McScM - Feature # 21: redesign of command line parameters with respect to different checker-modules

Status:	New	**Priority:**	Normal
Author:	Alexander Heußner	**Category:**	
Created:	07/05/2010	**Assignee:**	Alexander Heußner
Updated:	04/08/2011	**Due date:**	
Subject:	redesign of command line parameters with respect to different checker-modules		

Description:

currently, all modules expect and accept the same parameters/arguments on the command-lines even if they ignore those that are not meant for themthemselves. i propose that we allow each "module" (i will call module the different verification algorithms implemented) its own parameters in addition to global parameters (like --help and -no-verification etc.).

first ideas are to apply some kind of "mplayer" like syntax, that allow to give each module it's own, local parameters

{{{
mcscm.native -cegar limit=10:split-simplificaiton=left:graph-exploration=bwd-bfs -no-verification -statistics
}}}

this would also easily allow to use the same <param>=<value> syntax for a simple cfg file ;-)

some open questions:
* does one need to additionally choose the mc-engine ? (hence, can there be params for engines that i do not use?)
* what other syntaxes could be possible ?
* what is possible with standard OCaml features ?
* how to present this "nicely" when using --help (idea: clean up --help to a minimum and use a --long-help or --documentation for a long, man-like help feedback)

History

07/06/2010 03:35 pm - Grégoire Sutre

We should allow a configuration file that provides the same functionality as command-line options. As discussed, the format could be the one of Windows INI files (which is also used in e.g. kde, git, bazaar): http://en.wikipedia.org/wiki/INI_file.

<pre>
[Module1]
option1 = foo1
option2 = false
option3 = true

[Module2]
option1 = toto2
option23 = truc

[Main]
verbose = true
</pre>

The command-line equivalent would be:

<pre>
-Module1 "option1=foo1:-option2:option3" -Module2 "option1=toto2:option23=truc" -Main "verbose"
</pre>

The above example uses a shortcut syntax for booleans, but we could also write:

<pre>
-Module1 "option1=foo1:option2=false:option3=true" -Module2 "option1=toto2:option23=truc" -Main "verbose=true"
</pre>

For convenience, we should also allow simple command-line options that implicitly refer to the Main module. For instance, we could assume that options starting with an upper-case letter define module options, and options starting with a lower-case letter implicitly apply to Main.

We should allow command-line options even for (model-checking) modules that are not used (same for configuration file).

For documentation, I agree that -help should provide a minimal help (only options of the Main module?), and we could use -documentation to ask each module to dump a description of its configuration options.

04/08/2011 01:45 pm - Grégoire Sutre

- Target version set to 2.0

- Start date deleted (07/05/2010)