

Statemate[®]

ModelChecker & ModelCheckerPlus

Release 4.6.1

Release Notes



The IBM logo, consisting of the letters "IBM" in a stylized, blue, horizontal-striped font, followed by a registered trademark symbol (®).

License Agreement

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of the copyright owner, BTC Embedded Systems AG.

The information in this publication is subject to change without notice, and BTC Embedded Systems AG assumes no responsibility for any errors which may appear herein. No warranties, either expressed or implied, are made regarding Rhapsody software and its fitness for any particular purpose.

Trademarks

IBM, Rational, and Statemate are registered trademarks of IBM Corporation.

All other product or company names mentioned herein may be trademarks or registered trademarks of their respective owners.

© Copyright 2000-2009 BTC Embedded Systems AG. All rights reserved.

Contacting IBM Rational Software Support

IBM Rational Software Support provides you with technical assistance. The IBM Rational Software Support Home page for Rational products can be found at <http://www.ibm.com/software/rational/support/>.

For contact information and guidelines or reference materials that you need for support, read the [IBM Software Support Handbook](http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html) (<http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html>).

For Rational software product news, events, and other information, visit the [IBM Rational Software Web site](http://www.ibm.com/software/rational) (<http://www.ibm.com/software/rational>).

Voice support is available to all current contract holders by dialing a telephone number in your country (where available). For specific country phone numbers, go to <http://www.ibm.com/planetwide>.

Before you contact IBM Rational Software Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:

What software versions were you running when the problem occurred?

Do you have logs, traces, or messages that are related to the problem?

Can you reproduce the problem? If so, what steps do you take to reproduce it?

Is there a workaround for the problem? If so, be prepared to describe the workaround.

1 Package Identification

1.1 Vendor Identification

BTC Embedded Systems AG Buschstrasse 1 26127 Oldenburg +441 96 97 38 0 (voice) +441 96 97 38 64 (fax)

1.2 Tool Identification

IBM Rational Statemate ModelChecker 4.6.1 - Commercial Release

IBM Rational Statemate ModelCheckerPlus 4.6.1 - Commercial Release

1.3 Media Identification

CD-ROM (Solaris):

ModelChecker/ModelCheckerPlus/Documentation/Example Project

CD-ROM (Windows NT/2000/XP):

ModelChecker/ModelCheckerPlus/Documentation/Example Project

1.4 Date of Release

July 2010 for IBM Rational Statemate 4.6.1

1.5 Installation

See the ModelChecker/ModelCheckerPlus Installation Guide.

IBM Rational Statemate 4.6.1 has to be installed previously.

1.6 Platform Issues

Supported platforms are

Sparc Solaris 2.10

Windows 2000, Windows XP

with the same window systems as IBM Rational Statemate.

2 Features

There are two verification packages available, one is the *Statemate ModelChecker* (MC) and the second is the *Statemate ModelCheckerPlus*, which includes, in addition to the whole functionality of the first one, abstraction techniques and customizable templates to define certification tasks. Both packages are based on a technology called model checking. Model checking provides a technology, which checks the dynamic behavior of a design regarding specific properties. It is complete in a mathematical sense and is equivalent to exhaustive testing.

Today exhaustive testing is the only way to find design errors. This is becoming impossible with the complexity of embedded systems today. With Statemate ModelChecker and ModelCheckerPlus users can benefit from formal verification technology in a very easy to use tool that checks designs for hard to detect errors throughout the development process. The benefit of MC is to speed up the debugging process, and to find errors early in the process where they can be fixed easily and thus at lower cost.

2.1 Statemate ModelChecker

The first package is an easy to use tool targeted at every user of Statemate. This product provides push-button analysis...and requires no training. To this end, the ModelChecker provides predefined standard checks. In general, the user has only to select Statemate design objects (textual or graphical) in order to run a predefined check type.

First the user checks to see whether the design model is complete and that they have not built in any robustness conflicts such as:

Non-determinism checks to make sure the system does not reach a state where it can go in more than one direction and there is no definition of priority for which way to go.

Write-write conflict checks to ensure you are not trying to set the value of an element to 2 different values at the same time.

Read-write hazards - flags a situation where you are reading data that is in the process of changing. Here you need to define the right sequence or you may read an old value.

Range violation checks to test that there are no overflows and underflows in assignments to data items.

Once you've found a problem, ModelChecker will record a simulation scenario that can be replayed on the design model to see what happened – making debugging much easier. Today exhaustive simulation is the only way to find these errors.

Second users check their models more thoroughly during the course of designing the product. The ModelChecker allows the user to perform “drive to system configuration” or “reachability” testing which checks if the system can reach a certain state. This helps you find unwanted behavior.

A general input scheme called Drive-to-Property allows the user to specify any possible system state. Having a specification of the system state it is a pure push button task to execute the analysis. As result of the ModelChecker run, an executable SCP (Simulation Control Program) will be automatically generated, which drives the simulated Statemate design to the desired state.

2.2 Statemate Model Checker Plus

The second version we are offering contains everything in the everyday user package plus additional features used to certify a final model. This version can check to see if a series of events will or will not happen under a set of predefined conditions. While the ModelChecker is typically used on subsystems or system functions, during the model development process, the ModelCheckerPlus is used by lead engineers on the final system to ensure it meets the system requirements. It is also used by quality engineers for acceptance testing from systems implemented by suppliers or other groups in the company. Safety

engineers use the ModelCheckerPlus to ensure safety requirements are met for the overall system. The ModelCheckerPlus includes a library of templates that enable the user to capture requirements, in particular sequential, temporal dependencies. For example there is a template that checks to ensure one event will always occur before a second event happens. Another powerful capability is to abstract the model to mask those areas of the model that are not relevant. This simplifies the verification task significantly reducing processing time. Typical processing time to run a check takes minutes or even seconds, although on very complex models and when checking for very complex requirements it could take hours.

The expected result should in general simply be true, but in cases where the desired requirement is violated, an error-path SCP will be generated, driving the simulated system to the state where the violation becomes visible.

The key benefits from the ModelCheckerPlus include a reduction in testing time, improvements in product quality, and elimination of catastrophic failures.

2.3 Verification Profiles

ModelChecker and ModelCheckerPlus store all user input, analysis and proof selections and definitions, in a profile. These profiles can be exchanged via and stored in a databank as all other ingredients of a Statemate design project. The profiles of both, ModelChecker and ModelCheckerPlus, have the same file format. Thus, a ModelChecker profile can be used to open the ModelCheckerPlus. The other way round, when opening the ModelChecker with a ModelCheckerPlus profile, defined certification proofs are not executable.

3 Release News

3.1 New in this release of ModelChecker / ModelCheckerPlus

This version no longer supports Solaris 2.8 and 2.9. There are no feature updates, but the plugin and its documentation have been adapted to the new Statemate release.

3.2 New in release MC 4.6

The two model checking plugins are now part of the IBM Rational Statemate family. The pattern-based verification tool *ModelCertifier* has been renamed to *IBM Rational Statemate ModelChecker Plus*.

The version numbering of the previous plugin releases (ending in the previous release number 3.6) has been removed for this and future releases. The plugin releases are simply synchronized and numbered as the Statemate tool itself. Thus we now have ModelChecker and ModelCheckerPlus 4.6.

Dependencies from some open source (GNU) tools and libraries have been removed from the release. These free tools and libs can be taken from ftp://stm_plugins@btc-es.de/.

The installation has been adapted to the IBM look&feel. Furthermore, the installer for the Windows platform has been renewed. All documentation got an update.

3.3 New in release MC 3.6

Statemate ModelChecker/ModelCertifier 3.6 has been released for Statemate 4.5. There are no feature updates, but the plugin and its documentation have been adapted to the new Statemate release.

3.4 New in release MC 3.5

The ModelChecker/ModelCertifier 3.5 release has been adapted to Statemate 4.4. In particular, this includes support for the new **Transition Priorities** in state charts. The new **Complex Constant Value Definition** are not supported, only simple constant values may be used for MC. The Static Check messages of MC and the documentation have been adapted accordingly.

For Windows users, the MC user interface has been redesigned to get a better Windows-like look-and-feel. The dependency from an X-server has been removed. The set of packages that are installed for the cygwin runtime environment has been reduced. The user interface has been improved in its graphical presentation and some logging messages have been cleaned up to get better readability.

The ModelCertifier has been extended by a new special pattern called
P_triggering_Q_stable_X_steps_implies_S_within_Y_steps_if_stable_T.

The documentation in the Pattern Guide has been adapted. Furthermore for the following pattern minor documentation errors are fixed :

Cyclic: Q_notbefore_P

Cyclic: Q_while_P

Cyclic: finally_P_B

Flaws in the Semantics of the following pattern have been fixed:

Cyclic: P_triggers_Q_unless_S

Cyclic: P_triggers_Q_unless_S_within_B

First: P_implies_Q_atleast_X_steps_after_P

First: P_implies_finally_globally_Q_B
Init: P_implies_finally_globally_Q_B__immediate
Init: Q_while_P__after_N_steps
Init: finally_globally_P_B
First: P_triggers_Q_unless_S__after_N_steps
First: P_triggers_Q_unless_S_within_B__after_reaching_R

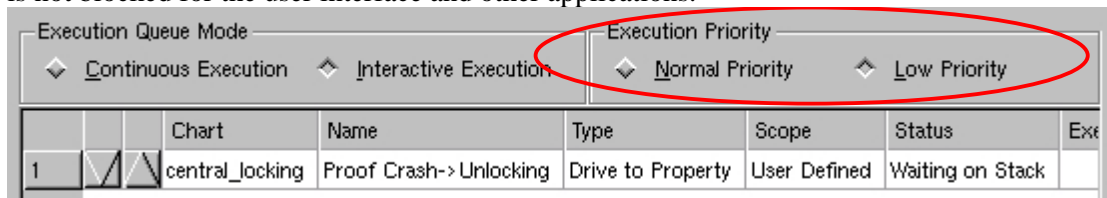
These changes and fixes may yield different results in your proof applications compared to previous releases.

3.5 New in release MC 3.4

The ModelChecker/ModelCertifier 3.4 release has been adapted to Statemate 4.3. The key improvement is a replacement of the previously used so-called “Bounded Engine”, now yielding much higher performance. With this release, a derivate of the SAT competition winner, MiniSat, has been integrated. Similar to the SAT competition results in particular for industrial applications, MiniSat has turned out to be the best available kernel engine for Bounded Analyses for Statemate models. The benefit is mainly a reduced runtime (about 30%, depending on the model) of complex verification tasks. Less memory consumption, less time-outs, and better results in Analyses tasks are also measured with the new engine.

Further enhancements are

A new option allowing the user to reduce the priority of the tasks that are executed. To this end, the Execution Queue tab in the user interface contains an option for selection of *Normal* versus *Low* Execution Priority. Low priority for the execution will lead to longer run time, but has the advantage that the machine is not blocked for the user interface and other applications.



The length of file names and paths has been drastically reduced to avoid the problem that Windows typically supports only a very limited length of file- and path-names. Execution errors due to “file name too long” should be solved now. Model and component names do not influence the (internal) file names any more.

Solaris 2.10 is now supported, too.

The Static Check has been slightly improved for some situations where Correctness Errors in the model (reported by STM’s Check Model tool) avoid an application of ModelChecker and ModelCertifier.

The installer (setup.exe) for the installation on Windows has been fixed: it could happen that the installer ended up in a deadlock when the user entered a non-existing installation path. Another fix has been implemented concerning the uninstallation of previous STM plugins in automated “Replay” of a recorded installation.

3.6 New in release MC 3.3.2

ModelChecker/ModelCertifier 3.3.2 are released for the maintenance release MR-2 of Statemate 4.2. The following changes and enhancements have been implemented:

A performance issue on some large models has been solved. When applying the so-called bounded engine it could happen in previous releases, that starting the engine took several minutes where in the new release only a few seconds are required.

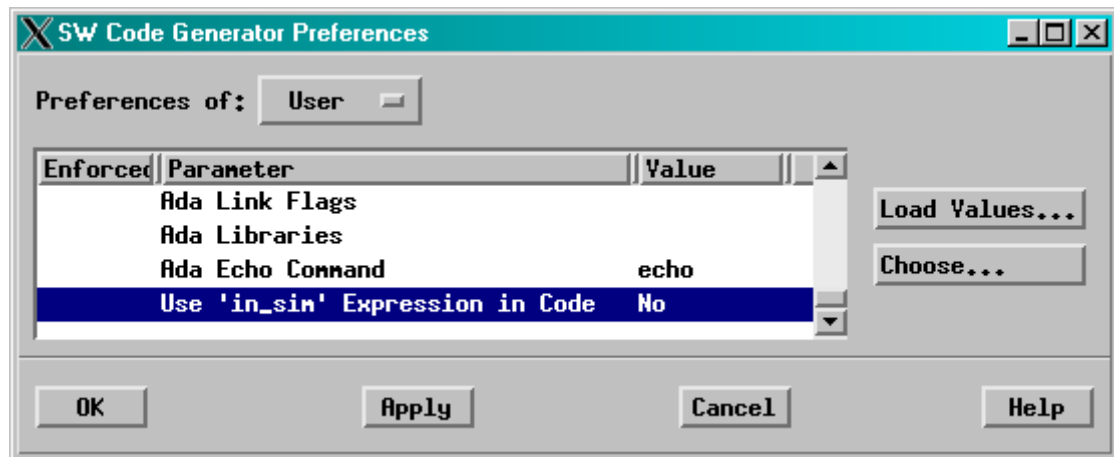
Parsing and checking properties in the drive-to-property dialogs has been improved with respect to the runtime to improve the interaction with the user interface.

The semantics of Statemate's `in_sim(event_expr)` operator say that the `event_expr` is evaluated and executed by Statemate's simulator, but is ignored by any other tools in the Statemate tool family. As consequence, the interpretation of the model information by MC/MCF was different from the one by the STM simulator. As further consequence, the SCPs generated by MC/MCF do not run in the STM simulator whenever an `in_sim()` operator is involved.

With this release, the STM plugin takes the information from the Statemate preferences settings into account. There is an option introduced with Statemate's 4.2 maintenance releases in the

```
Project -> Preferences Management
        -> Statemate Prototype C Code Generator...
            -> Use 'in_sim' Expression in Code
```

that allows the user to enable the interpretation of the `event_expr` in the code generation and in the ATG and ModelChecker and ModelCertifier plugins, too.



To this end, set the value to “Yes” and Re-Compile the design model inside MC/MCF.

A bug has been fixed that occurred when compiling the model into the plugins representation. Whenever the model contains a `length_of()` operator applied to an component of a structured data item, an unexpected compilation failures has been reported. This is fixed now.

3.7 New in release MC 3.3.1

ModelChecker/ModelCertifier 3.3.1 are released for the maintenance release MR-1 of Statemate 4.2. For this Statemate version the following improvements and enhancements are implemented in MC/MCF:

The interface computation has been fixed for the top level of models with asynchronous simulation semantics. Data_items which are read from the environment as inputs and modified in the model as well are now treated as so-called “inouts” in MC/MCF. The gap between the MC/MCF interpretation of the model and the STM simulator behavior is closed at this point now.

ModelCertifier only:

The pattern library has been extended by one new core pattern

`P_stable_X_steps_triggers_S_releasing_Q_within_Y_steps`

and the respective derivatives.

The activation mode for “iterative” and “invariant” pattern instances has been renamed to “cyclic”.

The semantics of some pattern (and the respective derivatives) has been slightly modified (see pattern library documentation for details):

`P_stable_X_steps_triggers_Q_within_Y_steps_unless_S` and

`P_stable_X_steps_implies_finally_Q_B`

The documentation Pattern Guide has been adapted to the new and changed pattern definitions and modes.

The operator `in_sim` has been added to the list of operators in the User Guides and Limitations document. Note that, as defined by Statemate “`in_sim(event_expr)`” is replaced by “`empty_event`” by all Statemate tools but the simulator. Thus, ModelChecker and ModelCertifier do not take the `in_sim(event_expr)` into account.

The analysis with the bounded engine has been improved. In ModelChecker tasks you may get the result “unreachable” where previous releases reported “Not reached” with “Engine depth <n>” steps. Proof results in the ModelCertifier may become more often “true” instead of “No Counterexample” with “engine depth <n>”.

3.8 New in release MC 3.3

ModelChecker/ModelCertifier 3.3 are released for Statemate 4.2. For this Statemate version the following improvements and enhancements are implemented in MC/MCF:

Support for Statemate’s new `reset_element` and `reset_all_elements` operators.

A new version (1.5.21) of the cygwin runtime environment for Windows is used. The new environment avoids conflicts that occurred when the user has cygwin for other purposes too. Furthermore, MC/MCF benefits from memory allocation improvements that come with the new cygwin.

Inside the MC/MCF technology building blocks, several improvements have been implemented. Overall performance improvements:

Reduced amount of allocated memory during runtime

Shorter runtime – in particular for applications with the bounded analysis engine

Applications of MC/MCF to small models will probably not take advantage of the improvements. Performance profits on larger models depends on the length of the (expected, generated) simulation runs (test vectors).

A complexity problem that yielded an internal parse error and blocked the generation of model checking results has been solved.

A bug fix for a compilation failure in the context of assignments of don’t care values to outputs in truth tables has been made.

The support for Solaris 2.6 and 2.7 platforms and for Windows NT have been skipped, since these are not supported by Statemate any more.

3.9 New in release MC 3.2.2

The ModelChecker and ModelCertifier 3.2.2 are released for Statemate 4.1 MR-2. There are only slight modifications that do not concern the application and usage of the tools. A bug has been fixed that concerns the treatment of history connectors. Their semantics in general and resetting history in particular were faulty in previous releases. A second bug fix is related to user defined functions of type string. When they are used in assignments to context variables then the previous release crashed during the static analysis of the model.

Due to adaptations to the latest API of Statemate this MC release does not work with previous Statemate releases. STM 4.1 MR-2 is mandatory.

3.10 New in release MC 3.2.1

MC 3.2.1 is a maintenance version of the previous release. It is tested with and released for the Statemate maintenance release STM 4.1 MR-1. Fixed problems in MC 3.2.1 relate

Truth-Table support: default rows in truth-tables were erroneously executed always at the end of the truth-tables execution.. Thus, the generated test vectors became useless for models with default rows in truth-tables.

Truth-Table support: implicit false “fs()” and true “tr()” events in table columns yielded a consistency error in the representation of the model’s behavior in MC. As consequence, the execution of MC failed during the model analysis with an “internal error” message.

Models where MC signals with an orange status lamp that some constructs are supported by abstraction only, got MC-generated SCPs that could not be compiled and executed. This is fixed.

On hierarchy levels below the top level, component (sub-system) interface inputs from other system components that are part of the system model and not environment could not be frozen (simple assumption freezing dialog). Potentially given freezing information was ignored. This bug is fixed in MC3.2.1.

Array limits in user-defined types raised a crash in the (Re-)Compile step. This bug is fixed.

The so-called *bounded engine* reported always one step to much for generated error traces and drive-to-result traces. This regression problem is solved.

The *Installation Guide* has been updated.

3.11 New in release MC 3.2

The ModelChecker and ModelCertifier Release 3.2 has been adapted to the API of Statemate 4.1. New features and fixed bugs are

- Support for Windows XP’s Service Pack 2.
- A solution for the so-called “rebase” problem: in previous releases it frequently happened on Windows systems that after the ATG installation a rebase procedure had to be executed. This problem is solved.
- A problem with the display of interfaces in the user interface for models with case sensitive data items is fixed.
- In addition to the communication modes “input”, “local”, and “output”, a new mode “inout” has been introduced. All variable that are locally in the model read and written, but are also part of input flow lines from the environment to the system, get now the mode INOUT. In assumptions and properties is is possible to refer to the local part of these variables x and to the input part $inp(x)$.

A new locking mechanism prevents the user to start the two ModelChecker and/or ModelCertifier applications on the same profile at the same point in time.

The printable Pattern Documentation has been rewritten to give a better and complete overview about all available specification pattern.

3.12 New in release MC 3.1.2

There are no changes compared to the previous release, except that the installer allows the installation for the MR-2 of Statemate 4.0.

3.13 New in release MC 3.1.1

The ModelChecker and ModelCertifier Release 3.1.1 is identical to the 3.1 in its functionality, but adapted to Statemate's maintenance release 4.0 MR-1.

When you install this maintenance release of Statemate it is necessary to install this MC update too.

3.14 New in release MC 3.1

MC 3.1 is released for Statemate 4.0.

New is that major limitations in the support of expressions in timing operators has been removed. Timing expressions that contain Reals and user-defined functions in Action Language do not raise red lamps any more. The related part in the restrictions section in the User Guide document is adapted.

Note that, even when Real valued expressions are supported now, ModelChecker and ModelCertifier still round the concrete values to the next greater integer value during model analysis and verification. This is only an approximation of the original model behavior for timeout operations at Real, non-integer points in time.

Bug fixes are provided concerning

the usage of constant arrays on string constants, and

the availability of generated abstraction functions in the interface when the model is marked Orange for abstracted model contents.

the install script and documentation for the installation on Solaris systems has been made more robust.

3.15 New in release MC 3.0 – STM 4.0

ModelChecker and ModelCertifier 3.0 are released for Statemate 4.0. Based on feedback from professional users, new requirements, and improvements in technical challenges this release of MC implements a many news. The changes compared to previous releases include an easier usage achieved by a complete redesign of the user interface and a better performance of bounded analyses due to internal technical improvements. Details are

User Interface

- New layout of the user interface with different views to settings, test definitions, and results. A new browser presents in a separated section of the user interface (similar to Statemate) the chart hierarchy.
- New tool buttons “Statistic Report” and “Test Vector Export” and new icons for old functions.
- Global settings directly under the General Tab, no clicking through menus any more for this.
- Application Report “online” as view in top level interface.
- The mode (interactive/continuous) of the Execution Queue can be selected directly at the execution queue view.
- The ModelCertifier definition dialog has a new integrated “keyboard”
- The ModelCertifier online description of proof pattern is extended by graphical specifications of the pattern semantics and mode and activation conditions.
- Some menu items renamed to become more intuitive.
- Easier entering and correction/editing of fixed-point resolution and boundary values.
- New result status display.
- New syntax info text for Property and Simple Assumption definitions.

- Detailed error messages and highlighting for incorrect or invalid Properties and Assumptions.
- New hints about problems that arise when Freezing lists or Simple Assumptions are adapted/adjusting to a new model in a Re-Compile step.
- List in the user interface allow to hide columns (header in context menu).
- Special symbols and national characters are allowed to be used in names and comments.
- “Are you sure?” popup before overwriting file with new application report.
- Field for comments in Application Report.
- Improved control via keyboard.

Property Language like Statemate Action Language

- The language used for entering properties and assumptions (and Freezing values) has been adapted to the Statemate action language used in trigger and guards.
- Syntax as well as some type checking is performed on user input. In case of errors, the erroneous part of the property is highlighted and a detailed message about the problem is shown.
- In cases where old definitions slipped through the syntax check in the past test definitions may become “not ready” for execution. Example are tr() and fs() applied to basic states or conditions compared with Reals. In trivial cases, the definitions are automatically updated, for example the old notation BASIC_STATE^^ENTERED is changed to en(BASIC_STATE).

Installation

- Slightly modified procedure.
- New cygwin version (1.5.10) as run-time environment on PCs.
- Check about required packages when using existing cygwin installations.
- Updated Installation Guide. Hints to deal a sporadic “rebase” problem.

Documentation

- Complete update of all documents with respect to new user interface and tool usage
- Extended description of known limitations and workaround in User Guide

3.16 New in previous release MC 2.5.3 – STM 3.3.1-MR2

The MC 2.5.3 is released for the maintenance version Statemate 3.3.1-MR2.

It is technically equal to MC 2.5.2 (see below), but comes with a completely redesigned and improved install setup for Windows platforms. In particular, recording (logging) installations and replaying recorded installations are now easy-to-use functionalities. On windows platforms, the installation of the cygwin runtime environment is now one click only.

3.17 New in previous MC 2.5.2 – STM 3.3.1-MR1

The ModelChecker and ModelCertifier 2.5.2 plugins are released for Statemate 3.3.1-MR1. Compared to the previous MC release, it contains no new features, but the following bug fixes:

Profile Update Progress window always in background: fixed

InstallGuide “adjusting cygwin memory limit” (Call 53767)

The adjustment is not necessary any more. This is no mentioned in the Install Guide.

Statemate models with library components raised failures of the MC compile step. Fixed.

Data items with type Condition Array or Event Array raised internal simulation errors with the effect that desired SCPs could not be generated.

3.18 New in previous release MC 2.5.1 – STM 3.3.1

The ModelChecker and ModelCertifier 2.5.1 provides

A bug fix for opening and updating old ModelChecker and ModelCertifier profiles,

A bug fix for using data items of type array of array in pattern instances, assumptions and properties,

Implements an performance improvement for ModelChecker analysis tasks when fixed-point mappings are used for REAL data items.

3.19 New in previous release MC 2.5 – STM 3.3.1

The ModelChecker and ModelCertifier 2.5 plugins are released for STM 3.3.1. STM 3.3.1 extends the action language by new else/default trigger. These new constructs are fully supported by MC 2.5 .

New in the list of action language elements that are supported by MC 2.5 are the predefined functions trunc() and round(). Models that use these functions do not yield a Red lamp any more, only the usage of trunc and round in properties and assumptions in MC definitions is still not possible.

The Statemate simulator and code generator options for the evaluations of truth tables are fully supported now. Like in Statemate, the setting of an “Upon Change” or “Every Step” evaluation of truth tables is selected in the MC profile.

On the technical side, some improvements have been made. In particular when input freezings are used, the required run-time and memory is reduced in MC applications. States and transitions that become directly unreachable as consequence of frozen inputs are detected as unreachable by a fast static analysis, i.e. without (expensive) formal verification engines.

The user interface of the ModelCertifier comes with a new online help for the selectable patterns. The description of the patterns that was previously in the Certifier Pattern document only, is now online visible for each pattern. Thus, it is much easier to find the right one for the intended property than in previous releases.

A new kernel pattern extends the pattern library. It allows to specify requirements where some reached condition triggers logically or temporary another requirement.

3.20 New in previous release MC2.4 – STM 3.3

ModelChecker and ModelCertifier are released for Statemate 3.3. The new possibility in STM 3.3 to specify bounds and default values for Integer and REAL data items is taken into account by MC/MCF. The bounds are default values in the fixed-point mapping tables. A new possibility to inherit already specified fixed point mappings from higher chart levels will further speed up the usage of MC/MCF.

Improvements like a better result presentation make the usage of ModelChecker and ModelCertifier more comfortable. The most prominent improvement is the integration of Bounded Proof in the ModelCertifier. While the already available Complete Proof has its limits in the size and complexity of the model, the Bounded Proof capability can analyze the model very fast up to a certain depth. This depth is measured in analyzed steps that can be set by the user.

3.21 New in previous release MC2.3 – STM 3.2

MC 2.3 supports 32 bit integer ranges and fixed point resolutions with 32 bit. This allows a higher precision in the treatment of computations on REALs. Data items of type REAL are now presented in groups, where each group contains data items that are directly related to each other inside the model. Thus, a more fine granular resolution per group allows to use high precision where necessary and a lower resolution to decrease complexity whenever it is possible.

The user interface (GUI) is improved: the data item list in the drive-to-property and pattern window can now be filtered like in the Statemate data dictionary browser ‘Name pattern:’ search. These windows are resizable now, thus there is more room for the property and requirement definition. The available operators that may be used in the definition can be selected in a new window that can be put aside and used with double clicking.

Of high value is the possibility to copy/past analysis and proof definitions, when existing definitions contain large and detailed freezing lists and simple assumptions. Entering them for each analysis and proof again is not necessary any more.

On a technical level, the 32 bit support has been added (previously there was a 16 bit limitation) without recognizable performance lost.

3.22 New in previous release MC 2.2 – STM 3.2 with Prover Plug-In

The Statemate construct of *nested generics* (generics used in generics) is now supported. The restriction to one level only of generic activities is removed.

New in this release is a generator for so-called *application documents*. These are ASCII-text- or html-based documents that contain the analysis and proof definitions together with your specified assumptions, fixed-point mappings and verification results.

MC 2.2 is also a maintenance release. The pattern selector in the ModelCertifier is slightly modified to get a better overview about the available verification templates. Analysis and proof definitions that require either a fixed point mapping for REALs or an explicit abstraction get now the new status ‘*abstraction required*’ when the definition is incomplete in this sense.

The Tutorial on the ModelChecker is completely rewritten. More basic concepts are explained together with a section on the different MC kernel engines and the treatment of complexity issues. The exercises are updated and completed on the basis of a renewed Statemate demo project.

The performance got an important speed-up for the *bounded analysis* technique introduced with the previous release. The run-time as well as the required memory are highly improved.

3.23 New in previous release MC 2.1 – STM 3.2 with Prover Plug-In

This release adds a new possibility to perform *bounded* analyses compared to *complete* analyses. Previous releases implemented always *complete* checks. In all ModelChecker functions (Non-determinism, race, overflow, reachability analyses) the user may now decide to reduce the run-time by restricting the number of simulation steps that are to be analyzed or by restricting the run-time itself. Technically, bounded analysis is implemented by a new integration of a verification engine of Prover Technologies.

While a complete analysis verifies for example that the design model is free of any race or overflow, the analysis bounded to n simulation steps would tell you that there is no race or overflow within the first n steps (for any possible combination of inputs). In almost all cases the bounded analysis is much faster than a complete verification and can handle larger models.

3.24 New in previous release MC 2.0 – STM 3.1

Compared to the previous release of MC 1.1, this version provides support for

A new Range Violation Check. This new analysis is available in ModelChecker and ModelCertifier. It provides a kind of over- and underflow checks in assignments to finite (bounded) data items.

Some more Statemate constructs, in particular Component Libraries, Default Transitions with Condition Connects, Best Match implementations of activities and actions, enum literals in array indices and predefined functions on enums, etc.

Automatic hiding of (abstraction from) previously unsupported predefined Statemate functions, C-code, Ada, and other user code inside ModelChecker/ ModelCertifier. Analysis and proofs are performed by ModelChecker/ ModelCertifier under the assumption that these hidden (abstracted) constructs provide arbitrary functionality to the design model.

A new abstraction technique with reduced model-checking complexity for asynchronous models.

Integer expressions in time expressions.

Additionally, some minor restrictions are removed, the report about parts in a model that are today unsupported by MC are extended. Further more, the explicit interface of input and output objects computed by MC is listed for each selectable chart in the MC graphical user interface.

4 Known Limitations

It is absolutely necessary to have the right expectations when starting to apply Model Checking and Model Certification to a Statemate design.

Both the ModelChecker and ModelCertifier support the different time models of the Statemate Simulator (i.e., synchronous and asynchronous). However, no support is given to the code-generator “Real Time” time model nor to C or ADA code implementations. The User Guides explain the impact of the selected time model to Model Checking and Certification.

4.1 Unsupported Chart and Action Language Features

Not all definitions that are possible in a Statemate design model are suitable for ModelChecker and ModelCertifier. For example, combinational assignments are not supported. A detailed list of unsupported features can be found at the end of the User Guide.

When applying ModelChecker and ModelCertifier to a design (for the first time), a static check will be performed, marking the design model components as Red, Orange, Yellow, or Green. Red components contain features that are not supported. The source of the problem is explained to the user in the User Interface. But: like a compiler terminates compilation after it finds an error, it may happen that the static check fails to show *all* sources for Red markings at once.

4.2 Execution Error

An Execution Error will appear, when user defined analyses or proofs refer to objects (data items, events) that are not present in a modified and recompiled design model.

4.3 SCP not generated

A similar situation, where the information for the user could be improved, occurs, when run-time overflows may occur in a Statemate simulation. All analyses and certification proofs except the ‘Range Violation Check’ are performed under the assumption that the design model is free of dynamic (range or array boundary) overflows. If they occur in a design model, this may prevent the ModelChecker from generating SCPs. The result messages and the Range Violation analysis can be used to determine which data-item has the dynamic overflow.

4.4 SCP do not drive the Statemate simulation as expected

This may happen if a chart is marked by the Static Check as Orange. Due to automatic abstraction from some function or subroutine the analysis or proof may assume effects that are not produced during the simulation. Data items that are affected by abstracted functions or subroutines become input objects that can be restricted with assumptions inside ModelChecker/ModelCertifier. Thus, user provided assumptions can avoid these failing SCPs. Other solutions may be to change the design model or to provide implementations for subroutines in Action Language instead of user code.

4.5 Executing Analyses and Proofs on a changed Design/Workarea

ModelChecker and ModelCertifier perform user selected/defined analyses and proofs on their own representation of the Statemate design. To this end, there is a compile step from the Statemate design data to a representation suitable for verification tasks. If a design is modified, a design is checked out from the databank, or if a verification profile is checked out from the databank, then a (re-) compilation is necessary. If doing this while keeping old analyses and proof definitions it may happen that these definitions become invalid. For example, a drive-to-state becomes invalid if the previously selected state is actually removed from or renamed in the design. Now, when keeping definitions in a re-compile the ModelChecker and ModelCertifier adapt the definitions – if necessary – automatically. Modified definitions are marked by “* *” which prefix the definition name. The stars “* *” disappear when the user edits the definition again.

Mapping tables, that map data items of type real or (unbounded) integer to finite fixed point representations, are automatically extended if there are more such data items in the re-compiled design than in the old, kept definitions. The newly added data items get a default range from zero to zero. The user has to edit such definitions afterwards to enter more appropriate values.

4.6 Bugs

There is a known bug in the compilation step of the Statemate design data into an internal representation: the timing of scheduled actions is not always accurate. There is an erroneous additional delay of one step, that may become visible when defining properties which relate a scheduled to unscheduled actions.

Simulation control programs generated by MC do not run, if the selected scope for the model check run was a basic activity that is implemented by a truth table or mini spec.

Implicit stop events that signal the termination of a sub-activity are not correctly provided to the parent activity on a high hierarchy level.. Use explicit user defined event, please, if the parent activity need this termination information (e.g. in a control activity).

5 Testing Performed

Interactive as well as regression tests with main focus on the correctness of computed verification results are performed.

6 Documentation

There are

- These Release Notes
- Installation Guide
- ModelChecker User Guide
- ModelCertifier User Guide
- ModelCertifier Pattern Manual
- ModelChecker Tutorial

The documents are accessible via the Help menu of the ModelChecker and ModelCertifier.

7 Training

Training on ModelChecker and ModelCertifier are available. For the ModelCertifier it is recommended to attend a training class.