Qfsm User Manual

Stefan Duffner

Camille Decock

Qfsm User Manual

Stefan Duffner
Camille Decock

Version 0.52

Table of Contents

List of Tables

Chapter 1. Introduction

What is Qfsm?

Qfsm is a graphical editor for finite state machines written in C++ using Qt the graphical Toolkit fromTrolltech [http://www.trolltech.com].

Finite state machines are models to describe complex objects or systems in terms of the states they may be in. In practice they can be used to create regular expressions, scanners or other program code as well as for integrated curcuit design.

Current features of Qfsm are:

- Drawing, Editing and Printing of states diagrams
- Binary, ASCII and "free text" condition codes
- Integrity check
- Interactive simulation
- Diagram export (EPS, SVG and PNG format)
- AHDL/VHDL/Verilog HDL/KISS/vvvv Automata code export
- State table export in Latex, HTML and plain text format
- Ragel file export (used for C/C++, Java or Ruby code generation)
- State Chart XML (SCXML) export

Copyright and license information

Installing the Qfsm software

Binary installation

Windows

There exists a (graphical) Windows installer that can be downloaded from here [http://www.sourceforge.net/projects/qfsm]. The installation should be straightforward and doesn't require any other software to be installed.

Linux

An RPM package for openSUSE is provided here [http://www.sourceforge.net/projects/qfsm]. RPM packages for other Linux distributions might be provided on other websites.

Mac OS X

Binary packages for Mac OS X may be provided by third-party websites, for example here [http://mac.softpedia.com/get/Math-Scientific/Qfms.shtml].

Installing from source code

Qfsm has to be compiled using the CMake buildsystem. Among the advantages of CMake are that it provides excellent cross-platform support, that it can create project files for various IDEs such as KDevelop, XCode and MS Visual Studio and that it builds out-of-source, keeping the source tree clean of temporary files.

Requirements

- CMake [http://www.cmake.org], version 2.6 or higher
- Qt SDK [http://qt.nokia.com], version 4.4.3 or higher

See the documentation of these tools/libraries for an explanation how to install them.

Supported Platforms

In principal, Qfsm compiles and runs on any platform supported by the Qt library version 4.4 (or higher), i.e. Linux/Unix (e.g. AIX, FreeBSD, HP-UX, IRIX, Solaris), Windows (98, NT 4.0, ME, 2000, XP and Vista) and Mac OS X (version 10.3.9 and higher).

Building under GNU Linux/Unix

1. Unpack the source code into the current directory
   tar xvfz qfsm-0.52.0-Source.tar.gz --directory .
   and change to that directory.

2. Before invoking CMake you have to make sure that the program qmake is in your global PATH environment variable. The location of qmake depends on your operating system or Linux distribution. On openSUSE for example it is under: /usr/lib/qt4/bin/ Thus, if you are using bash, you would have to type:
   export PATH=/usr/lib/qt4/bin:$PATH

3. Type:
   cmake .

4. Then:
   make

5. Finally, install Qfsm (as root):
   make install

This will install the executable "qfsm" to /usr/bin/ and the documentation to /usr/share/doc/qfsm/.

If CMake fails to find any of the dependencies but you know you have the development headers/libraries installed, add -DLIBRARY_SEARCH_DIRS=<path/to/lib> and -DINCLUDE_SEARCH_DIRS=<path/to/include> to the cmake command.

If you want to create a project file for your IDE (e.g. KDevelop), add -GKDevelop3 to the cmake command above. This will create a .kdevelop file in the build directory which can be opened by KDevelop.

Building under Windows

In order to build Qfsm under Windows you have to execute the program CMakeSetup, specify input (source) and output directory and click on "Configure". You may choose between different development environments (e.g. Visual Studio) and CMake will create the respective project files. You can then compile Qfsm as usual with your preferred build tool. For more details, refer to the CMake documentation: http://www.cmake.org/HTML/Documentation.html.

Chapter 2. Using Qfsm

Using the main menu

This section briefly explains the functions available through the main menu.

File

To create a new file:	select “New”. For more details, see the section called “Creating and modifying a Finite State Machine”
To open an existing Qfsm file:	select “Open”.
To open the most recently opened Qfsm files:	select “Open Recent”.
To save the current FSM to a Qfsm file:	select “Save”.
To save the current FSM under a different name:	select “Save As”.
To export the current FSM to a foreign file format:	select “Export”. For more details, see the section called “Exporting”
To print the current FSM:	select “Print”.
To open a new window with a separate working area where a different FSM can be edited:	select “New Window”. Note that you can copy, cut and paste states and transitions from/to different FSMs.
To close the current FSM:	select “Close”.
To quit Qfsm:	select “Quit”.

Edit

To undo the last action:	select “Undo”.
To cut the currently selected states and transitions to the clipboard:	select “Cut”.
To copy the currently selected states and transitions to the clipboard:	select “Copy”.
To paste the clipboard into the current FSM:	select “Paste”.
To delete the currently selected states and transitions:	select “Delete”.
To switch to the select mode:	select “Select”. For more details, see the section called “Using the Select mode”
To select all states and transitions of the current FSM:	select “Select All”.
To deselect all objects:	select “Deselect All”.
To open the options dialog:	select “Options”. For more details, see the section called “Setting the options”

View

To show/hide the state codes inside the states:	select “State Codes”. Each state has a unique identifier, called state code, which is an integer that is automatically determined by Qfsm.
To show/hide the Moore outputs inside the states:	select “Moore Outputs”. Each state defines its Moore outputs which are the values that are sent to the outputs of the FSM when the respective state is reached.
To show/hide the Mealy input conditions on the transitions:	select “Mealy Inputs”. Mealy inputs are asynchronous inputs to the FSM. They can trigger transitions from one state to another if the condition of the respective transition is satisfied.
To show/hide the Mealy outputs on the transitions:	select “Mealy Outputs”. Mealy outputs are outputs of the FSM that were sent when a transition is triggered. Thus, each transition can define the Moore outputs that are sent when it is triggered.
To show/hide the shadows of the states:	select “Shadows”.
To show/hide the grid on the working area:	select “Grid”.
To show/hide the window displaying the names of the inputs and outputs of a finite state machine:	select “IO View”.
To switch to the pan mode:	select “Pan View”. For more details, see the section called “Using the Pan mode”
To switch to the zoom mode:	select “Zoom”. For more details, see the section called “Using the Zoom mode”
To zoom the view in:	select “zoom in”. The current zoom value is shown in the leftmost part of the status bar.
To zoom the view out:	select “zoom out”. The current zoom value is shown in the leftmost part of the status bar.
To set the zoom to the original value (100%):	select “Zoom 100%”. The current zoom value is shown in the leftmost part of the status bar.

Machine

To modify the properties of the current FSM:	select “Edit”. It opens a dialog in which you can modify the properties of the FSM. For more details, see the section called “Creating and modifying a Finite State Machine”
To correct automatically the codes (or identifiers) of all states such that each state code is unique:	select “Auto correct State Codes”
To switch to the simulation mode:	select “Simulate”. For more details, see the section called “Using the Simulate mode”andthe section called “Simulating a FSM”
To perform an integrity check on the current FSM:	select “Integrity Check”. For more details, see the section called “Checking the integrity of a FSM”

State

To switch to the "add state" mode and create a new state:	select “New”. For more details, see the section called “Using the Add State mode” and the section called “Adding and modifying states”
To modify the properties of the currently selected state:	select “Edit”. It opens a dialog in which you can modify the properties of the state. For more details, see the section called “Adding and modifying states”
To define the currently selected state as the initial state of the FSM:	select “Set Initial State”.
To define the currently selected state as a final or non-final state:	select “Toggle Final State”.

Transition

To switch to the "add transition" mode and create a new transition:	select “New”. For more details, see the section called “Using the Add Transition mode” and the section called “Adding and modifying transitions”
To modify the properties of the currently selected transition:	select “Edit”. It opens a dialog in which you can modify the properties of the transition. For more details, see the section called “Adding and modifying transitions”
To straighten the currently selected transition:	select “Straighten”.

Creating and modifying a Finite State Machine

To create a new Finite State Machine (FSM), choose the menu item *File->New* . The "Machine properties" dialog opens and lets you specify the properties of the FSM. Here is an explanation of each item of the dialog.

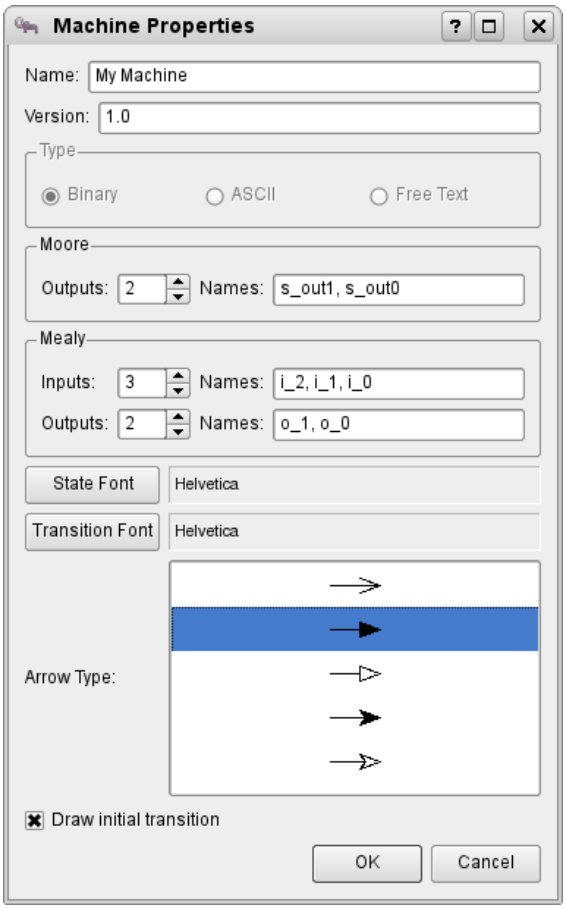


Table 2.1. Attributes of a Finite State Machine

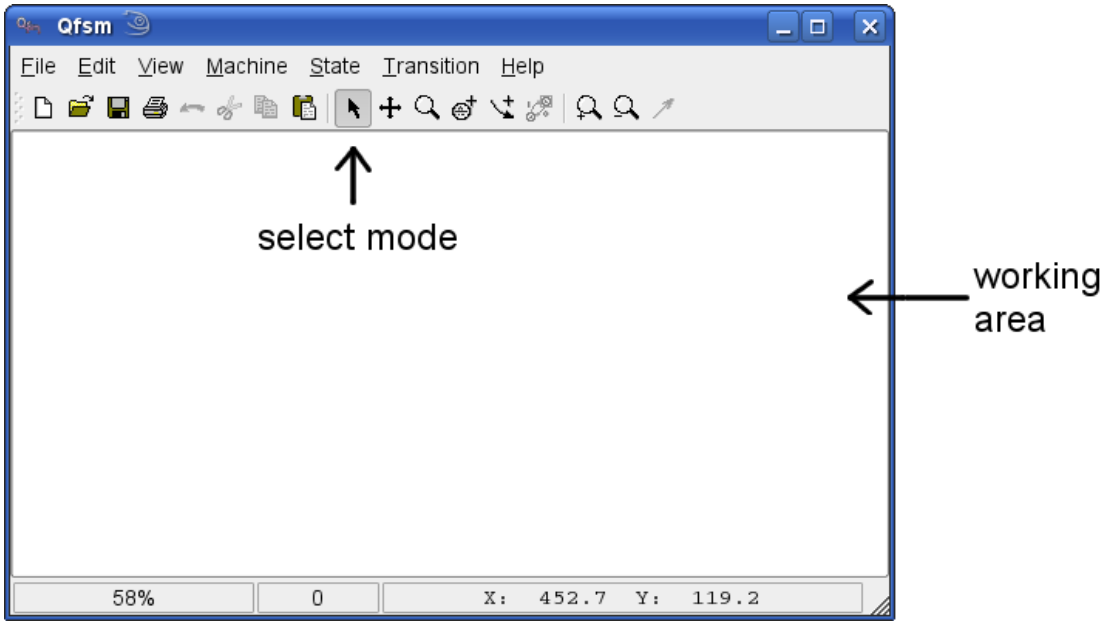
Name	Enter a name for the FSM. The name is used for example when exporting to the following formats: VHDL, Verilog HDL or Ragel.
Version	You can enter the version of the FSM. This is a free character string that is only used when printing the diagram.
Type	Select the type. The type attribute determines which type of information is processed by the FSM, i.e. the inputs, the outputs etc. Binary FSMs process zeros and ones at the inputs or outputs. This is the main type used for hardware design. ASCII FSMs process characters (i.e. letters, digits etc.). These characters are coded in ASCII format using 8 bits. This type of FSM can be used either for hardware design or to create string parsers. The "Free Text" type allows to specify inputs and outputs using any kind of character string of variable length. This type of FSM cannot be simulated afterwards because the input conditions won't be interpreted.

Moore outputs and Mealy inputs/outputs	If you are creating a "binary" FSM you can specify the number of bits of the moore output and mealy input/output and their respective names. The names are lists of character strings separated by commas. If you do not want to choose the names you can leave these fields blank and they will be automatically set.
Fonts	You can specify the font to use for the state names and for the input conditions and outputs displayed on the transitions.
Arrow Type	You can choose the type of arrow to use for drawing transitions.
Draw initial transition	You can tick or not this option if you want to draw or not the initial transition or if you want to reset or start the transition.

When you want to modify the properties of an existing FSM you can select *Machine->Edit* from the main menu and the same dialog box displays. As soon as you click OK the changes take effect.

Using the Working Area

The working area denotes the area of the Qfsm window that shows the state diagram. Once you have created a new FSM you see a blank working area and you are in the *select*mode.



There are six different modes you can be in and which determine what happens when you click or drag the mouse inside the working area of Qfsm.

1. Select
2. Pan
3. Zoom
4. Add State
5. Add Transition

6. Simulate

Only one mode can be activated at a time. To change the mode, click on one of the icons in the middle of the toolbar.

When a diagram is larger the working area, you have the possibility to move the view of a diagram. To move the view of a diagram, click on the middle mouse button and drag the mouse pointer. As soon as you release the middle mouse button the application reverts to the selected mode.

Alternatively, you can select the respective menu entry or press the respective short cut. The active mode is indicated by a highlighted toolbar button. In some modes, the form of the mouse cursor also changes, e.g. a magnifier for the zoom mode.

Using the Select mode

In this mode you can select, move or modify graphical objects.

To **select a state or a transition**, click on it with the left mouse button.

To **select several states or transitions at the same time**, hold down the shift key. You can then apply further actions on selected graphical objects, i.e. copy or edit, by using the menu.

To **unselect all the selected graphical objects**, click on the background.

To **show the context menu for a state or a transition**, click with the right mouse button on the state or transition.

To **modify the properties of a state or a transition**, double-click on the state or transition. It opens a dialog in which you can modify the properties.

To **select multiple graphical objects**, draw a rectangle holding the left mouse button around the graphical objects you want to select.

To **move state or transitions**, select them and drag them to the desired position.

To **move the transition control points**, select the respective transition and then drag one of the control points. The transition control points are indicated by small red and green points when a transition is selected. The red points control the form of the transition, i.e. the bend. The green ones are used to attach them to a starting and end state.

Using the Pan mode

In this mode, you can move the view to a different part of your diagram when your diagram is larger than the working area of the window.

To move the view, drag the mouse pointer.

Using the Zoom mode

In this mode you can zoom in the view.

To zoom in the view, click with the left mouse button on the working area.

To zoom out, keep the CTRL key pressed at the same time you click.

Using the Add State mode

In this mode you can add states and specify their properties.

To create a new state, click with the left mouse button and fill the dialog with the properties. For more details, see the section called “Adding and modifying states”

Using the Add Transition mode

In this mode you can add transitions and specify their properties.

To add a transition:

1) click on a state and drag the mouse pointer to another state.

2) fill the dialog with the properties of the transition.

For more details, see the section called “Adding and modifying transitions”

Using the Simulate mode

In the simulate mode, you can test the behaviour of your state machine with respect to external input.

When you enter this state, the simulator dialog opens and all interaction with the state diagram is disabled until you close the dialog. For more details, see the section called “Simulating a FSM”

Adding and modifying states

To create a state:

1) first you need to:

- create a new FSM or;

- open an existing file.

To create a new FSM, select *File->New* (see the section called “Creating and modifying a Finite State Machine”)

To open an existing file, select *File->Open*

and choose the desired file.

2) Switch to the "add state" mode (see the section called “Using the Add State mode”).

3) Click at the position of the working area where you want the new state to be.

The following dialog opens.

To specify or modify the properties of the state, fill the "State properties" dialog:

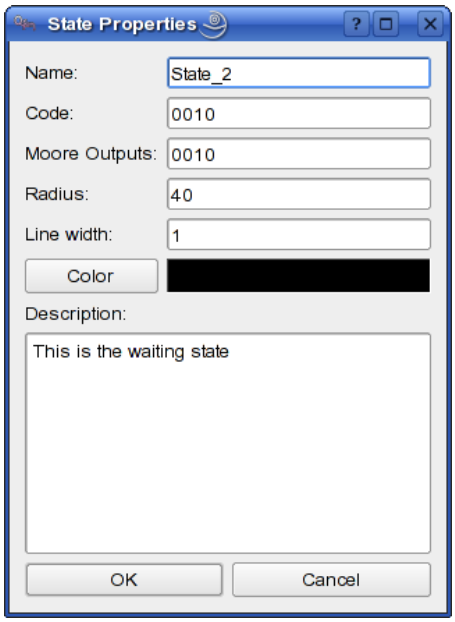


Table 2.2. Attributes of a state

Name	Enter a name for the state.
Code	Normally you do not need to fill this field. This is a unique identifier of the state you create, it is filled automatically.
Moore Outputs	Enter the outputs sent by the FSM. In "binary" FSMs this is a string of zeros and ones and in "ASCII" FSMs this is just one character.
Radius	Enter the radius of the drawn circle of the state, in pixels.
Line width	Enter the line width of the outline of the state.
Colour	If you want to change the colour of the outline of the state, click on the “Colour” button.
Description	You can enter a description of the state. This is only for documentation purposes.

To modify an existing state:

1) switch to the "select mode" (see the section called “Using the Select mode”).

2) double-click on the respective state

Or

2) select one state and select *State->Edit* from the main menu.

Adding and modifying transitions

To create a new transition:

1) you need first to:

- create a FSM or

- open an existing file.

To create a FSM, select *File->New* (see the section called “Creating and modifying a Finite State Machine”)

To open an existing file, select *File->Open* from the main menu and choose the desired file.

2) create at least one state (see the section called “Adding and modifying states”).

3) switch to the "add transition" mode (see the section called “Using the Add Transition mode”).

To create a transition from state A to state B, press and hold the left mouse button on state A and release it on state B.

To draw loops, i.e. transitions that go from one state to itself, press and release the mouse button on the same state.

The "Transition properties" dialog opens.To specify or modify the properties of the state, fill the dialog.

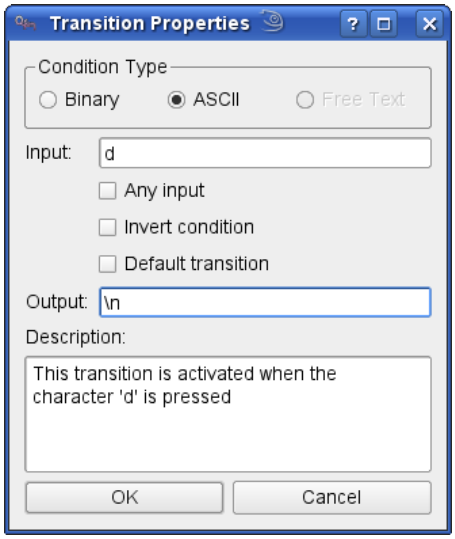


Table 2.3. Attributes of a transition

Condition Type	Choose the type of the condition. The type of the condition determines the format in which you enter the input condition in the next field.
Input	If the condition type is binary, enter a string of zeros and ones representing the Mealy input that should

	trigger this transition. You can also use the character 'x' meaning: "do not care". If the condition type is ASCII enter a character or specify an expression in a specific format which is explained in detail in the section called “Using input ASCII conditions”. For "free text" conditions enter any input character string. However, it has no logical meaning and won't be interpreted, for example when simulating the machine.
Output	Enter the Mealy output sent from the FSM when the transition is activated, i.e. the input condition is satisfied. Depending on the type of the condition the format is either a string of zeros and ones (binary) a character (ASCII) or any character string (free text). Note that in case of an ASCII character it can also be an escape sequence. For more details on escape sequences, see the section called “Using input ASCII conditions”
Description	You can enter a description of the transition. This is only for documentation purposes.

To modify the properties of an existing transition:

1) switch to the "select mode" (see the section called “Using the Select mode”).

2) double-click on the respective transition

Or

2) select one transition and select*Transition->Edit*from the main menu.

To change the bend of the transition as well as its start state and end state:

1) switch to the “Select mode”

2) click on the transition.

Four control points appear. You can drag them around with the left mouse button.

The green ones allow you to change the start and end state. With the red ones you can change the bend of the transition.

Using input ASCII conditions

When you create a transition of FSM that processes ASCII characters you have to enter an input condition. This condition can be a simple character, e.g. 'a', or several characters that are expressed by a special notation explained in the following.

Single character

This is the most simple form of condition. It contains one ASCII character, e.g. 'a' or 'z'.

Note that for special characters, e.g. '-' (minus sign) or the space character you need to use an escape sequence (see the section called “Escape sequences”).

Multiple characters

If you want the condition to contain multiple characters, i.e. 'a' or 'f' or '+' you just enter the string: 'af +'.Clearly, the order is not important.

Note that it is*not*possible to use a concatenation of characters as input condition, for example 'print' in order to recognize the word "print". To do this, you have to create a transition and a state for each character and build a chain with the respective characters.

Escape sequences

Special characters like the newline character need to be escaped, i.e. backslash + some character. The following table shows the recognized escape sequences.

Table 2.4. Recognized escape sequences

escape sequence	meaning
\t	tab
\n	newline
\r	carriage return
\s	space
\-	minus
\d	digit (0-9)

Note that the last escape sequence '\d' actually represents 10 characters.

Characters that are neither printable nor in the above table can be specified by '\0' (backslash zero) followed by their hexadecimal code. For example, '\0CF' would represent the ASCII character 207 (decimal).

Ranges

You can further specify ranges by using the minus sign. Thus, 'a-z' means one of the characters between 'a' and 'z' (including). Any character, even escaped ones, can be used as start or end point of a range.

Mixed formats

You can combine several conditions, each of them in one of the above mentioned notations, into one long condition by just concatenating them. Note that you must not separate them by any character, like white space or comma.

Here are some examples:

- A-F0-9
- +\-\d

- `\n\r\tXYZ`
- `xyz0-3\010A-Z`

Checking the integrity of a FSM

To check the integrity of a FSM, select *Machine->Integrity Check* from the main menu.

Warning

This may take a long time for larger FSMs, be careful not to interrupt the procedure.

The following tests are performed:

Unambigous Conditions	Checks if the FSM has transitions with conditions that are ambiguous, i.e. transitions that are activated simultaniously by the same input (in the same state). Note that ambiguous transitions are only allowed in*non-deterministic*FSMs, which are currently not supported by Qfsm.
Initial state	Checks if the FSM has an initial state.
Final state	Checks if the FSM has a final state.
No dead locks	Checks if the FSM has states where it can get of out, i.e. states with no transitions going out.
Completeness	Checks if for every possible input in every state there exists a transition that is activated.
States reachable	Checks if all the states of the FSM are reachable.
Final states reachable	Checks if all the final states of the FSM are reachable.
Transitions connected	Checks if all the transitions of the diagram are actually connected to a start end an end state. Note that sometimes a transition looks as if it is connected to a state but in fact the connection point is slightly away from it.

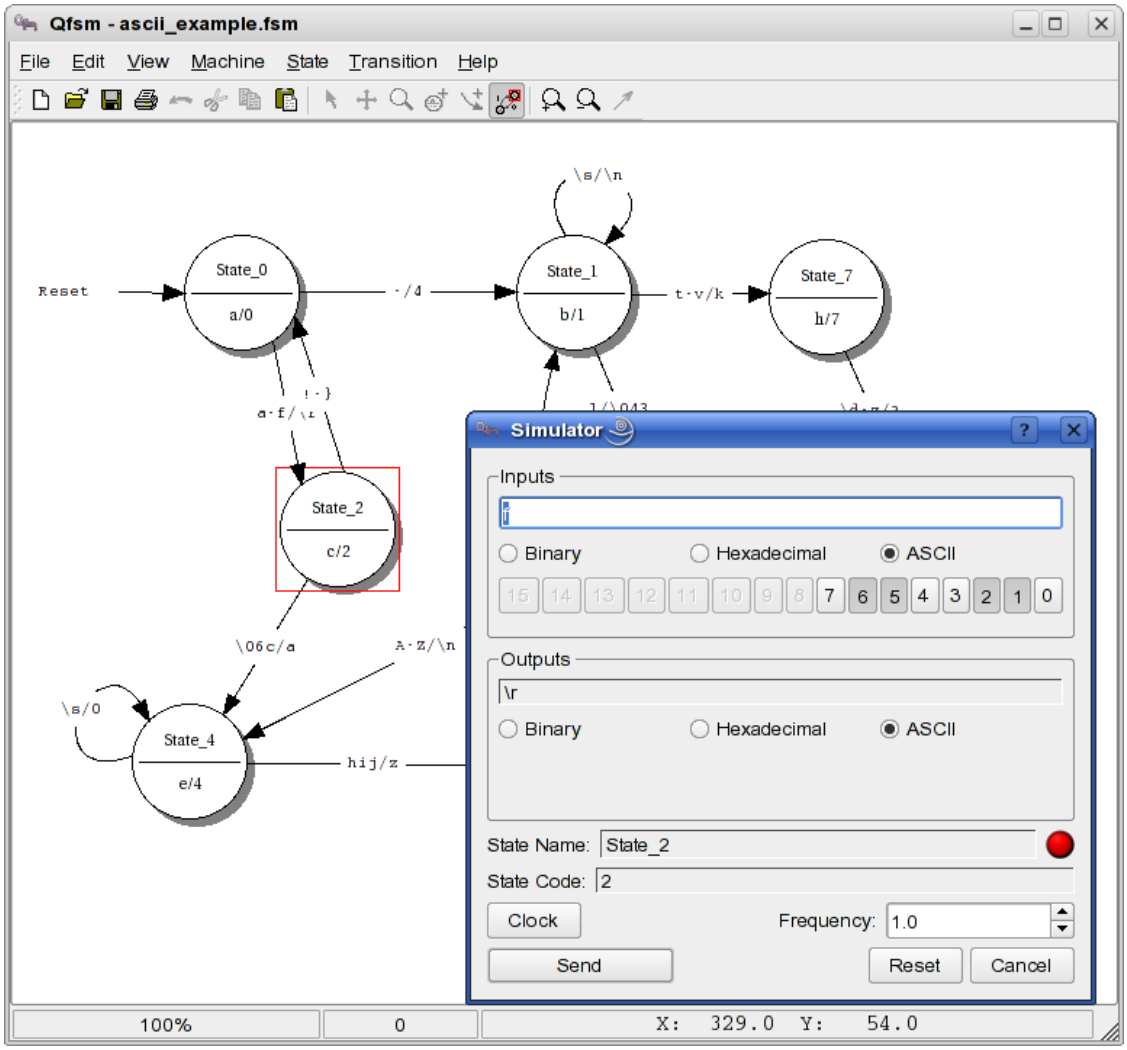
A log of the checks that have been performed is printed in the protocol field at the bottom of the window.

Simulating a FSM

Once you have created a FSM with some states and transitions you can simulate its behaviour with respect to varying input signals.

To start the simulation of an FSM, select *Machine->Simulate* from the main menu.

The "simulator" dialog opens allowing you to input data to the machine while displaying its current state and output.



Input	1) Enter the input data in the text field 2) select one of the formats: binary, hexadecimal or ASCII. 3) you can set or unset input bits using the buttons 0 to 15. When you choose the ASCII format you can also use escape sequences as detailed in the section called “Escape sequences”. However, you can only enter a single character. Thus, '\d' or ranges, for example, are not allowed.
Output	Choose the format of the current FSM, i.e. binary, hexadecimal or ASCII.
State Name and *State Code* fields	display the current state of the FSM. The red or green point next to it indicates if the FSM is in a final state or not, i.e. green for final state and red otherwise.

There are two modes to send input data:

- click on the *Send* button or, press the *Enter* key)

- click on the *Clock* button.

The FSM periodically sends the data in the input text field on the top of the dialog.

You can specify the frequency of the clock in the input field at the bottom right.

To exit the *Clock* Mode, click once again on the Clock button.

To reset the FSM, click on the *Reset* Button.

This sets the FSM to its initial state.

Exporting

There are several export functions in Qfsm.

To export, select *File->Export* in the main menu.

They can be divided into five categories:

1. state diagrams,
2. hardware description languages,
3. state tables,
4. code generation languages,
5. hardwarte test code,
6. I/O description,
7. vvvv Automata code
8. State Chart XML (SCXML)

Here are some explanations about the different export functions:

State diagrams

A state diagram as it is seen in the working area can be exported in various graphics formats. The formats that are supported are:

- Encapsulated Postscript (EPS)
- Scalable Vector Graphics (SVG)
- Portable Network Graphics (PNG)

Hardware description languages

Hardware description languages are high level descriptions that can be synthesized into integrated circuits like FPGAs using special software. The languages supported are the following:

1. AHDL

2. VHDL
3. Verilog HDL
4. KISS

Some of the export functions open a dialog allowing you to specify additional export options. However, they should be self-contained for the users having experience with hardware description languages.

State Tables

State tables can be exported in the formats: ASCII (plain text), Latex or HTML. State tables show for each possible state and input (here called*event*) the respective resulting states.

To change the options concerning the layout of the state table, a dialog allows you to change some options.

Include asynchronous output	Tick this option if you want the asynchronous outputs (Mealy outputs) to be printed in the table cells together with the resulting states.
Resolve inverted conditions	Tick this option if you want the inverted conditions to be printed using the inversion descriptor, e.g.`'NOT a'`, or without it, i.e. printing every character (or binary string) except the ones in the condition. In the case of`'NOT a'`this would be the two ranges`'\000-`` '`and`'b-\0FF'`. Which one is clearest depends on the respective FSM.
Orientation	Tick the different options in order to determine the orientation of the table, i.e. if current states represent the different rows of the table and the events the columns or vice-versa.

Code generation languages

For this type of output there is only one option, namely the *ragel* file format, and only ASCII FSMs can be exported. The resulting file serves as an input for the ragel state machine compiler. The ragel state machine compiler is a compiler that generates code from a high-level state machine description language. In this way, you can create parsers for example. For details refer to the ragel homepage [http://www.cs.queensu.ca/~thurston/ragel/].

A dialog allows you to create a so-called *action file*. That means, the ragel state machine specification is divided into two files. One that contains the state machine logic (which I will call FSM file here) and an action file that contains the action definitions and a framework calling the state machine. Thus, the action file actually includes the FSM file. The name of the action file is determined automatically by appending`'_action'`at the end of the file name.

Example: suppose you have created an ASCII FSM and you export it under the name `myFSM.rl`. If you check the option *'Create action file'* the action file will be created under the name`myFSM_actions.rl`.

Using ragel you can compile the action file: **ragel -C -o myFSM.c myFSM_actions.rl**

This will create a file, called myFSM.c with the C code of the FSM. It will contain a function:`int parse(char* string)`that parses an input string and returns 1 if the FSM accepts it, i.e. finishes in a final state, and 0 otherwise.

Hardware test code

The hardware test code is used together with the output of a hardware description export. The supported language for test code ist VHDL. The VHDL testbench export will create a set of files that enables a hardware simulation and debugging software to analyse the VHDL code exported by Qfsm.

The testbench code is separated into three files. A VHDL testbench file that contains routines to load test data and run a test. A VHDL package file with the required subroutines and a test vector file with all input and output signal values that are needed to check the function of the state machine.

I/O description

When exporting the I/O description of the state diagram a file containing the names of all the inputs and outputs of the FSM is created. The description is stored in a comma separated value (CSV) format, where the separator is a semicolon and the values are inside double quotes. The first value of each line is a textual description of the values in that line, i.e. "Inputs", "Mealy Outputs" and so on.

The following lines are written:

- "Inputs": the names of the (Mealy) inputs
- "Mealy Outputs": the names of the Mealy outputs
- "State/Output": the names of the Moore outputs
- Finally a line for each state containing its name and its Moore outputs

vvvv Automata code

A "free text" FSM can be exported to vvvv Automata code. This code can be used in conjunction with the tool vvvv [http://vvvv.org/tiki-index.php], a toolkit for real-time video synthesis. This textual format is a list of quadrupels, where each item/line describes a transition and is composed of:

- start state,
- event,
- end state,
- action.

When selecting this export function from the menu a non-modal dialog box is opened that displays the resulting code. This is automatically updated as the diagram is modified. An additional reset transition can optionally be added for each state by checking the box at the bottom of the dialog. There you can also specify the name of the reset event and the name of the respective action to be triggered.

State Chart XML

A "free text" FSM can be exported in the State Chart XML format as proposed by the W3C. The specification of SCXML can be found here [http://www.w3.org/TR/scxml/]. Note that each transition is triggered by an event. And the text that is entered as transition input is used as the name of the event.

The text that is entered as the transition output is interpreted as the name of an event that is to be sent when the transition is triggered. Thus, if the output linked to a transition is non-empty a corresponding *<send>* tag will be written.

Setting the options

To display the options dialog select *Edit->Options* from the main menu.

Changing language

To change the language:
1) select Edit>Options>General

2) select the language

3) click on OK

4) restart the application

The language change is now effective.

Changing the display

To change the display, select Edit>Options>Display from the main menu

Grid	You can choose the colour and size of the grid displayed on the working area. You can activate the grid via the main menu entry *View->Grid* .
Shadows	You can determine if shadows are to be drawn and the their colour.
Transitions	You can determine the appearance of input conditions and outputs drawn on top of the transitions.
Tooltips	You can determine if tooltips should be shown or not when moving the mouse pointer over a state or a transition.
Initial transition descriptor	Enter the text that is displayed next to the initial transition. Default: "Reset".
Inversion descriptor	Enter the text that is displayed before inverted transition conditions. Default: "NOT".
"Any input" descriptor	Enter the text that is displayed for transitions that are activated by any input. Default: "any".
Default transition descriptor	Enter the text that is displayed for default transitions. Default: "default".

Printing

To change the printing options, select Edit>Options>Printing from the main menu.

Print header	Tick this option if you want to print a header with the FSM name and version at the top of the diagram.

Chapter 3. Contact

If you have questions or suggestions concerning Qfsm feel free to contact me at:

<qfsm(at)duffner(dash)net(dot)de>

I'm also glad about any contribution you want to make to the project, e.g. code, bug fixes, documentation, packaging, testing etc.

Index

S

T

U

V

W

Z