COST-EFFECTIVE DIAGNOSING USING A BAYESIAN SYSTEM MODEL

COMMAND-LINE CHEATSHEET

STARTING THE PROGRAM

Copy the two jar-files 'bayesserver-9.5.jar' (in folder 'lib') and cedubam-xxx.jar' (in folder 'target') and also the folder 'resources' (scripts and systems) to a local folder.
The program can be started in the CLI. Go to the directory of the program. Then type

```
java -cp ./* app.App
```

or, if a valid license key for Bayesserver[1] is available,

```
java -cp ./* app.App license:<bayesserver-licence-key>
```

The program starts and the commands can be typed.

A java version of JRE 1.8 is required.

COMMANDS

The app is driven by a command line interface. A list of commands is described in the remaining. The command is executed by typing the ENTER-key. The command must be typed in lowercases but for the values the case mostly does not matter (except when typing url's). The app cannot change the network's structure. This must be done in the BaysesServer-interface.

NOTATION OF THE COMMANDS

Structure	command1 command2
Meaning	Type the commands separated by a space
Example structure	display variables
Example to type	display variables
Structure	<type:name>
Meaning	A value of type 'type' for name. Mind that some quote-signs are not accepted (accepted is Unicode 0022 and 0027)
Example structure	<string:networkname>
Example to type	"name with spaces" *or* 'name with spaces' *or* namewithoutspaces
Structure	value1|value2|value3
Meaning	Choose one of the values
Example structure	true|false|clear
Example to type	false
Structure	parameter:value
Meaning	Give the parameter a value
Example structure	alpha:<double>
Example to type	alpha:0.01
Structure	[value]
Meaning	Optional argument

[1] The trial version can only be used for two hours and saving changes to the network is not possible.

Example structure	command [variable:<string:name>|<integer:index>]
Example to type	command variable:light1 *or* command variable:2 *or* command
Structure	[value]+
Meaning	Optional argument which can be given 0 or more times, separated by a space
Example structure	command [variable:<string:name>|<integer:index>]+
Example to type	command variable:light1 variable:2 variable:"light two" *or* command variable:light1 *or* command

OVERVIEW OF THE COMMANDS

GENERAL

Command	exit|quit
What it does	Closes the app gracefully
Example	exit
Command	run <string:scriptresource>
What it does	Runs the script
Note	Each command on a new line. # can be used to write comments. Blank lines are allowed.
Example	run resources/scripts/test1 *or* run "D:\Documenten\My Script.txt"
Command	print <string:message>
What it does	Displays the message in the console.
Note	Use quote signs if spaces the message includes blancs. Non visible signs as "\n" or "\t" can be used. See examples in the corresponding section of this document.
Example	print "Message in a Bottle"

NETWORK

Command	display networks
What it does	Displays the available networks and their index
Note	
Example	display networks
Command	set network <string:name>|<integer:index>
What it does	Set the network with the given name as String, the given index as integer or path and filename
Note	must match the information as given by the command 'display networks' or must be a valid local path.
Example	set network WaterPipes *or* set network 0 *or* set network "D:\Documenten\My Network.bayes"
Command	display network
What it does	Displays the current network
Example	display network
Command	display groups
What it does	Displays the groups of the network and their indices
Example	display groups
Command	set healthgroups [<string:groupname>]
What it does	Sets the groups indicating the healthnodes in an 'or' relation (all variables of all given groups are included)
Note	if no groupnames are given, the list of healthgroups is emptied
Example	set healthgroups connection component

Command	display healthgroups
What it does	Display the healthgroups
Example	display healthgroups
Command	display variables
What it does	displays all the variables of the network and their index
Example	display variables
Command	display variable <string:name>|<integer:index>
What it does	display detailed information of the given variable
Note	there is still a problem of displaying the distribution for children
Example	display variable health_plug1 *or* display variable 1
Command	display variables type:rootinput|output|health|leaf
What it does	Displays the variables of the given type
Example	display variables type:health *or* display variables type:output

EVIDENCE

Command	set pdvs [<string:varname>|<integer:varindex>]
What it does	Sets the **p**roblem **d**efining **v**ariable**s**
Note	For now only one variable can be given. No evidence is set on the problem defining variables. if no variables are given, the list pf pdvs is emptied.
Example	set pdvs light1
Command	set pdvs <string:varname>|<integer:varindex> null [active:true|false]
What it does	Sets the pdvs and removes the evidence on it
Note	If active is set to false, the evidence is not activated (default:true).
Example	set pdvs light1 null
Command	set pdvs <string:varname>|<integer:varindex>
What it does	sets the pdvs and sets hard evidence on the given state
Example	set pdvs light1 off *or* set pdvs light1 1
Command	set pdvs <string:varname>|<integer:varindex> <string:state|<integer:index>:<double:value> [<string:state|<integer:index>:<double:value>]+ [active:true|false]
What it does	Sets or replaces the given (soft) evidence on the states.
Note	Make sure all states are mentioned in the command. There is no check on the sum of the probabilities
Example	set pdvs light1 off:0.6 on:0.4
Command	display pdvs
What it does	Displays the pdvs and the evidence set on it
Note	Displays the pdvs and the evidence set on it
Example	display pdvs
Command	set evidence <string:varname>|<integer:varindex> <string:evstate|<integer:evindex|null> [<string:varname>|<integer:varindex> <string:evstate|<integer:evindex>|null]+ [active:true|false]
What it does	Sets or replaces hard evidence on the given state of the given variables. If null is given no new evidence is set.
Note	If active is set to false, the evidence is not activated (default:false)
Example	set evidence battery present light2 on active:true
Command	set evidence <string:varname>|<integer:varindex> <string:state|<integer:index>:<double:value> [<string:state|<integer:index>:<double:value>]+ [set pdvs <string:varname>|<integer:varindex> <string:state|<integer:index>:<double:value> [<string:state|<integer:index>:<double:value>]+] [active:true|false]

What it does	Sets or replaces the given (soft) evidence on the states. the given variables
Note	Make sure all states are mentioned in the command. There is no check on the sum of the probabilities
Example	set evidence battery present:0.6 absent:0.4 light2 on:0.3 off:0.7
Command	remove evidence <string:varname>|<integer:varindex> [<string:varname>|<integer:varindex>]+
What it does	Removes the evidence of the given variables
Example	remove evidence battery light2
Command	display evidence
What it does	Display the current evidences
Example	display evidence

DIAGNOSTIC SETTINGS

Command	set diagnoses
What it does	Generates the possible diagnoses
Note	This command must be called to be sure the diagnoses are computed.
Example	set diagnoses
Command	display diagnoses
What it does	Displays the possible diagnoses
Example	display diagnoses
Command	display informationfunctions
What it does	Displays the available information functions and their index
Example	display informationfunctions
Command	display informationfunction
What it does	Displays the current information function
Example	display informationfunction
Command	set informationfunction <string:name>|<integer:index>
What it does	Sets or replaces the information function
Note	The names and indices are shown by the command 'display informationfunctions'
Example	set informationfunction entropy_reversed *or* set informationfunction 1
Command	display utilityfunctions
What it does	Displays the available utility functions
Example	display utilityfunctions
Command	display utilityfunction
What it does	Displays the current utility function
Example	display utilityfunction
Command	set utilityfunction <string:name>|<integer:index> [c:<double:value>]|[alpha: <double:value>]|[a: <double:value>] [s: <double:value>]
What it does	Sets or replaces the utility function
Note	Mind that the constants must correspond with the given utility function. If the constant is not given, the default value is attributed. The names and indices are shown by the command 'display utilityfunctions'
Example	set utilityfunction weighted_cost alpha:0.21
Command	display strategies
What it does	Displays the available strategies
Example	display strategies
Command	display strategy
What it does	Displays the current strategy

Example	display strategy
Command	set strategy <string:name>|<integer:index>
What it does	sets or replaces the strategy
Note	The names and indices are shown by the command 'display strategies'
Example	set strategy meu

PROBES

Command	display defaultcost
What it does	Displays the defaultcost of the probes
Example	display defaultcost
Command	set defaultcost <double:cost>
What it does	Sets the defaultcost to the given value
Example	set defaultcost 25.5
Command	set probes [cost:<double:cost>]
What it does	Generates all the probes automatically. If a cost is given, all probes have this cost.
Example	set probes cost:50
Note	This command must be called to be sure the probes are computed.
Command	display probes
What it does	Displays the probes and their cost
Example	display probes
Command	display probe <string:varname>|<integer:varindex>
What it does	displays the probe on the given variable
Example	display probe trigger1
Command	set probe <string:varname>|<integer:varindex> [cost:<double:cost>] [enabled:true|false] [<string:varname>|<integer:varindex> [cost:<double:cost>] [enabled:true|false]]+
What it does	Sets or modifies a probe for the given variable(s) with the given cost. If enabled is set to false, the probe is not taken into account. Default:true
Example	set probe battery cost:40 plug2 cost:80 light2 cost:50
Command	remove probe
What it does	Removes all the probes or the given ones
Example	remove probe trigger2 light2 *or* remove all

REPORTS

Command	set report [display:false|true] [displaydetail:int] [export:true|false] [exportdetail:int] [path:folderURL] [filename:string] [csvname:name] [suffix:count|time|none] [counter:int]
What it does	Sets the settings for reporting. Reports are displayed in the console or not (display) and can be exported to a file (.txt) (export) and a cvs file. The detail of the report is implemented as an integer. For now this van be an integer from0 (least details) to 3 (most details). The path can be relative or absolute. Be aware that files are created, but folders must exist. A suffix can be added to the filename (to avoid files to be overwritten). The suffix can be a counter (starting in this session of the app with 'counter') or a timestamp (then every file is surely a new one).
Note	This is only tested on my Windows pc. Mind that files can be overwritten if values of filename or suffix are given. The csv file always adds the new information. There are default values for all parameters.
Example	set report display:true displaydetail:0 export:true exportdetail:3 path:"allresults/myresults" filename:myfile cvsname:mycvs suffix:count counter:0

Command	display reports
What it does	Displays the reports in the current directory
Example	display reports
Command	display report <string:name|integer:index>
What it does	Displays the given report in the console
Example	display report myname_1.txt
Command	display report settings
What it does	Display the current report settings
Example	display report settings
Command	remove report
What it does	Removes the given reports in the current directory
Example	remove report myname_1.txt myname_3.txt

COMPUTING

Command	display settings
What it does	Display the current settings of pdvs, functions, probes
Note	Other evidence and report settings are not included.
Example	display settings
Command	compute information <string:probename>|<string:varname> <string:statename>|<integer:stateindex> [dg:<integer:dgindex>]+ [confirm:false|true]
What it does	Computes the information given the existing evidence and the evidence of the probe given its state. If one or more diagnoses are given, the utlity is computed only over these diagnoses, otherwise over all diagnoses.
Note	If ‘confirm’ is true (default) an overview of the settings is shown and a confirmation to continue is asked. The result shows the utility for each diagnosis given this state of the probe.
Example	compute information battery present dg:1 dg:4 confirm:false
Command	compute information <string:probename>|<string:varname> [dg:<integer:dgindex>]+ [confirm:false|true]
What it does	Computes the information given the existing evidence for all states of the probe.
Example	compute information battery dg:1 dg:4 confirm:false
Command	compute ei <string:probename>|<string:varname> [dg:<integer:dgindex>]+ [confirm:false|true] [weighted:false|true]
What it does	Computes the **e**xpected **i**nformation for the given probe. If weighted is ‘true’ the information is summed weighted by the probability of the diagnoses, otherwise the expected information is given for each probe and each diagnosis.
Example	compute ei battery confirm:false *or* compute ei battery weighted:true
Command	compute utility <string:probename>|<string:varname> <string:statename>|<integer:stateindex> [dg:<integer:dgindex>]+ [confirm:false|true]
What it does	Computes the utility given the existing evidence and the evidence of the probe given its state. If one or more diagnoses are given, the utlity is computed only over these diagnoses, otherwise over all diagnoses.
Note	If ‘confirm’ is true (default) an overview of the settings is shown and a confirmation to continue is asked. The result shows the utility for each diagnosis given this state of the probe
Example	compute utility battery absent confirm:false
Command	compute utility <string:probename>|<string:varname> [dg:<integer:dgindex>]+

	[confirm:false|true]
What it does	Computes the utility given the existing evidence for all the states of the probe.
Example	compute utility battery confirm:false
Command	compute eu <string:probename>|<string:varname> [dg:<integer:dgindex>]+ [confirm:false|true]
What it does	Computes the **e**xpected **u**tility for his probe.
Example	compute eu battery confirm:false
Command	compute meu [dg:<integer:dgindex>]+ [confirm:false|true]
What it does	Computes the **m**aximum **e**xpected **u**tility over all active probes.
Note	The results for each probe and each state for each diagnosis is displayed also.
Example	compute meu dg:3 dg:4 confirm:false
Command	compute suggested-probe [confirm:false|true]
What it does	Computes the suggested probe given the current evidence.
Example	compute suggested-probe
Command	compute probe-scenario [confirm:false|true]
What it does	Computes the probe scenario.
Note	A report is displayed and exported as given in the report settings.
Example	compute probe-scenario
Command	compute probe-scenario [infoprevalence;false|true]+ [confirm:false|true]
What it does	Computes the probe scenario, with the infoprevalence as given for each level.
Note	Infoprevalence is only used in utility function 'weighted cost' as described in the assignment (section 7.3.1). Repeat infoprevalence for each level. If no level is given (anymore) the last alpha-value is used for all next levels. If an infoprevalence is given, the value of alpha will be recomputed. If infoprevalence is set to clear all infoprevalences are removed. A report is displayed and exported as given in the report settings.
Example	compute probe-scenario confirm:false infoprevalence:false infoprevalence:true

EXPERIMENTS

Command	set costvariance equal [cost:<double:cost>]
What it does	Sets the cost equal to 'cost' (or the default cost of 'cost' is not provided) for the given probes. If no probes are given, the cost is given to all probes.
Note	If probes are given, the other probes keep their current cost.
Example	set costvariance equal cost:30 *or* set costvariance equal cost:40 trigger1 trigger2
Command	set costvariance polar min:<double:cost>
What it does	Polar costs are given as provided: cost of the probes between 'min' and 'max' are set to min; cost of the probes after 'max' are set to max.
Note	If not all probes are provided, the other probes, not given in the command, keep their current cost.
Example	set costvariance polar min:10 battery trigger1 trigger2 max:90 plug1 plug2 light2
Command	set costvariance polar min:<double:cost> max:<double:cost> distribution:random
What it does	The polar costvariance is randomly distributed for all probes. The two sets with minimum and maximum cost do not have necessarily the same size. One of them can be empty (ending in equal costs).
Example	set costvariance polar min:10 max:90 distribution:random
Command	set costvariance scattered min:<double:cost>
What it does	Scattered costs are given at the probes in the order given in the command: the minimum at the first probe, the maximum at the last. If not all probes are provided, the scattered costs are distributed over the given probes. The other probes, not given in the command, keep their current cost.
Example	set costvariance scattered min:10 max:90 battery trigger1 trigger2 plug1 plug2 light2
Command	set costvariance scattered min:<double:cost>

What it does	The scattered costvariance is distributed randomly for all probes.
Example	set costvariance scattered min:10.0 max:90.0 distribution:random
Command	set experiment [infoprevalence:false|true|clear] [count:all|count]
What it does	If the strategy is ‘random’, computes <count> probescenarios (default:1000) If the strategy is set to ‘cheapest’ or ‘meu’. If utility is ‘weigted cost’, repeat infoprevalence for each level. If no level is given (anymore) the last alpha-value is used for all next levels. If an infoprevalence is given, the value of alpha will be recomputed. If infoprevalence is set to ‘clear’, all infoprevalences are removed. If count is set to all, all permutations are computed for scattered and polar costs. If count is set to a number, the probe-scenario is computed 'count' times with random attributed costs (given the cost distribution). This is also the case for equal costs, because, if information is equal, the next probe is also chosen randomly. For other strategies the experiment is not defined.
Example	set experiment infoprevalence:true infoprevalence:false count:all
Command	run experiment
What it does	Runs the experiment with the given settings
Example	run experiment
Command	compute optimal [iteration-limit:<integer:nr-of-iterations>] [time-limit:<int:seconds>] [confirm:false|true]
What it does	Computes the optimal constant(s) for the given utilityfunction, following the heuristic as described in the paper. The iterationlimit limits the number of calculations. For utilityfunction weighted_cost, this is the number of levels, the infoprevalence is integrated (mind that for each level the nr of computations is 2^n). For utilityfunction linear_utility, this is the number of computations to refine the constant. The defaultvalue of nr-of-iterations is 1. The time-limit limits the start of thecomputation time to this limit (in seconds). A computation can be started within this limit, and still proceed longer, till it ends. The default value of time-limit is -1, which means no limit. If there is a conflict between number of iterations and time limit, the latter wins. This command does not matter for the utilityfunction Information_per_cost.
Note	An additional report is displayed and exported as given in the report settings.
Example	compute optimal iteration-limit:3 time-limit:120
Command	compute minimal [confirm:false|true]
What it does	Computes the minimal diagnostic cost for a system by tracking all possible combinations of sequences of probes dependent on the result of the previous probe.
Note	This command has a heavy computational load.
Example	compute minimal confirm:false

SCRIPTS

It is possible to put a sequence of commands together in a script. With the command `run <script>`, the script can be executed. It is possible to integrate running scripts in scripts. The URI of the script can be given as an absolute path to the file or relatively to the place of execution.

In the script each command must start on a new line. It is possible to write comments by putting a number-sign (hashtag) at the beginning of a line. Non-visible signs as blank lines or tab-signs are negated.

After executing the script, when success message is showed, it is possible to type manually other commands, except if the script ended with ‘quit’ or ‘exit’. When using the ‘compute’ command, parameter ‘confirm’ must be set to ‘false’, otherwise the script will be waiting for an input. The ‘print’ command can be useful to structure the output displayed

In that way a script can e.g., be used to initialize a particular situation. Mind that the commands ‘set diagnoses’ and ‘set probes’ must be called to generate the possible diagnoses and probes, given the network, the healthgroups and the problem defining variable

A typical initialization script, with some other commands afterwards, is as follows:

```
set network ‘testnetwork’
set healthgroups connection
set pdvs light1 off
set diagnoses
set probes cost:50
set report display:true displaydetail:0 export:true exportdetail:3
path:"resources\results" filename:"test.txt" csvname:"test.csv" suffix:time
set informationfunction entropy_reversed
set utilityfunction weighted_cost alpha:0.1
set strategy meu

# do something
print “Compute the probe scenario”
compute probe-scenario confirm:false

# compute probe scenario if battery is present
set evidence batter present active:true
compute probe-scenario confirm:false
```