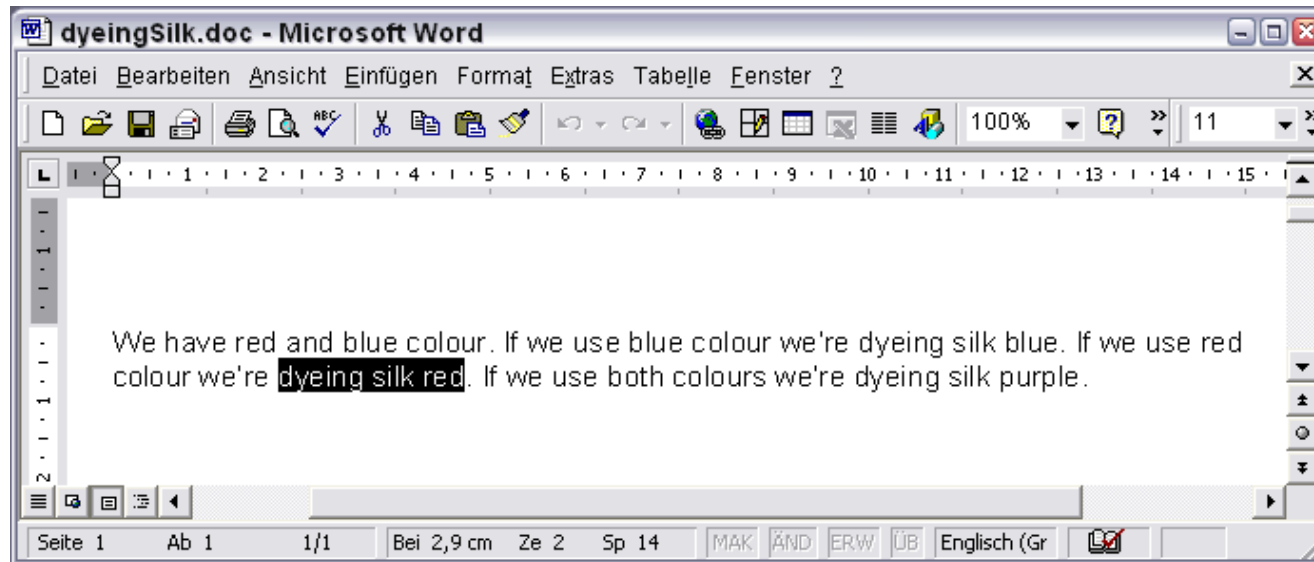
Menu item	Function
New	Creates a new decision table.
Open	Opens a dialog to open a decision table. The file name suffix for decision tables is <b>*.5dt</b> . <b>This are the only kind of files which you can open with JDecisiontable!</b> There are no facilities in JDecisiontable to open files as *.5ts (test specifications) or *.csv! These are for processing with other applications.
Close	Closes the decision table in the tab which is active yet. If you've changed the decision table but have not saved the changes yet it will ask you to save the changes now or to close the table without saving. The short cut is same as in Firefox (for Linux?) and Notepad++.
Close All	Closes all decision table in all tabs. If one table was changed but not saved yet it asks you to save your changes.
Save	If the decision table was created but not saved to disk it will raise a dialog where you may save the decision table. Otherwise it will save it quietly.
Save As	It will always raise a dialog where you may save the decision table. You do not need to type the file name suffix (*.5dt) after the file name – this is done automatically if the suffix is missed.
Save All	It will save all decision tables to disk. If one decision table was created but not saved to disk it will raise a dialog where you may save the decision table.
Exit	Closes all decision tables and quits the application. If one decision table was changed but not saved to disk yet it will ask you if you want to save the decision table.

### ***New from Node Descriptions***

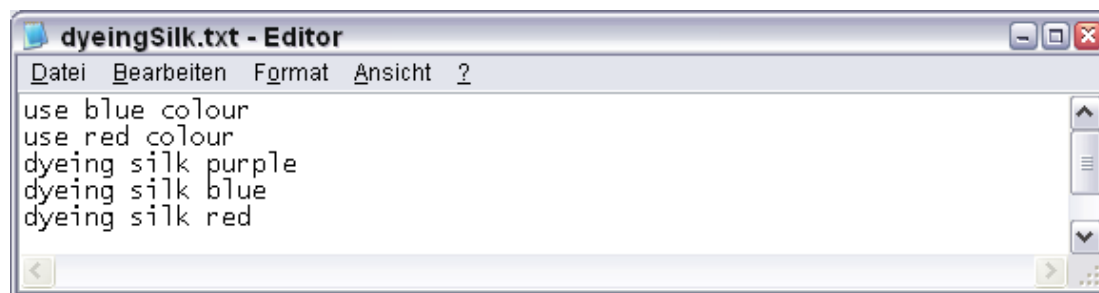
This menu item let you create a new decision table using a text file to read the node descriptions from. One line is used as one description so here will be as much nodes as there are lines in the text file. All fields except Decision will have same default values as with Decisiontable →

New. Also there is one rule only starting with “Yes”. How can you use it?

<sup>2</sup>First, copy some text from word processor or spread sheet e.g. Microsoft® Word® 2000:

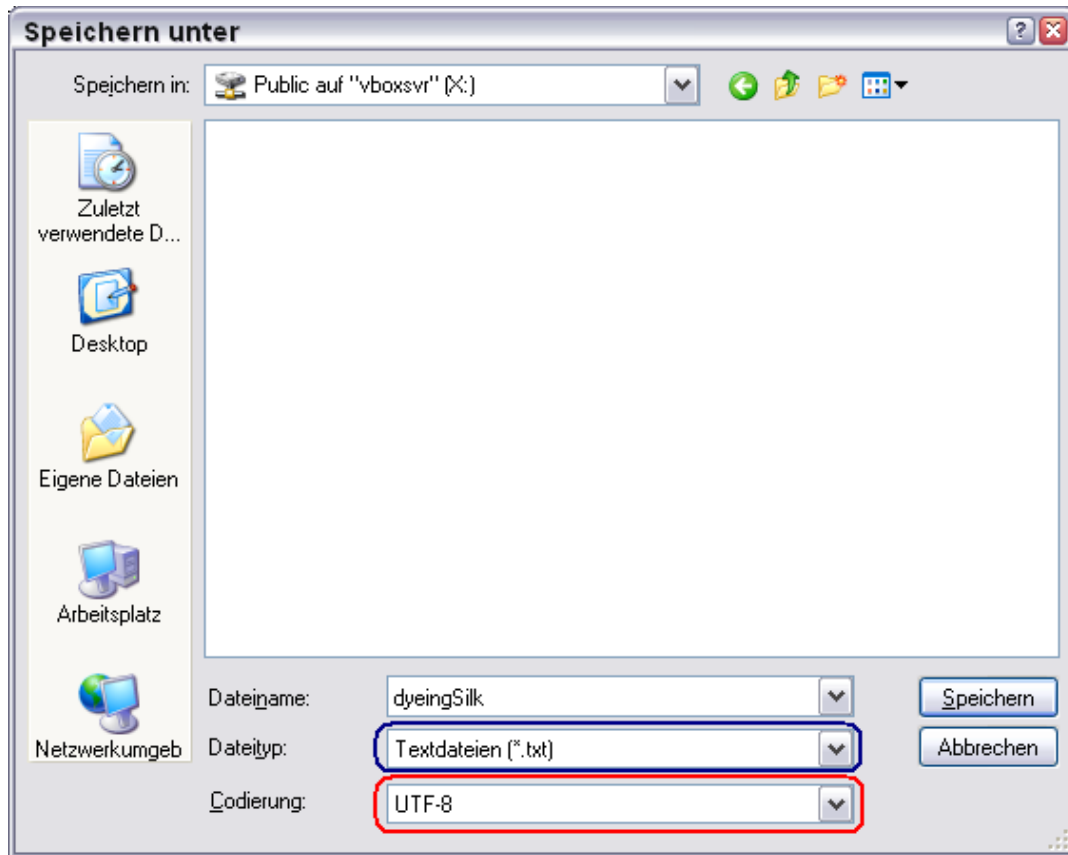


Then paste it to a text editor e.g. Accessories → Editor in Microsoft® Windows®:



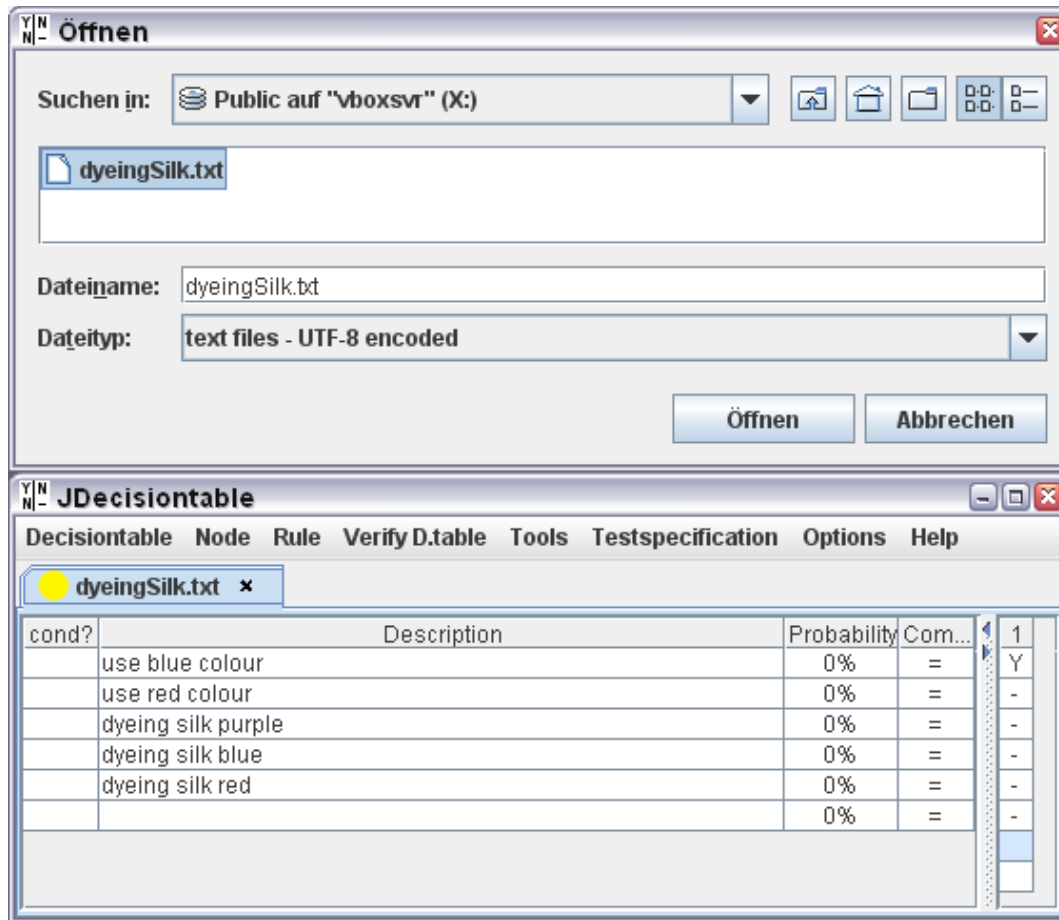
---

2 This way may doesn't look straight-forward but is (almost) fail-save. Don't use “ANSI” as encoding. See “File Formats” for details.



Now save the file using format "Text" and encoding "UTF-8" (this file is called "the text file"). You do not need to type the file name suffix. At last go to JDecisiontable and open the text file with Decisiontable → New from Node Descriptions:





**Please do not forget to save the file as decision table now!** The suffix "txt" in the tab is left with intend to remember this step.

## Export to Csv

This will write the decision table in the current active tab to a csv file. A File Save dialog will appear so you may give it a path there to save the file. Please note that there is no facility in this application to read it in again. Its for other application only. There is no way back<sup>3</sup>!

Before generating the file it will run all checks – same as you choose Verify D.table → Run All Checks. If the check passed it will just write "VALID"

<sup>3</sup> Since the regular file format is JSON you may write your own class or script which reads this csv into proper JSON. See "File Formats".

in cell 1,1. Otherwise there will be written “NOT VALID” in this cell. The application will not inform you in this case (no message pops up ...).

And this is what the files look like (left example generated from rangeOfNumbers.5dt).

VALID				1	2	3	4	5	6	7
x	upper bound	0.0	>=	Y	Y	Y	N	N	N	N
x	lower bound	0.0	>=	Y	N	N	Y	Y	N	N
x	below lower bound	0.0	<	-	Y	N	Y	N	Y	N
					x		x	x		

NOT VALID				1	2	3	4	5	6	7
x	upper bound	0.0	>=	Y	Y	Y	N	N	N	N
x	lower bound	0.0	>=	Y	N	N	Y	Y	N	N
x	below lower bound	0.0	<	Y	Y	N	Y	N	Y	N
						x		x	x	

Please see chapter “File Formats” for a technical specification of these files.

Hint: If you toggle the “Use English” option these reports will be formatted for the English locale.

## ***Copy to Clipboard***

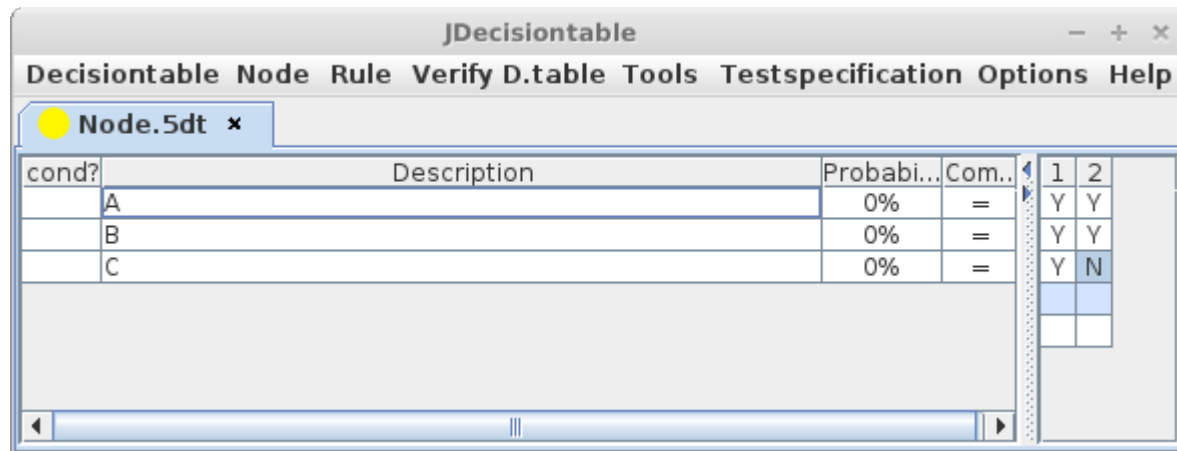
Copies the decision table as CSV to the clip board.

## Menu Node

### *New Node*

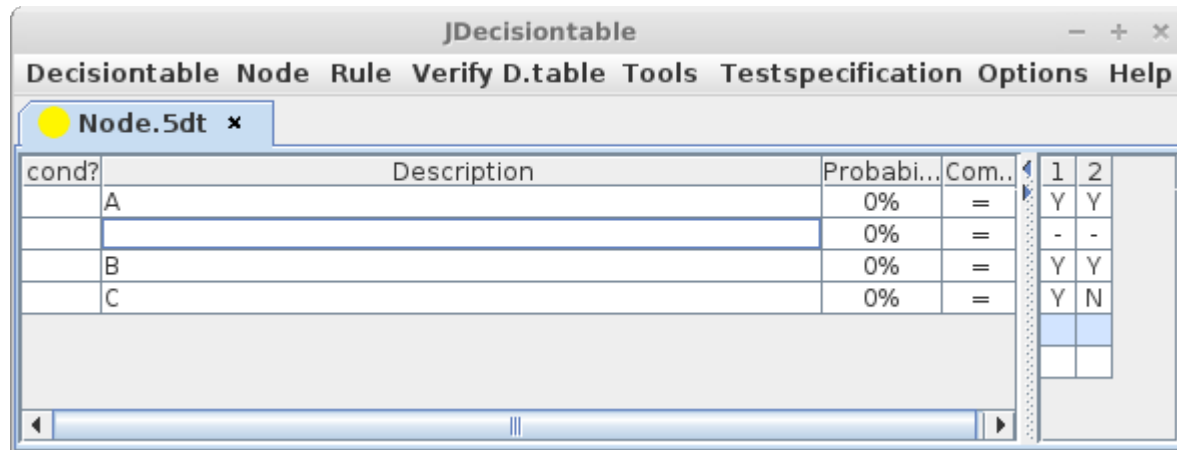
This creates a new node below the row of the cell which is selected.

**Please make sure that there is a cell selected in left hand table as shown below to make sure that the new node is inserted on the expected place.<sup>4</sup>**



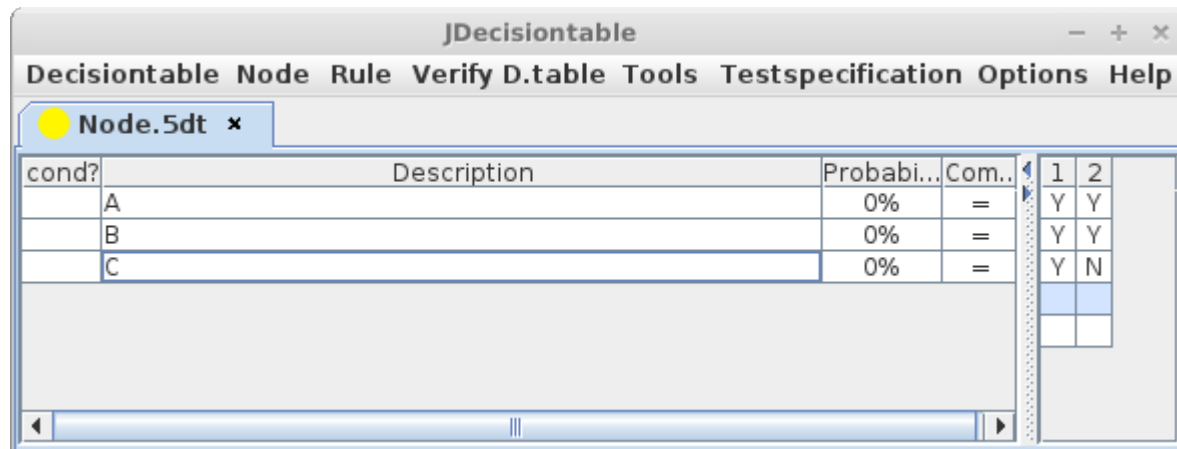
- 
- 4 If you click elsewhere in the application the left table will loose the focus. But it will still remember the cell selected at last. One could say: When left table lost the focus take selected cell from right table (assuming that the right hand table will have the focus). But if you click on a menu both tables loose the focus and both remember the last selected cell. Which cell is the one you want a new node below? As a fall back if there was no cell selected since table was opened it will insert the new node below the last node.

Of course this creates also a new row in all rules. This row is filled with Don't Care decisions.

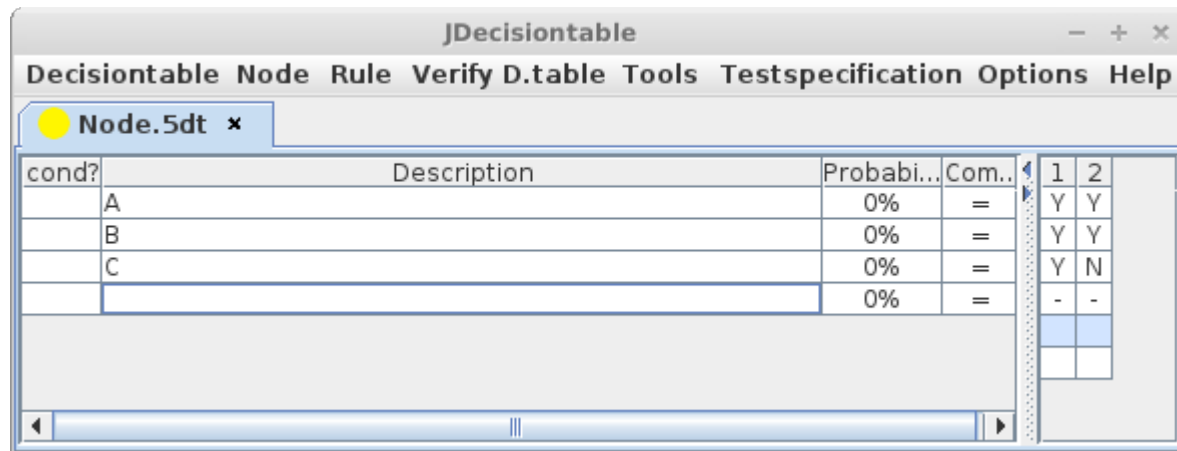


cond?	Description	Probabi...	Com...
	A	0%	=
		0%	=
	B	0%	=
	C	0%	=

This works of course also for the last decision. The new node is append after the last row.



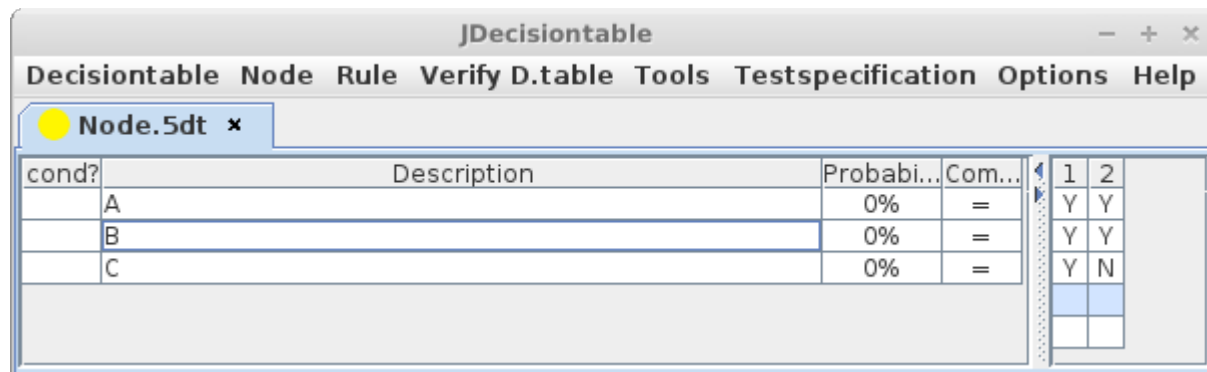
cond?	Description	Probabi...	Com...
	A	0%	=
	B	0%	=
	C	0%	=



## Remove Node

Removing nodes works same way as create new ones. Just select a node and select Node → Remove Node to remove it.

**Please make sure that there is a cell selected in left hand table as shown below to make sure that the expected node is removed.<sup>5</sup>**



<sup>5</sup> For same reasons as in Node → New Node. And as in Node → New node there is same fall back: If there was no cell selected since table was opened it will remove the last node.

And of course it will remove the cells in same row in right hand table.

**JDecisiontable**

Decisiontable Node Rule Verify D.table Tools Testspecification Options Help

Node.5dt x

cond?	Description	Probabi...	Com...
	A	0%	=
	C	0%	=

1	2
Y	Y
Y	N

Q: What would happen if I remove the last remaining node?

**JDecisiontable**

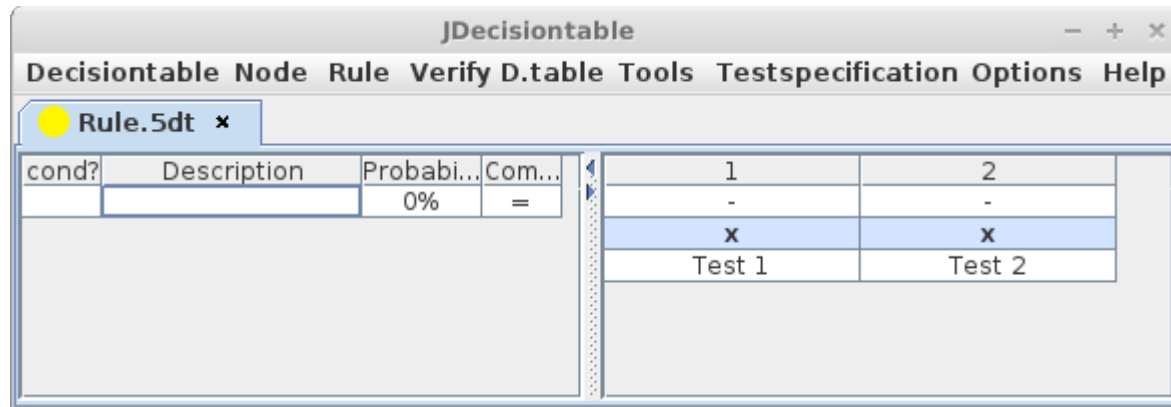
Decisiontable Node Rule Verify D.table Tools Testspecification Options Help

Rule.5dt x

cond?	Description	Probabi...	Com...
x	Test	34%	>

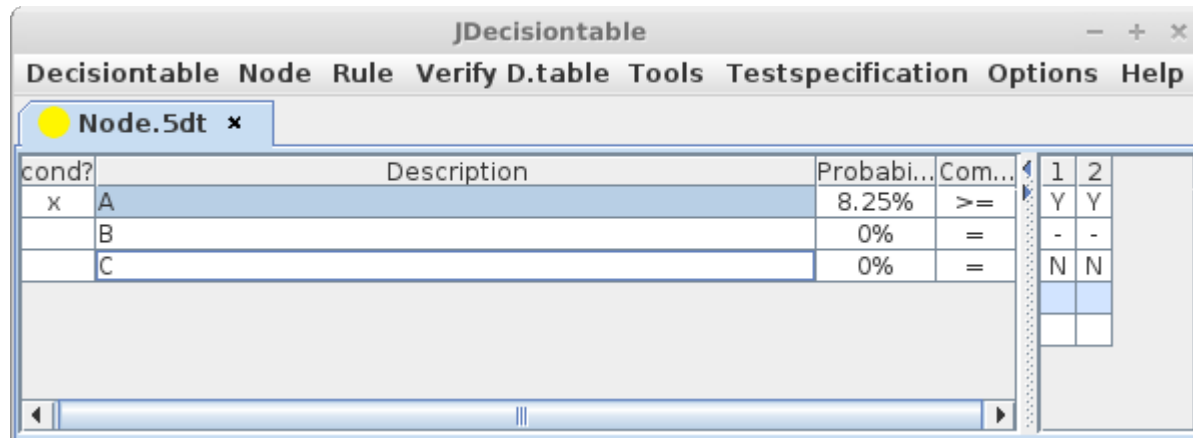
1	2
Y	N
x	x
Test 1	Test 2

A: JDecisiontable accepts your wish and removes this node but it creates a new one immediately. The rules remain but the decisions will be turned to Don't Care-decisions.



## Copy Node

To copy a file you need a source and a destination, isn't it? Same as for a node: You need a node to duplicate (= source) and a place where to insert the copy. You can tell these two items by selecting two nodes. The upper one will be recognized as source and the lower one as destination<sup>6</sup>. Like this:



<sup>6</sup> To be the destination means that the copy will be inserted below the selected cell.

You may select more than one cell by hold down Ctrl key while selecting. Please make sure to select cells in left hand table (only) to get the expected result<sup>7</sup>.

cond?	Description	Probabi...	Com...
x	A	8.25%	>=
	B	0%	=
	C	0%	=
x	A	8.25%	>=

Of course are the decisions of the selected node copied with this node.

The column of the selected cells doesn't matter - selecting the cell in the first row in column Condition and the cell in second row in column Comparison gives same result as selecting the cells in first and second row in column Description. It also doesn't matter which cell was selected first (in time). Only the row matters - if the cell is in left hand table.

You may also select a single cell:

cond?	Description	Probabi...	Com..
		0%	=
A		0%	=
B		0%	=
C		0%	=

<sup>7</sup> For same reason as Node → New Node. Fall back if no cell was selected before: It will copy the last node and append it below the last node.



In this case the selected cell is source and destination so a copy of this node will be inserted below this node.

cond?	Description	Probabi...	Com...
	A	0%	=
	B	0%	=
	B	0%	=
	C	0%	=

Selecting a range of cells results in copying the most upper selected cell below the last selected cell<sup>8</sup>:

cond?	Description	Probabi...	Com...
	A	0%	=
	B	0%	=
	C	0%	=
	D	0%	=

By the way - the focus is always set automatically to the copy:

<sup>8</sup> “upper” and “last” means the row of the selected cells.

JDecisiontable					
Decisiontable Node Rule Verify D.table Tools Testspecification Options Help					
Node.5dt x					
cond?	Description	Probabi...	Comp...	1	2
	A	0%	=	Y	Y
	B	0%	=	Y	Y
	C	0%	=	Y	N
	A	0%	=	Y	Y
	D	0%	=	-	-

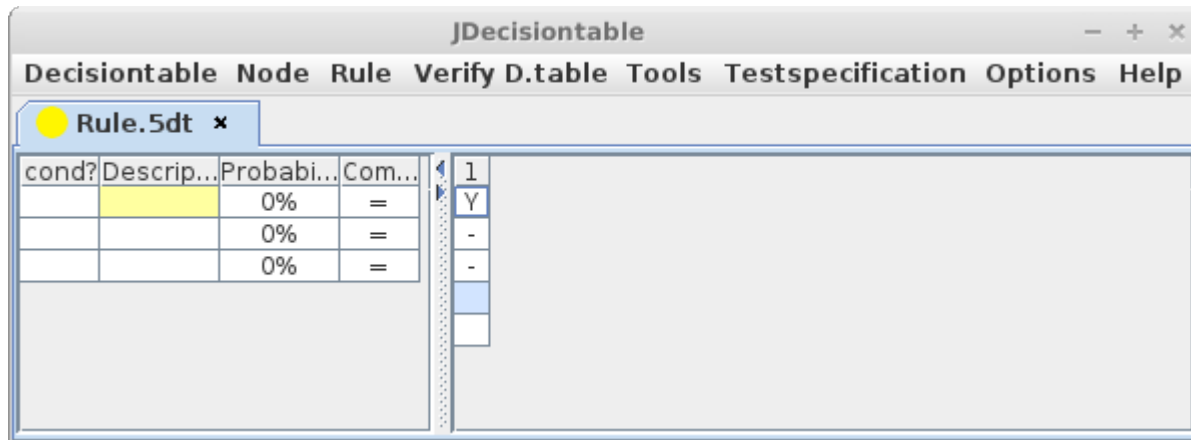
## Menu Rule

While there are two ways to get a new node (New Node and Copy Node) there are three ways to get a new rule:

- **New Empty Rule** is similar to New Node
- **Copy Rule** is similar to Copy Node
- **New Rule** and New Node are “false friends” since they do completely different things<sup>9</sup>

## New Rule

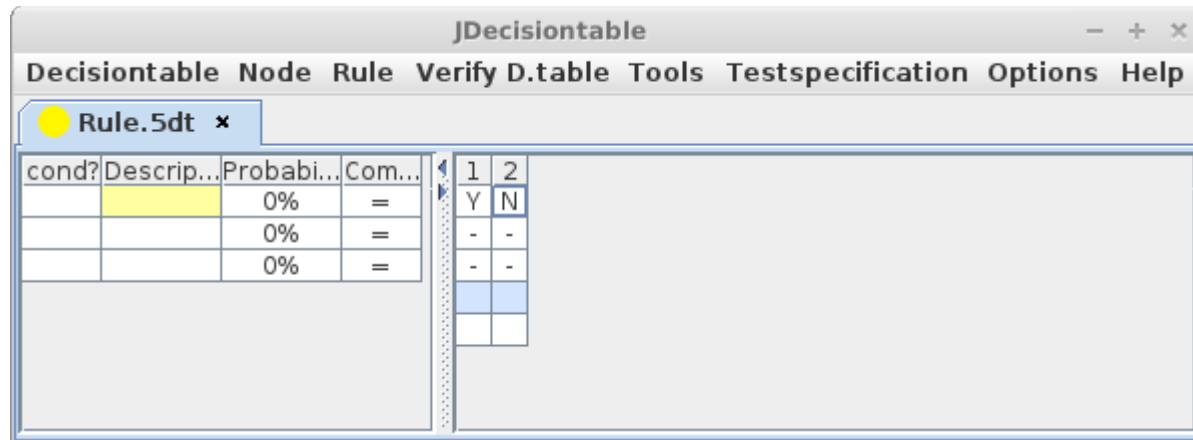
New Rule does not create a rule – it copies a rule and modifies the copy! Like this:



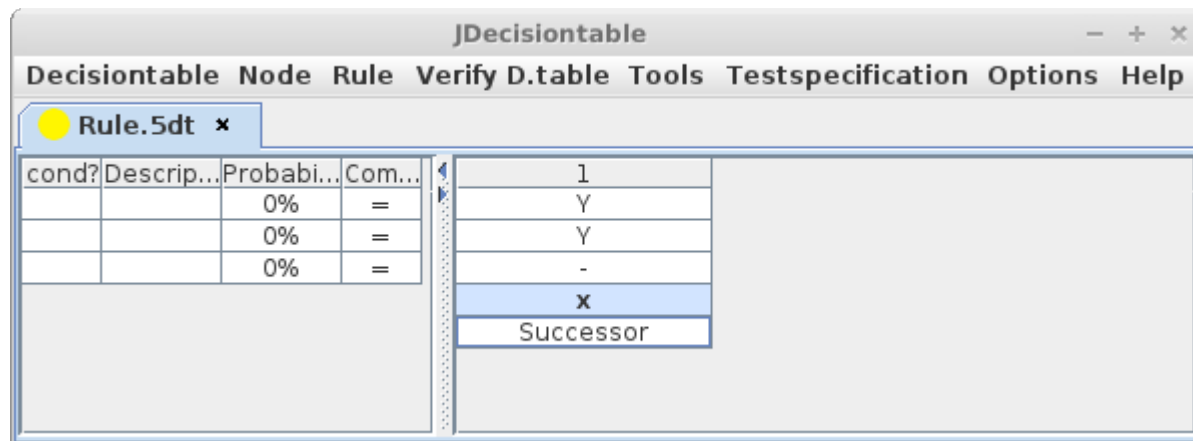
Rule 1 is the source and 2 is the copy.

---

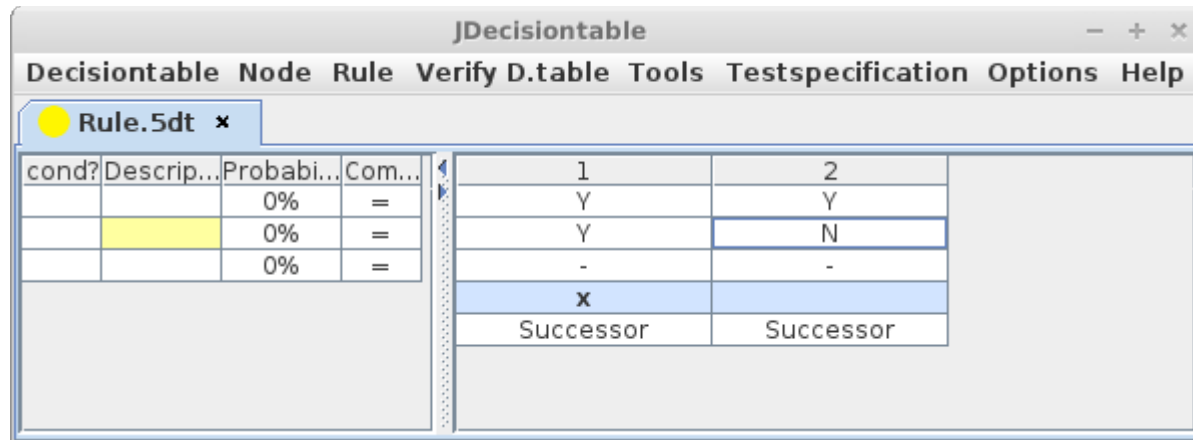
<sup>9</sup> The name “New Rule” is still given with intend because you will use this menu to get a new rule all the time. Probably you will forget what New Empty Rule does because you never use it.



You don't believe that 2 is a modified copy of 1? Ok, lets do again:

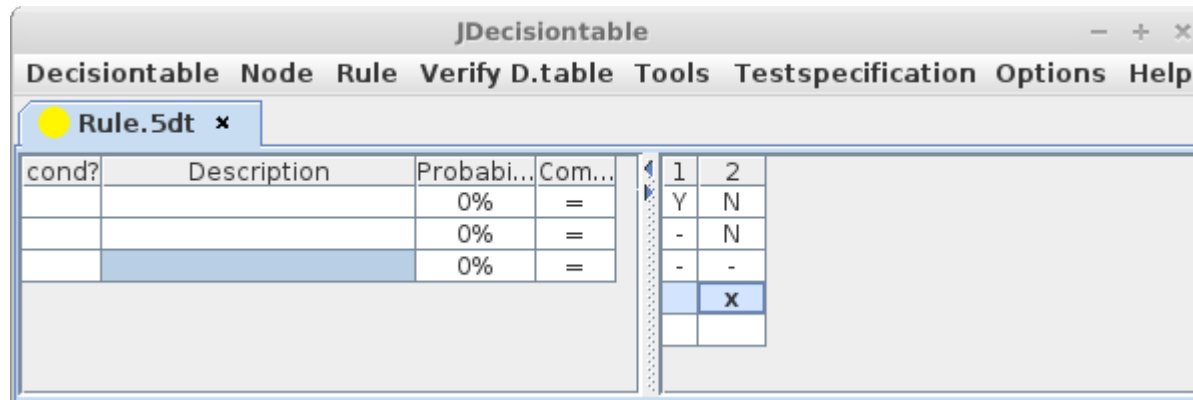


It's really a copy:



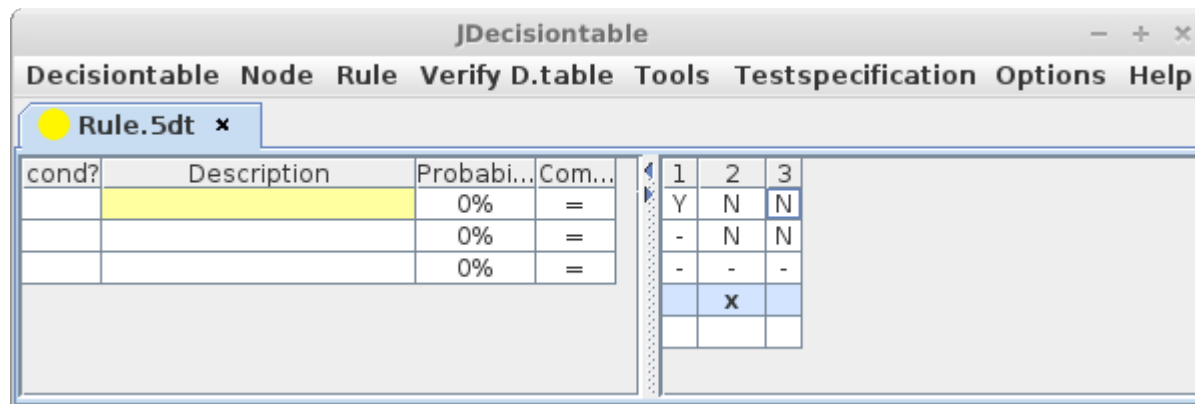
This is the most important feature for creating this kind of decision tables for which JDecisiontable was made for. It copies the selected rule and changes the last Yes-decision into a No-decision. The new rule is always assumed to be invalid. After completing this rule you may mark it as valid.

If there is no Yes-decision in the selected rule it does the same as Rule → Copy Rule<sup>10</sup>. It copies the rule and does not change anything but the isValid mark.



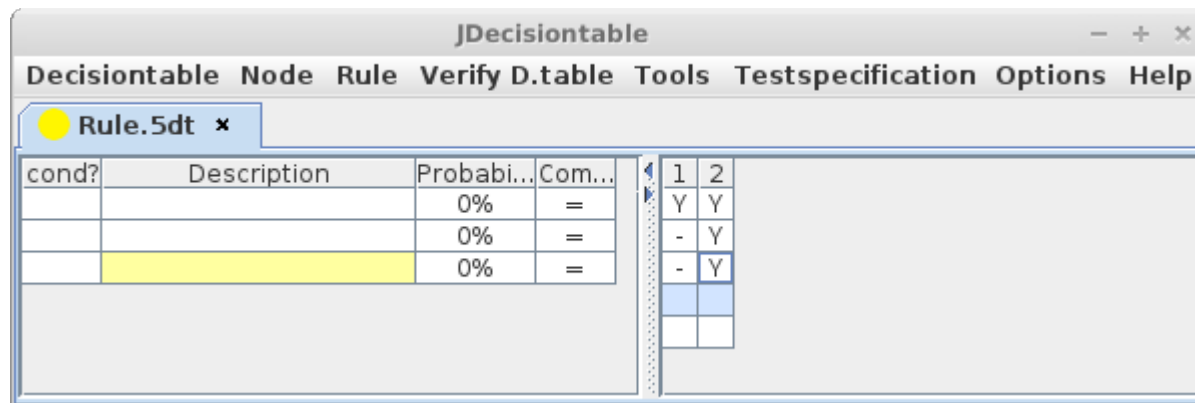
<sup>10</sup> It makes no sense to create a new rule from a rule without Yes-decision since your decision table is complete if you got a rule by Rule → New Rule with No- and Don't Care-decisions only. (Assumed that you did follow the method as described in "How to make decision tables" and did no mistakes.)

Rule 3 is the modified copy of node 2. If a rule is valid the column is automatically wider then the column for invalid rules.



To improve the workflow the changed decision in the new rule will be selected automatically.

The source is always the selected rule:



Now rule 2 was the source and rule 3 is the modified copy. As always, the last Yes decision in the copy was changed into a No decision.

JDecisiontable

Decisiontable Node Rule Verify D.table Tools Testspecification Options Help

● Rule.5dt x

cond?	Descrip...	Probabi...	Com...
		0%	=
		0%	=
		0%	=

1	2	3
Y	Y	Y
-	Y	Y
-	Y	N

You may of course select any rule. The new rule will be inserted right from the selected rule.

**Q:** What happens if there is already a No decision in the rule?

JDecisiontable

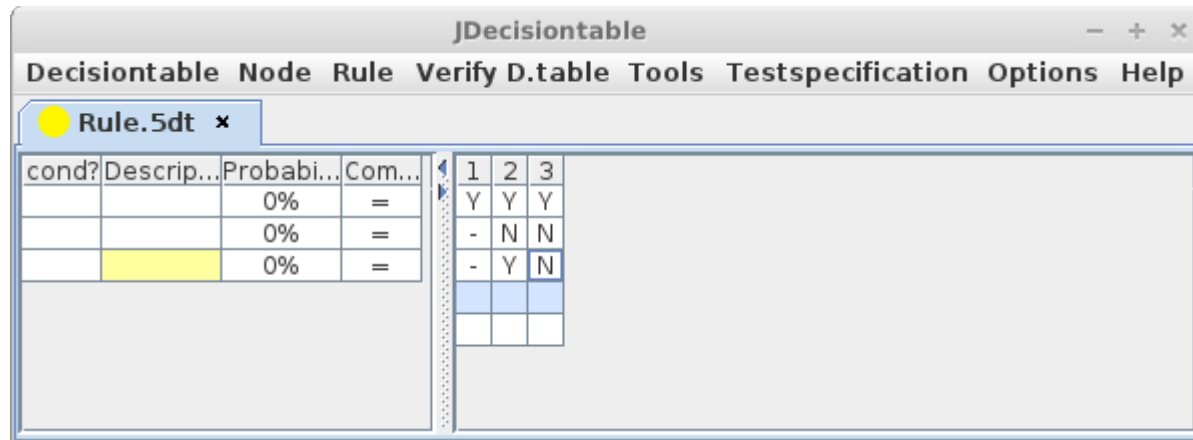
Decisiontable Node Rule Verify D.table Tools Testspecification Options Help

● Rule.5dt x

cond?	Descrip...	Probabi...	Com...
		0%	=
		0%	=
		0%	=

1	2
Y	Y
-	N
-	Y

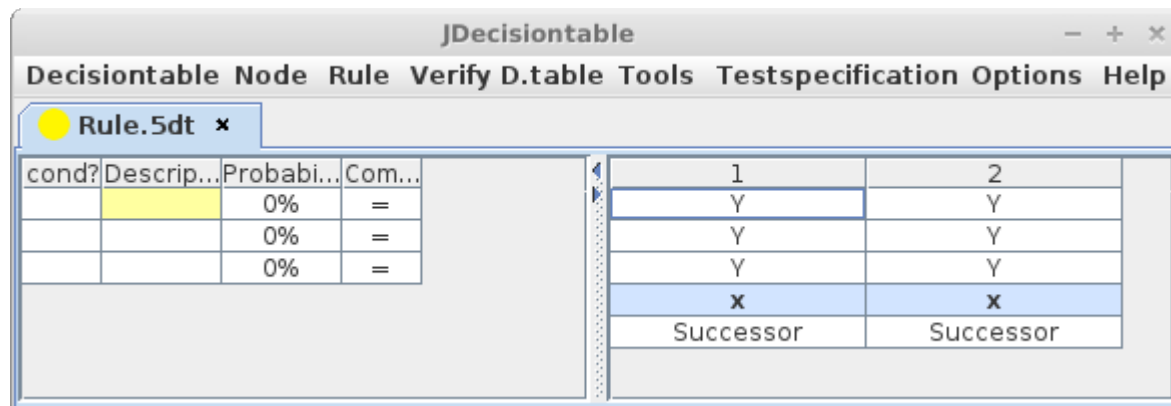
**A:** It still changes the last Yes decision in the copy.



Using this menu is the standard way to get new rules. All other menus are for correcting mistakes or inserting a new node after the decision table was done. But the latter is error prone. The author got better results by copying the old decision table, removing all rules and creating new rules while having a look on the valid rules of the old decision table.

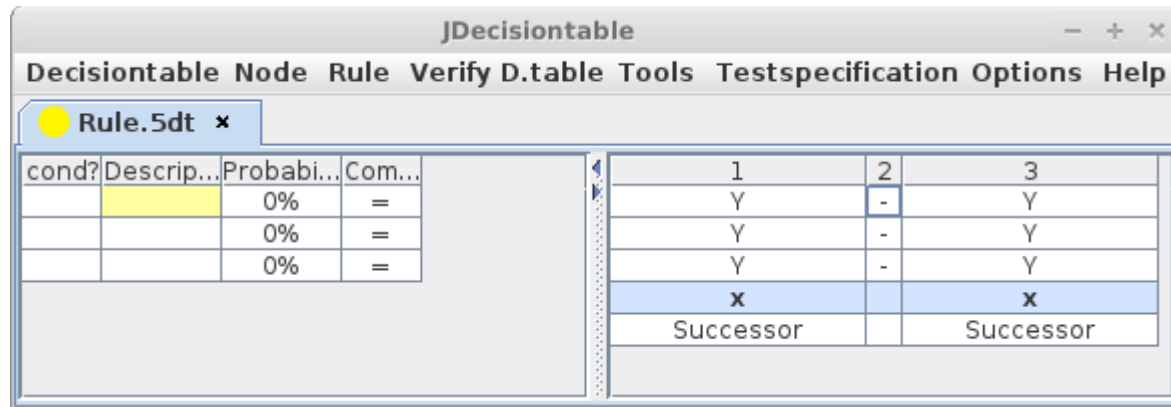
### ***New Empty Rule***

This menu inserts a new created rule right from the selected rule. Other as in Rule → New Rule the selected rule is not copied.



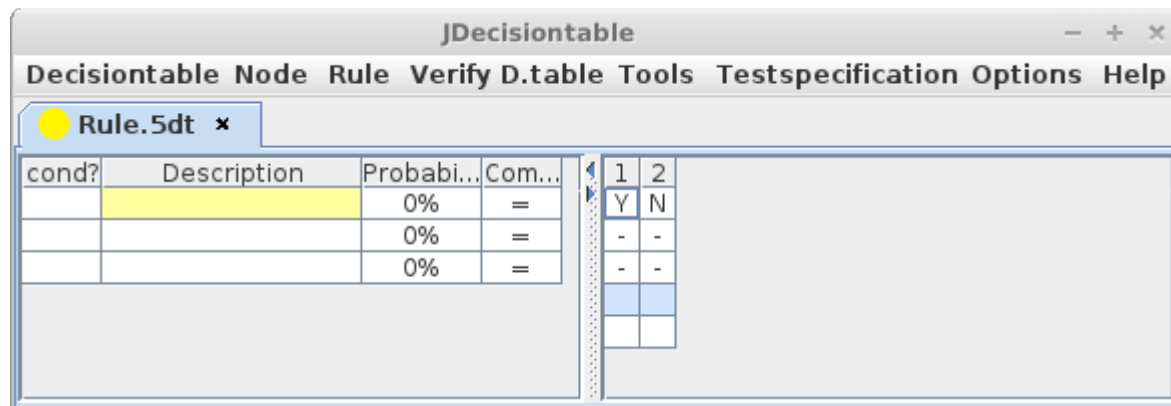
The default value for decisions is Don't Care.



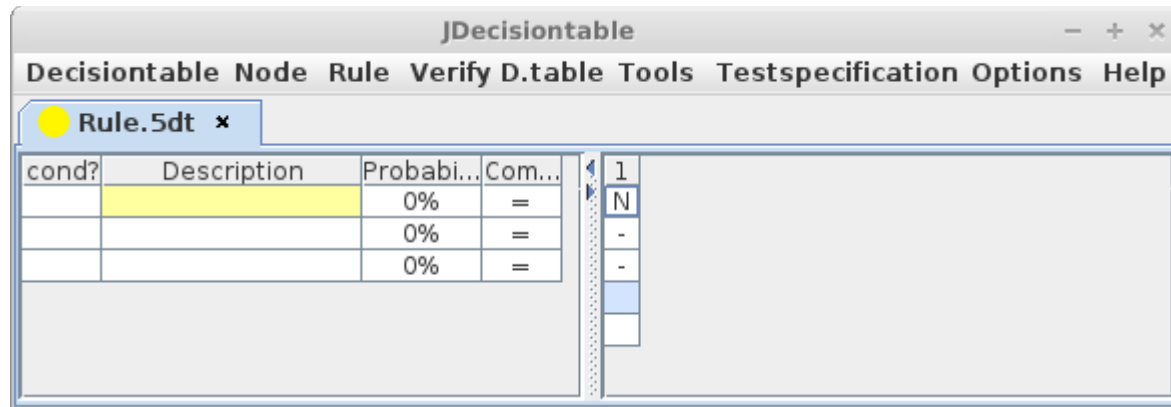


## Remove Rule

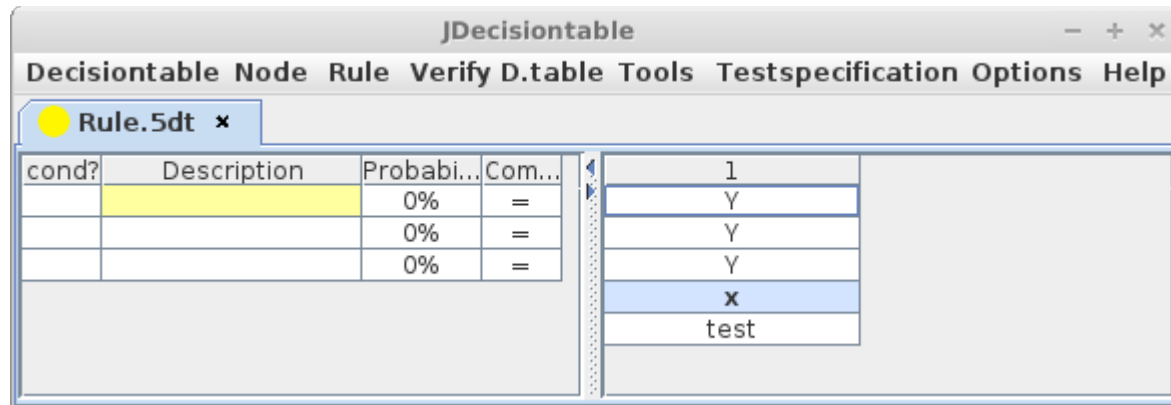
Remove Rule remove the selected rule.



Removed!

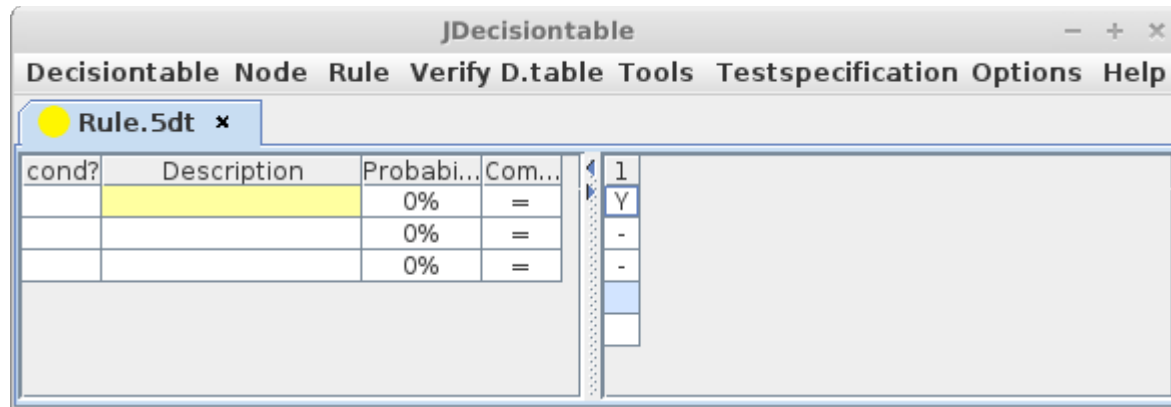


**Q:** What if I remove the last remaining rule from a decision table?



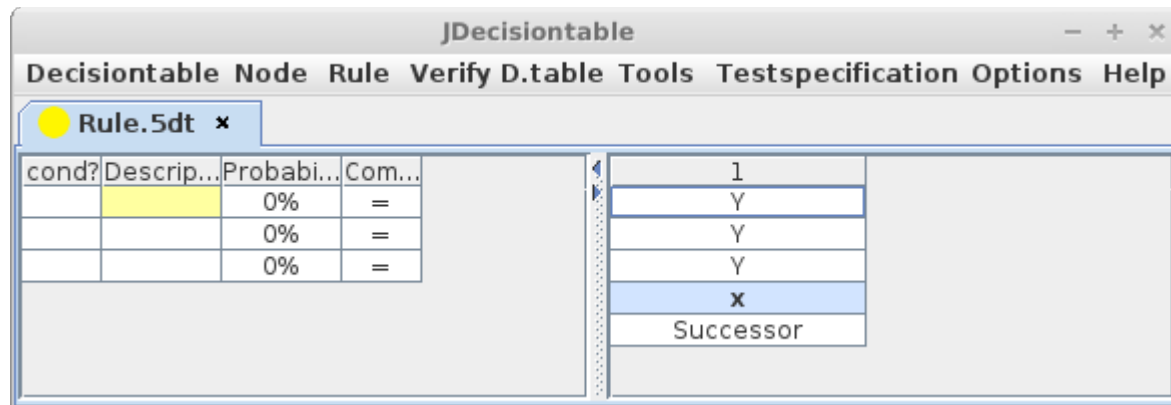
**A:** JDecisiontable will remove this rule but create a new empty rule soon. The it sets a Yes-decision in the first cell<sup>11</sup>.

<sup>11</sup> Since the first rule of a decision table always starts with a Yes-decision.

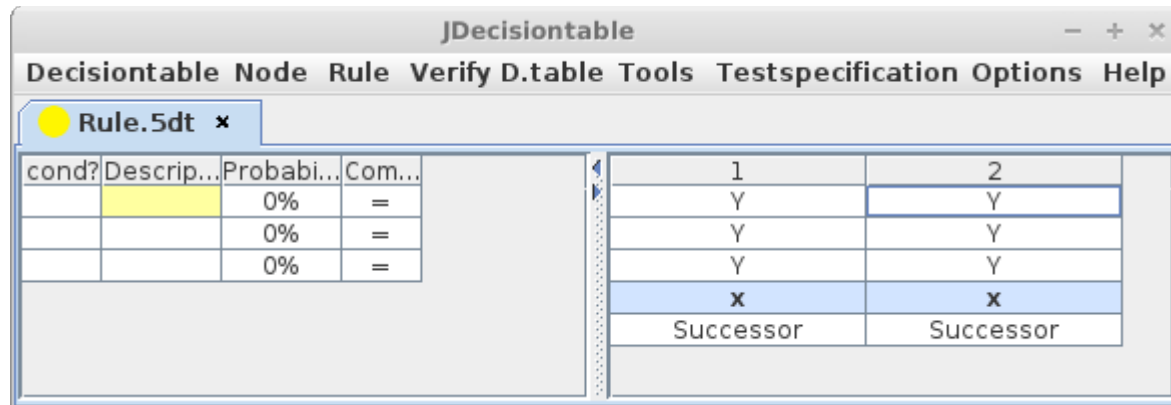


## Copy Rule

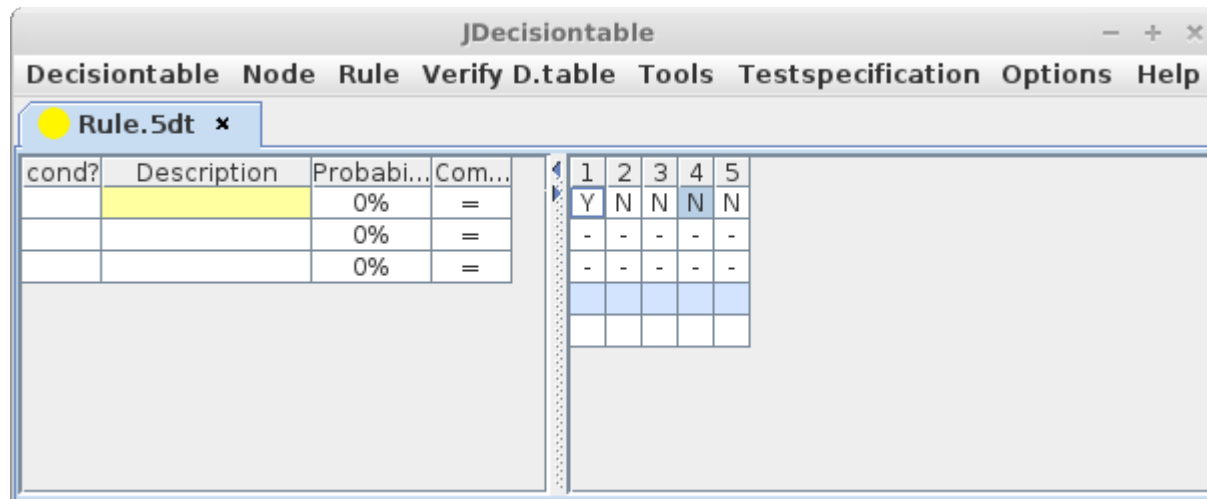
Copy Rule just copies the selected rule.



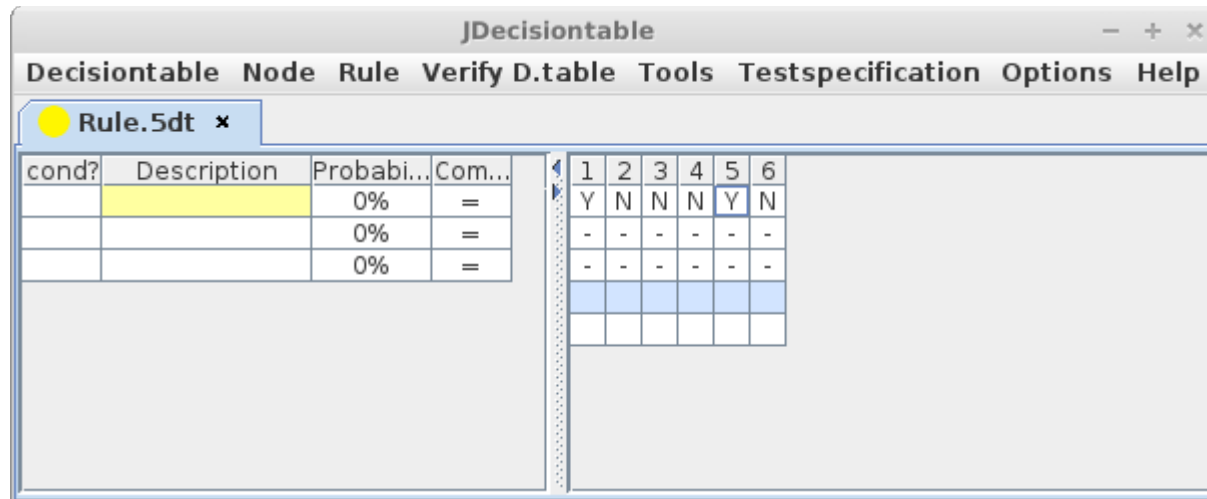
If the source was valid the copy is valid too:



As with nodes you may also define source and destination of the copy by selecting rules which are not neighbors:



The copy of the left hand selected cell (rule 1) will be inserted behind the right hand selected cell (this is rule 4, so the copy will be rule 5).



As with nodes, you may also mark a couple of cells (a range) with same result as selecting single cells would have.

## Menu Verify D.table

This menu provides tools to determine whether a decision table is valid or not. How this is done see “How to make decision tables” among the help files for this application. There are three checks:

No.	Description	Check passed if ...	Menu item <sup>12</sup>
1	Check if there is no node with Don't Care – decisions only	no such node was found	Check Nodes_Dontcare
2	Check if all rules are disjunct	all rules are disjunct	Check Rules_Disjunct
3	Check if expected and actual number of rules are equal	expected and actual number of rules are equal	Check Rules_Number
4	Run check 1..3 in the order 1,2,3	checks 1..3 passed	Run All Checks

To determine whether a decision table is valid or not these three checks are executed in a particular order from No. 1 to No. 3 by menu “Run All Checks” (this is check 4 in the table above).

If “Run All Checks” fails (means the decision table is not valid) you may run each of the checks No. 1..3 individually to find out why. It's very useful to start with check 1, if this check passed run check 2, if this check passed run check 3 since the checks 1..3 are build on top of each other<sup>13</sup>.

When a check is completed a message box tells the result.

---

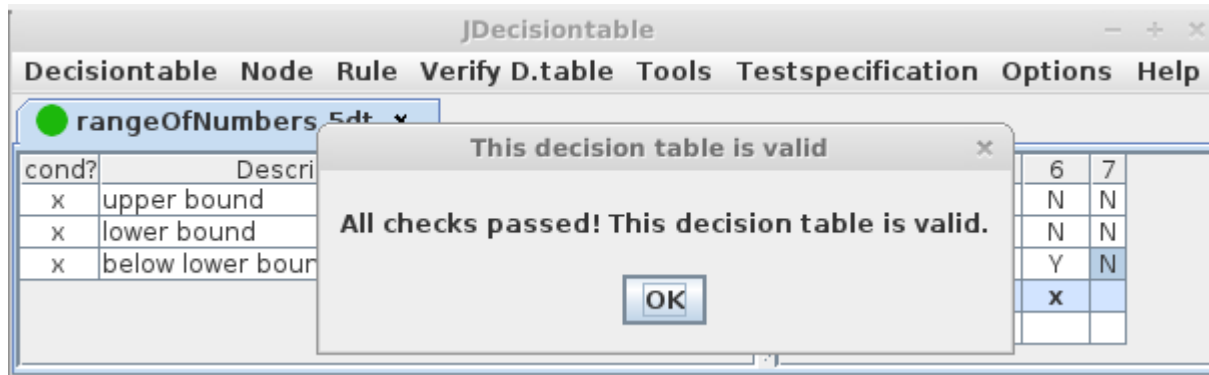
<sup>12</sup> Each check has also a name: 1 = Nodes\_Dontcare, 2 = Rules\_Disjunct, 3 = Rules\_Number

<sup>13</sup> For example it is possible to create a decision table for which check 3 pass but check 1 fail. But if check 1 to 3 runs one after each other and they all passed there is no chance to pass for an invalid table. This is the reason why checks are executed in this order and why there is a single menu point “Run All Checks”. Please note that these checks examine the right hand table (the rules) only! There are still enough options for mistakes on left hand side!

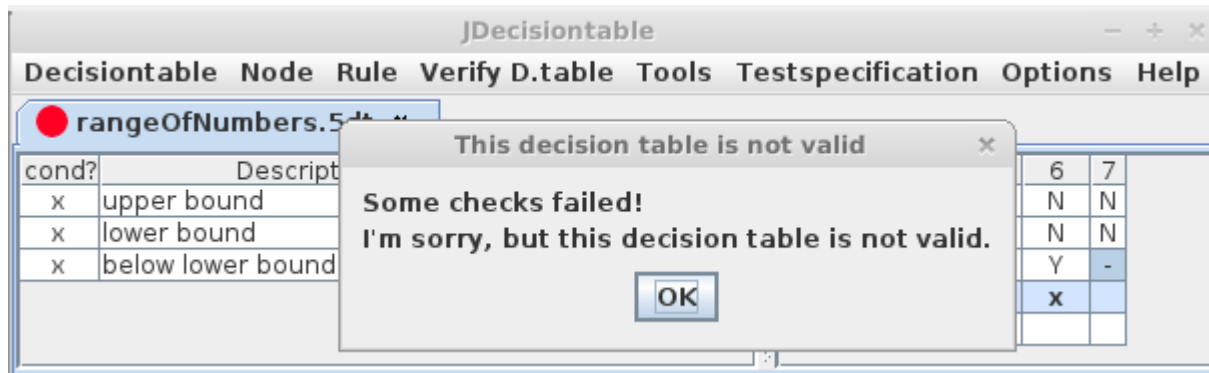
## Run All Checks

This menu runs all three checks 1..3 described above to determine whether a decision table is valid or not. It runs exactly the same checks in same order which are used when exporting a decision table to CSV or getting the test specifications. If one check fails the remaining check will not run.

Example for a valid decision table









Example for an invalid decision table<sup>14</sup>



<sup>14</sup> Why did the check fail? Both pictures show almost same decision table. Please watch the last decision in rule 7. Changing it from No to Don't Care let rule 7 not disjunct from rule 6.

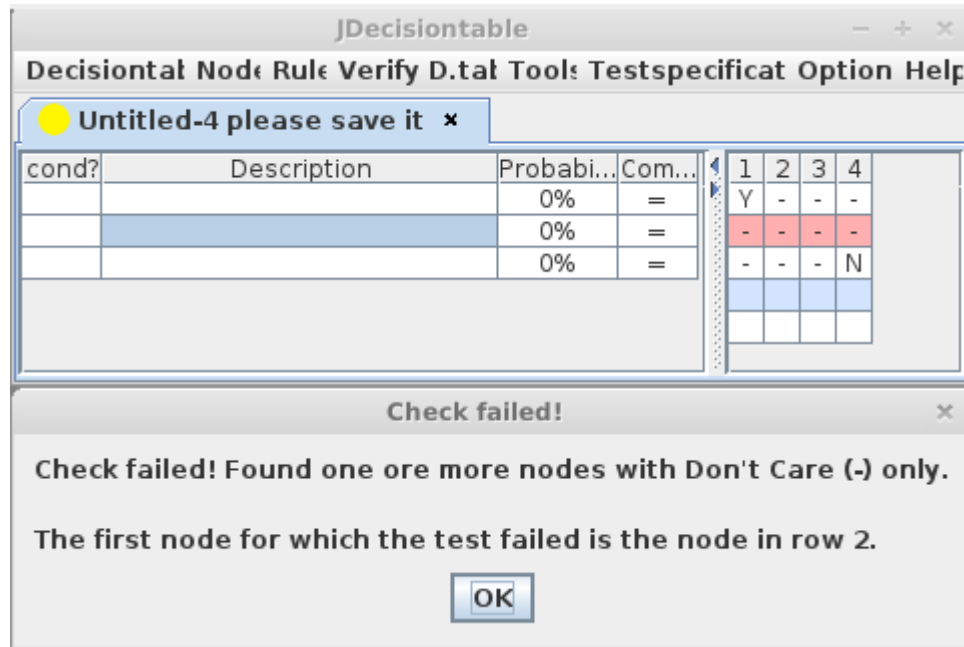
The yellow dot on the tab will turn green if all checks passed and red if not. If you change the decision table this dot will return to yellow. For disabled users there is an alternative icon set using black-on-white-background symbols. Please see chapter “Menu Options” how to switch them on and of. This is the only menu item which changes this icon. Thus if the dot is green it is meant as an optical sign that this decision table is valid.

State	Icon set 1	Icon set 2
Decision table is not valid		
Decision table was changed since last check or was not checked yet		
Decision table is valid		

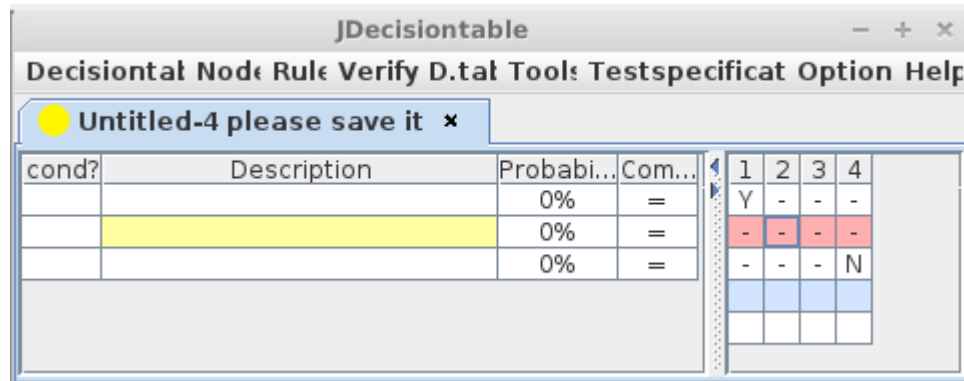
### ***Check Nodes\_Dontcare***

If any node has only Don't Care decisions this check fails and colours this node in red.





You may highlight the description of this node by closing the message box and selecting any cell with red background in the right hand table<sup>15</sup>.



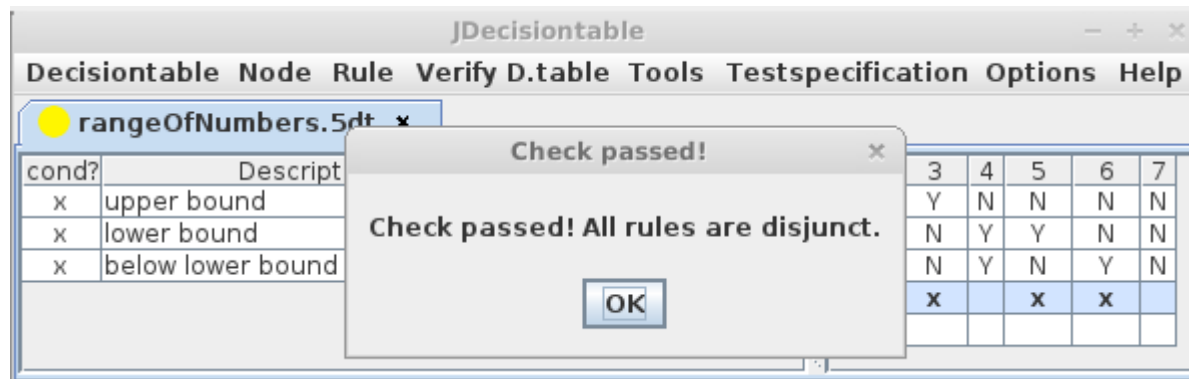
The background colour of the red cells is reset automatically when you run this check again.

<sup>15</sup> You may use Tools → Clear Check Results to reset the background to its default colour.

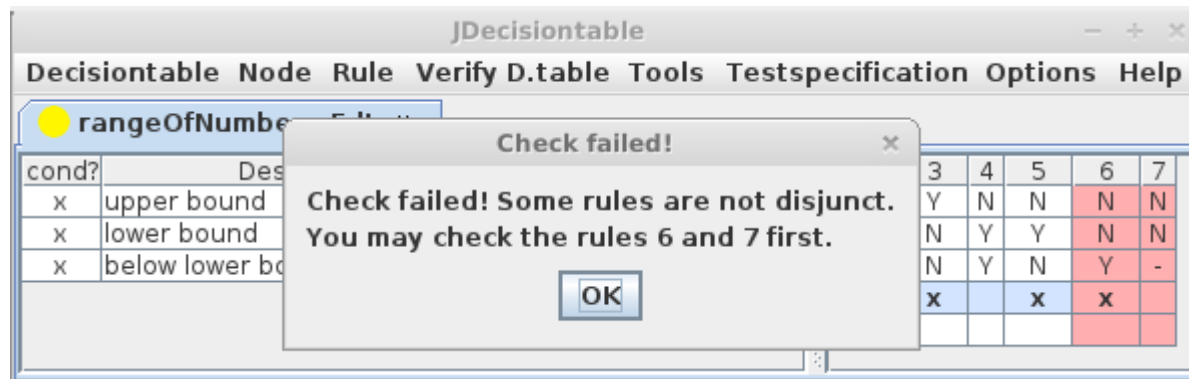
## Check Rules\_Disjunct

This menu item checks if all rules are disjunct. The document “How to make decision tables” among the help files tells in detail how this check is done. In this application the check will stop if two rules which are not disjunct will found. These two rules get a red background colour.

Example with disjunct rules



Example with non-disjunct rules<sup>16</sup>



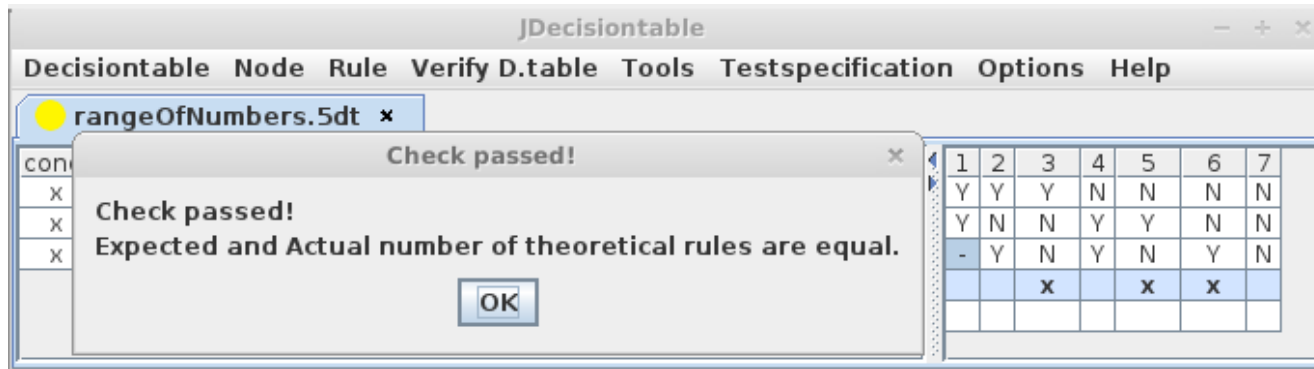
The background colour of the red cells is reset automatically when you run this check again.

<sup>16</sup> Both examples show same file but in the example with non-disjunct rules the last decision in rule 7 was changed from Yes-decision to Don't Care-decision.

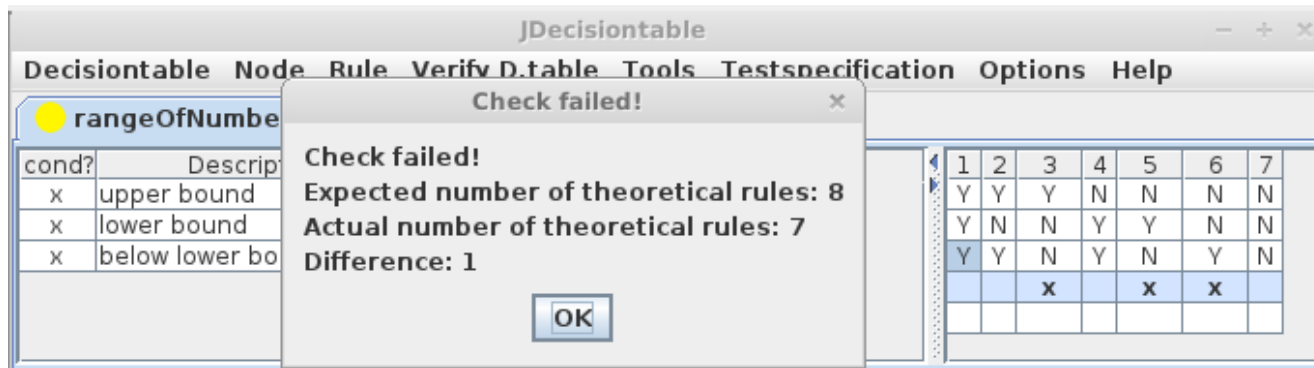
## Check Rules\_Number

This menu compares the expected and the actual number of theoretical rules. “How to make decision tables” among the help files tells in detail how this check is done.

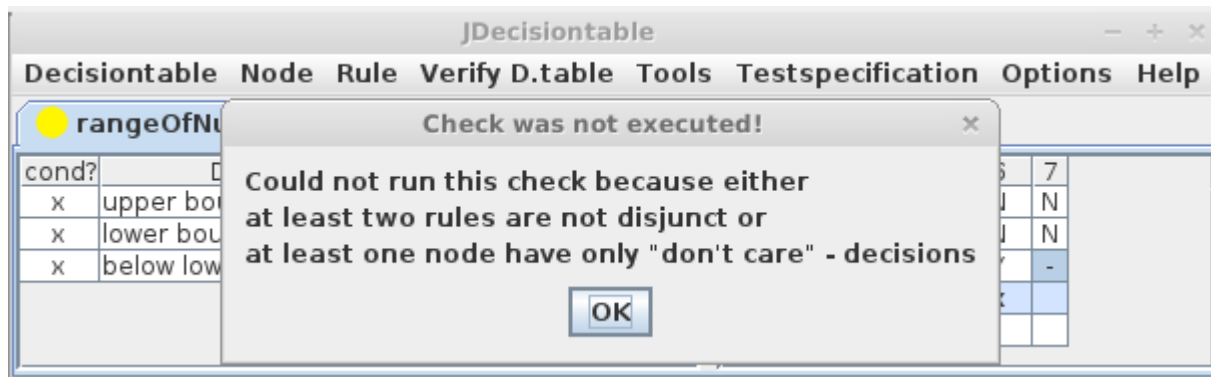
Example with expected and actual number of theoretical rules matching:



Example with expected and actual number of theoretical rules don't matching<sup>17</sup>:



This check runs the checks Nodes\_Dontcare and Rules\_Disjunct before it runs itself. If one of these checks failed this check is not executed. A message box will inform you about what happened.



It is recommended to run the checks Nodes\_Dontcare and Rules\_Disjunct individually to correct the decision table before running this check again. This check will not change the background colour of any cell because this is the job of the particular check.

<sup>17</sup> To make the check fail we changed the last decision in rule 1 from Don't Care to Yes. Since each Don't Care leads to two theoretical rules but Yes or No leads to one theoretical rule (see "How to make decision tables") the expected and actual number of theoretical rules does not match.

### ***Show actual num. of th. rules***

Shall mean “Show actual number of theoretical Rules”. This menu item lets pop up a message box which tells the actual number of theoretical rules. The document “How to make decision tables” among the help files tells in detail how the actual number of theoretical rules is computed.

### ***Show expected num. of th. rules***

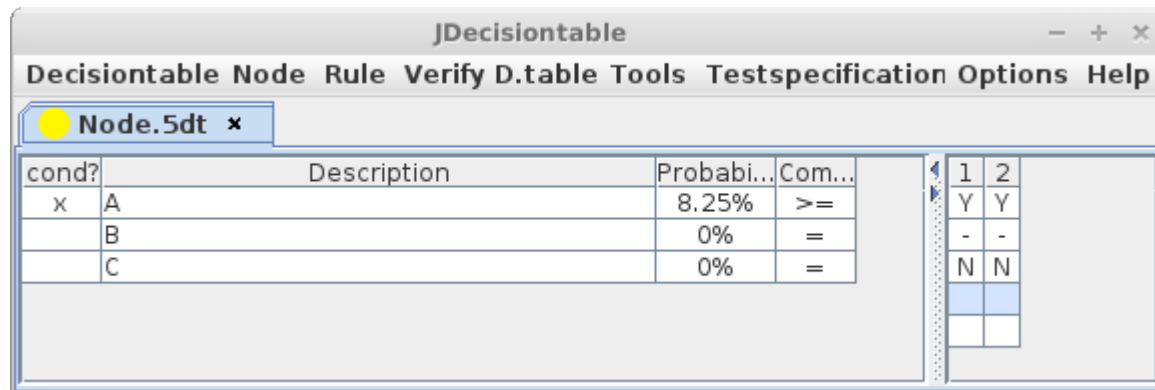
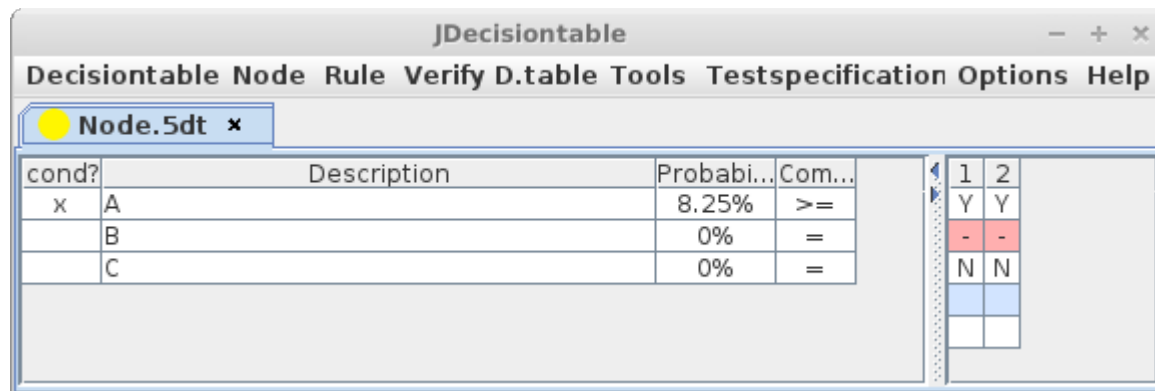
Shall mean “Show expected number of theoretical Rules”. This menu item lets pop up a message box which tells the expected number of theoretical rules. The document “How to make decision tables” among the help files tells in detail how the expected number of theoretical rules is computed.

## Menu Tools

This menu contains some items which provide some interesting functions. Some of these functions are additional checks of the decision table. Although these checks are useful to detect some “smells”<sup>18</sup> or possible improvements.

### ***Clear Check Results***

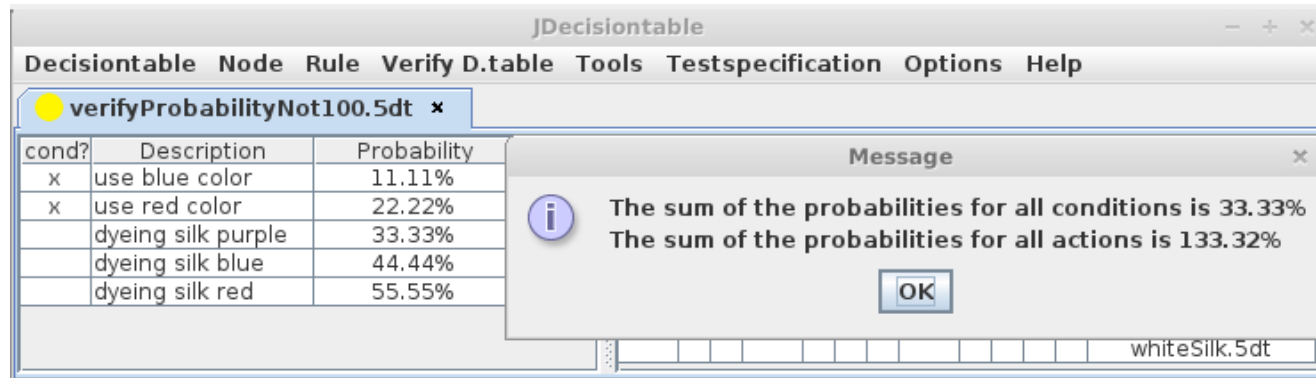
If any check changes the background colour of some cells to mark them, this menu item let you reset the background colour.



<sup>18</sup> Things that are odd but not wrong. “Smells” may cover deeper problems like misunderstandings or improper design. The term “code smell” is common in software development and means a code which works under current circumstances but may break the system under slightly different circumstances.

## Sum Probabilities

As promised this menu item will sum up the probabilities of all conditions and of all actions separately. In detail:



### Conditions:

- use blue color => 11.11 %
- use red color => 22.22 %

Sum: 33.33 %

### Actions:

- dyeing silk purple => 33.33 %
- dyeing silk blue => 44.44 %
- dyeing silk red => 55.55 %

Sum: 133.32 %

This feature may be used for

- usually the sum of probabilities of nodes should sum up each to 100 % and sum of probabilities of rules should sum up to 100 %, so you may use this instrument to check the nodes => the left hand side of the decision table<sup>19</sup>  
(the validation checks in menu "Verify D.table" check the rules => the right hand side of the decision table only)

<sup>19</sup> There is also an entry in the project blog <http://sourceforge.net/p/jdecisiontable/blog/2012/10/use-of-probability-in-decision-tables/> about probability in decision tables. This point tells briefly what this entry tells in detail.

- determining the test cases (each valid rule is a test case) which to run first along the nodes / rules which probability is closest to 100 %
- documenting of business processes in a company (probability of a node: Which condition appears how often?; probability of a rule: Which process runs how often?)
- scientific and market research (e.g. probability of purchases) purposes

### **Show Nodes without Y in Valid Rule**

Shall mean “Show Nodes without Yes-decision in at least one Valid Rule”. This check helps to detect a possible serious problem which the validation checks in menu “Verify D.table” are unable to detect: A node has no Yes in any valid rule. Such a decision table can be valid of course<sup>20</sup> and it may fit for a particular purpose. But

- Why keep a node which is always ignored or denied?
- What is the use of a condition which never comes true? Or of an action which is never executed?

These questions may help to improve your decision table. If this check detects a node without Yes-decision in at least one valid rule you may

- set a rule to valid
- add a valid rule
- remove the node
- if you find a reason to do so leave the decision table as it is; it may be still a valid decision table

Example: Except node 1, no node has a Yes-decision in a valid rule

cond?	Description	Probabi...	Com...
	A	0%	=
	B	0%	=
	C	0%	=
	D	0%	=

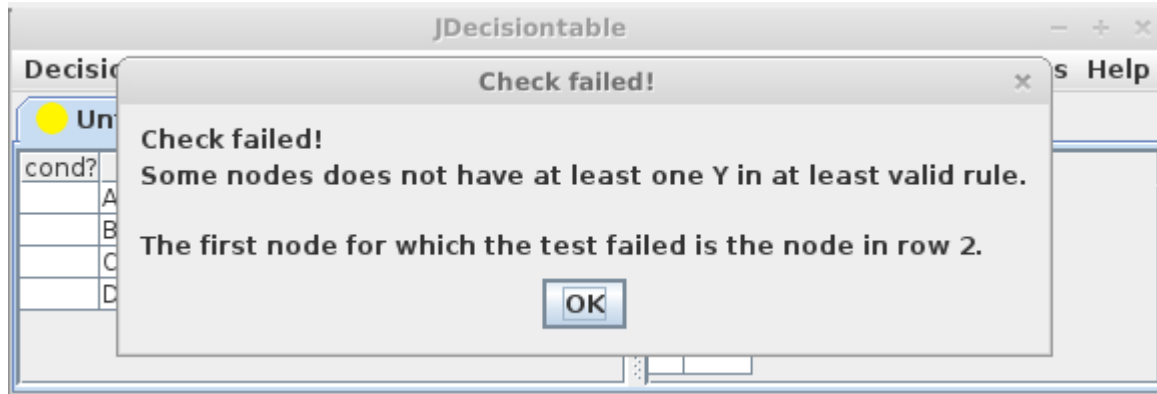
  

1	2
-	Y
Y	-
N	-
-	-
	x

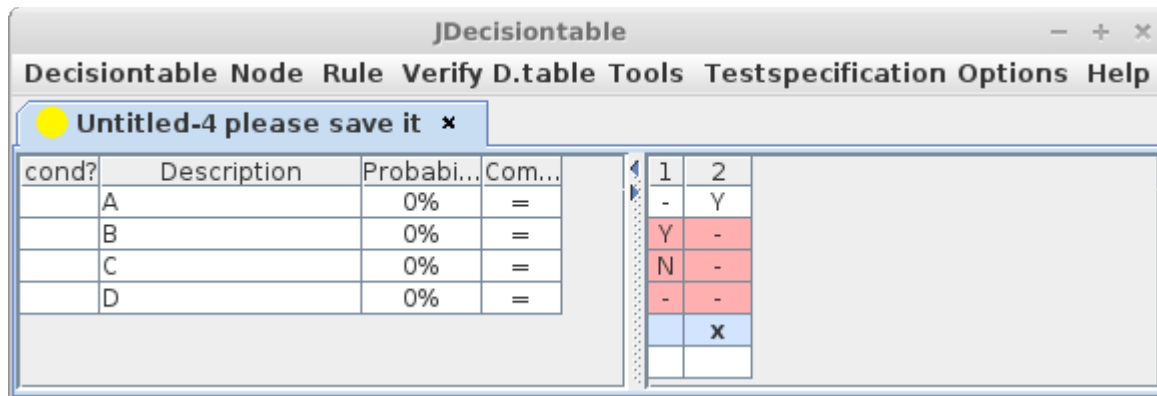
<sup>20</sup> Because validity is determined by definition by a couple of mathematical checks. Validity is not about the if a decision table makes sense.



The message box gives you a hint there to look first.



All nodes found without Yes-decision in at least one valid rule will be shown with red background colour



The menu item Tools → Clear Check Results resets the red background colour.

## Undo/Redo

Since JDecisiontable 2.3.0 there is a undo/redo feature. You may revert any action by choosing **Tools** → **Undo** and repeat the reverted actions by choosing **Tools** → **Redo**.

## Screenshot

With this menu you can make a screen-shot of the program window. The screen-shot will not include the titlebar of the program window.

## Menu TestSpecification

This menu let you get test specifications<sup>21</sup> from decision tables. Test specifications can exported only, there is no way back into the application.

There are two different formats available:

- JSON for using the test specifications in other software
- CSV for using the test specifications in spreadsheets and documents

Furthermore, getting test specifications may or may not include two functions:

- check if the decision table is valid
- split rules<sup>22</sup>

The table below shows the eight possible combinations of these four activities.

	1	2	3	4	5	6	7	8
test specification as JSON	Y	Y	Y	Y	N	N	N	N
test specification as CSV	N	N	N	N	Y	Y	Y	Y
check if decision table is valid	Y	Y	N	N	Y	Y	N	N
split rules	Y	N	Y	N	Y	N	Y	N

Only four activities are available (<number> = <menu item>):

- for JSON the two with green background:
  - 1 = Export to JSON +check +split
  - 2 = Export to JSON +check -split

---

21 A test specification is a valid rule in connection with nodes. Please see “How to make decision tables” for details.

22 Rules which have a Yes-decision for a node with comparison “>=” or “<=” contain two test specifications: One for “=” and one for either “>” or “<”. For a fast overview over a work in progress you do not need to split them but for final test specification this is needed – you would miss a test case otherwise. Please see “How to make decision tables” for details.

- for CSV the two with yellow background:
  - 5 = Export to CSV +check +split
  - 8 = Export to CSV -check -split

As you see JSON files are always checked if the decision table valid.

For CSV files, **Export to CSV +check +split** (no. 5) should always be used to get the final test specification for adding test data and executing tests since it makes sure that the test specifications came from always a valid decision table.

**Export to CSV -check -split** is meant for having a quick self-check while creating a decision table. If the test specification does not show what you intend please check the decision table first. This technique enables you to find mistakes before someone else does.

Hint: If you toggle the “Use English” option the CSV reports will be formatted for the English locale.

## Menu Options

### Make Rows Higher

Make Rows Higher increases the row height like this

[illegible]

## *Use Icon Set 2*

Use Icon Set 2 lets you use an alternative icon set instead the red/yellow/green dot top left on each tab. See chapter “Menu Verify D.table” → “Run All Checks” for how this icons looks like and what they mean.

## *Use English*

Use English switches all menus, messages and button labels to the default language (English). You need to select the option and to close and launch the program to bring it to effect.

For what is it good for? This is meant for better collaboration. Imagine another team member calls you. English so at least one's system languages isn't you common language but one/both desktops aren't localised to English. Instead of holding the phone while trying to switch the system language just tell JDecisiontable to “Use English” and share a view to the application via remote desktop connection! It's also useful if you want to do some documentation or blogging for international audit.

It also does influence how decimal numbers appear in csv files. They will be the formatted for English locale now regardless of the locale settings of you system. Use: This option is meant to communicate with someone else who has different language settings (and thus, localisation settings too). He will stuck if you send him files with a different localisation.

## **Menu Help**

### ***About, Help and License***

These let pop up a dialog windows which tells about author, copyright (where to find the original sources) and a few quick hints.

## **Command Line**

You may open one or more decision tables at launch if you provide them as arguments like

```
JDecisiontable dyeingSilk.5dt getTestSpecifications.5dt
```

All arguments will taken as file names. Arguments known in versions prior to 1.1.0 were removed in favour of an option file.

## Option File

JDecisiontable remembers its settings you made in menu Option. This is done by Java's standard facility for storing options.

The first time when you set an option and close JDecisiontable it will create some folders and files in the home folder of current user. While this place depends on operation systems we will tell it below. The folders and files look like `de/mgmechanics/jdecisiontable/`.

There is a file `prefs.xml` in each folder. The file used to store the settings for JDecisiontable is `de/mgmechanics/jdecisiontable/prefs.xml`. The other files are unused.

## Linux

The file used to store the settings for JDecisiontable is `~/ .java/.userPrefs/de/mgmechanics/jdecisiontable/prefs.xml`<sup>23</sup>

There are also `prefs.xml` files in the folders `de` and `mgmechanics` but these are not used by JDecisiontable. JDecisiontable is not responsible for it.

JDecisiontable always writes the actual settings when the application is quitted. If this file and folders does not exist they will be created first time when you set an option and close JDecisiontable.

## Windows®

Under Windows®XP® registry keys were created below the registry path:

`HKEY_CURRENT_USER/Software/JavaSoft/Prefs/de/mgmechanics/jdecisiontable`

## File Formats

First, this application uses **JSON** (a data format) to read and write decision tables and to write test specifications. JSON<sup>24</sup> is a widely used human-readable<sup>25</sup> data format available for many programming languages. It is a text format stored in text files. This should make it easy to use, alter

---

<sup>23</sup> “~” is an usual place holder meaning the home folder of the current user e.g. `/home/jane/`

<sup>24</sup> “JSON” means JavaScript Object Notation. It was originally developed for Javascript and is still very common here. This was one reason to choose it – it integrates well with internet technology.

<sup>25</sup> There is the plug-in “JSONView” for the web browser Mozilla Firefox <http://jsonview.com>. Just install the plug-in, rename any `*.5dt` or `*.5ts` file to `*.json` and open it with Firefox.

or create decision tables and test specifications with other applications and scripts.

### **Writing JSON text files:**

All JSON text files which JDecisiontable writes are encoded as **UTF-8** without adding a BOM<sup>26</sup>. According to [RFC3629](https://tools.ietf.org/html/rfc3629) it is possible to save Chinese, Japanese and Korean characters in UTF-8 encoded strings.

### **Reading JSON text files:**

All JSON text files which JDecisiontable reads are expected to be encoded as **UTF-8**. It does not matter if they have a BOM. Successful tests were made to read UTF-8 encoded files having a BOM<sup>27</sup> and having no BOM with JDecisiontable version 1.1.1 and at every build of later versions. These tests use German umlauts and special chars to prove JDecisiontable.

Furthermore, this application may write decision tables and test specifications as **CSV** files. These files are encoded as **UTF-16LE with BOM**. The field delimiter (= the char between two columns) is a tab stop (= ASCII 09). The record delimiter (= the char or chars between two rows) depends on the defaults of your operation system: CR LF (= carriage return + line feed = ASCII 13 10) for Microsoft® Windows® and LF (= ASCII 10) for the rest of us (Linux®, Unix and so on). This configuration was successfully tested on a system running Microsoft® Windows® 7. I just needed to double-click on a csv file and it was opened in Microsoft® Excel® 2010. I was able to open these files with LibreOffice® 3.6.2.2 the same way. If nothing else helps there is a trick: Open the CSV file with a text editor and copy-and-paste all content right into an empty spreadsheet.

Last but not least this application can read a list of conditions from a text file. This file is expected to be encoded in UTF-8 with or without BOM as JSON text files.

---

26 A BOM (= Byte Order Mark) is some extra information at the beginning of a UTF-x encoded string. You can see it as bytes EF BB BF before the very first char of text in a hex editor. JDecisiontable does not add a BOM at the beginning of UTF-8 encoded files. It relies on Java which does not add it.

27 The test uses Windows® "Editor" editor (notepad.exe) to open a \*.5dt file containing characters "ABCxyz123ÄÖÜäöüß@€|μ~<sup>123</sup>¼½↩{[]}\". and saved as text file with encoding "UTF-8". "Editor" added a BOM visible as byte EF BB BF at the beginning of the file in a hex editor (Gnome ghex). By the way: The encoding named "ANSI" by Windows® "Editor" is known as "windows-1252" according to IANA: <http://www.iana.org/assignments/character-sets/character-sets.xml>.

Name	Read/ write	Text/ binary	Description	Suffix
Decision table JSON	read + write	Text	Used to save and read decision tables. This files contain objects of class de.mgmechanics.decisiontablelib.Decisiontable.	5dt
Test specification JSON	write only	Text	Used to store test specifications to open them with other applications. This files contain objects of class de.mgmechanics.decisiontablelib.Testspecification. You can not open these files with JDecisiontable.	5ts
Csv	write only	Text	Used to export decision tables and test specifications suitable to open them as a spreadsheet. Cells are delimited by tab characters (“\t”, hexadecimal 0x09, ASCII code: 09). It was tested with LibreOffice Calc. You can not open these files with this application. The tab character is used as delimiter between cells with intend because you may open any of such csv files in a text editor, copy the content and paste it right into Libre Office Calc or Microsoft® Excel® <sup>28</sup> .	csv
Text	read only	Text	Used to create a decision table using it's content.	txt
Localisation	read only	Text	Used to keep Strings for menus, messages, button labels and even contents for the tables.	json

---

<sup>28</sup> Furthermore, in some European countries the comma is used as decimal separator instead of a period. This might the reason why Microsoft® Excel® uses the colon (;) in the German language version and the comma in the English language version. Using tab stop character as delimiter avoids confusion.

## For translators

JDecisiontable doesn't use the properties files commonly used for Java software. It comes with a unique facility which uses a JSON string stored in UTF-8 encoded text files named **StringResource[\_xx][\_YY].json**. These are stored in the jar file in the folder **de/mgmechanics/jdecisiontable/**.

After reading the values for language and country from your system e.g. "de" for language (always lower case) and "DE" for country (always upper case) it uses three steps to get the strings:

1. It reads all values from **StringResource.json**. There are no strings stored somewhere in the source code except program name and version number. Thus, this file keeps all keys which the software uses. The language used in this file is called "default language" and should be English. If the option "Use English" is on the string values from this will be used.
2. It looks for a file named **StringResource\_<language>.json** e.g. StringResource\_de.json. If found it overrides the values from step before. This file does not need to contain all keys which StringResource.json has. If a key is missing here it takes the value from StringResource.json.
3. At last looks for a file named **StringResource\_<language>\_<country>.json** e.g. StringResource\_de\_DE.json. If found it overrides the values from step before. If a key is missing here it takes the value from StringResource\_<language>.json or - if the key is missing in this file also - from StringResource.json.

In each step it looks first in the same folder as the jar file is. If the resource file is not there then it looks inside the jar file. You may place your own resource file in the folder where the jar file is and JDecisiontable will use this file. This makes it easier to make translation because you can start with a single **key : value** pair and can add pair for pair subsequently.

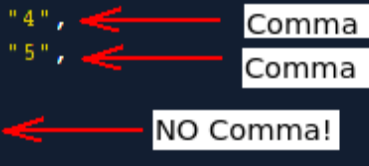
Each JSON contains merely a Map object (a list of key and values, sometimes called "dictionary") like:

```
{  
"msgRunAllChecksItem0" : "This decision table is valid",  
"msgRunAllChecksItem1" : "All checks passed! This decision table is valid."  
}
```



REALLY IMPORTANT: There must be no comma after last entry / before the “}”!

```
"tableStringToSetComparisonGT" : "4",  
"tableStringToSetComparisonLT" : "5",  
"tableDisplayTextSuccessor" : ""  
}
```



The diagram shows three lines of JSON code. Red arrows point from the labels 'Comma' to the commas at the end of the first two lines. Another red arrow points from the label 'NO Comma!' to the space before the closing brace of the third line.

Its build as “key” : “value”.

The first string is the key which the software uses to find the value. Please do not alter!

After the “:” follows the value: This is the text displayed and in some cases the key to type causing to change the value of a cell. This is to replace with the translated string.

For example, "msgRunAllChecksItem1" : "This decision table is valid"

"msgRunAllChecksItem1" => key

"This decision table is valid" => value

If you need a linefeed just place “\n” where you need it. “{0}”, “{1}”, {2} and so on are replaced with strings generated at runtime. You may display the message for same key for English to see what it is replaced with. You may consider using option “Use English” to get all menus and messages displayed in English regardless your systems language.

Which keys are used to set non-editable cells is hard-coded yet because JDecisiontable for Microsoft® .NET® would not work otherwise. Thus these keys are same for all languages. “Keys” means that we really evaluate which key was pressed rather than asking for the letter which was typed.

For the unexpected case that someone should need to change the chars displayed in non-editable cells they are also mapped in the resource files. Currently they are only in the StringResource.json file but you **could** change them by copying theses keys to your resource file and changing the values:

#### Column “Cond?” in left hand table

```
"tableDisplayFlagIsConditionTrue" : "x",  
"tableDisplayFlagIsConditionFalse" : "",
```

#### Decisions in right hand table

```
"tableDisplayDecisionYes" : "Y",  
"tableDisplayDecisionNo" : "N",  
"tableDisplayDecisionDontCare" : "- ",
```

#### “isValid”-flag for each rule (also in right hand table)

```
"tableDisplayFlagRuleIsValidTrue" : "x",  
"tableDisplayFlagRuleIsValidFalse" : "",
```

#### Column “Comparison” in left hand table

These characters are hard-coded because mathematical symbols should be the same all over the world.

Same thing for the csv reports. For decision tables the keys are as before but starting with “dtReport”. For test specifications they are starting with “tsReport”. Additionally there are:

#### default values for the field “Successor”

```
"dtReportTextSuccessor" : "",  
"tsReportTextSuccessor" : "",
```

#### default values for the fields for test data to be filled in by the user

```
"tsReportTestdataPlaceholder" : ""
```

It is strongly recommended NOT to change these values, please. Do not rely on the undo/redo feature. If you do serious work with your decision tables using version control software is strongly recommended.

## Please do not rely on the undo/redo feature – use version control

Benefit of using a version control software: You can add comment to each version as well as using tags e.g. “these are the decision tables for version x.y of our software”.

Because the decision tables are stored as plain text you may even show the differences between two decision tables using the version control software.

If you can not use a version control system make at least backups of your data before and during work.

The version control software **Mercurial** ([www.mercurial-scm.org](http://www.mercurial-scm.org)) works great for the author (Michael Groß). It doesn't require to set up a server. All it needs is a folder for your data. There is a great GUI for it named **Tortoise Hg** ([tortoisehg.bitbucket.org](http://tortoisehg.bitbucket.org)).