VN - Visualization of Nondeterminism
User's Guide

Version 2.3

Mordechai (Moti) Ben-Ari
Department of Science Teaching
Weizmann Institute of Science
Rehovot 76100 Israel
http://stwww.weizmann.ac.il/g-cs/benari/

August 6, 2008

1 Introduction

VN is a tool for studying the behavior of nondeterministic finite automata (NDFA). It takes a description of an automaton and generates a nondeterministic program; the program can then be executed randomly or guided interactively. The automaton and the execution path are graphically displayed.

VN is written in Java for portability. It is based on other software tools:

- The interactive editor of JFLAP is used to create an automaton, which is saved as an XML file that is the input to VN. See: Susan H. Rodger and Thomas W. Finley. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett, 2006. The website is `http://jflap.org/`.
- The program generated from the automaton is written in PROMELA, the language of the SPIN model checker. See: Gerard J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004. The website is `http://spinroot.com`.
- The graphical description of the automaton and path are created in the DOT language and layed out by the DOT tool. Graphs in PNG format are created and are then displayed within VN. DOT is part of the GRAPHVIZ package; see: `http://graphviz.org/`.
- An ANSI C compiler for compiling a SPIN verifier; this is used to find an accepting computation in the automaton. For Windows you can use `MinGW` `http://mingw.org` or `Cygwin` `http://cygwin.com`.

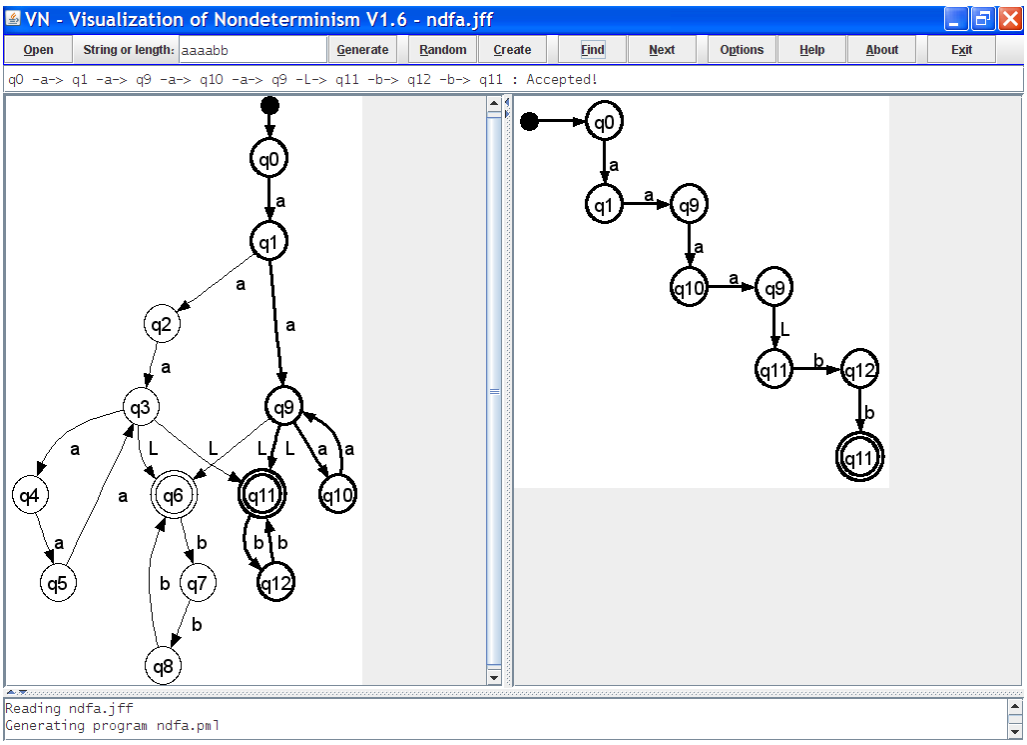
The VN software is copyrighted according the the GNU General Public License. See the files `copyright.txt` and `gpl.txt` in the archive.

The VN webpage is `http://stwww.weizmann.ac.il/g-cs/benari/vn/`.

The design of VN is explained in: M. Ben-Ari. *Principles of the Spin Model Checker*, Springer, 2008.

2 Installation and execution

- Install the Java SDK or JRE (`http://java.sun.com`). **VN needs Java 1.5 at least.**
- Simplified installation for Windows:
  - Download the VN installation file called `vn-N.exe`, where `N` is the version number. Execute the installation file.
  - Create a directory `c:\mingw` and download the C compiler archive `mingw.exe` into that directory. Execute this self-extracting file.
- Custom installation:
  - Download the VN distribution file called `vn-N.zip`, where `N` is the version number. Extract the files into a clean directory.
  - Install SPIN, DOT and JFLAP. (JFLAP optional and is only needed if you want to invoke it from within VN.) To simplify installation, a minimal set of files for running these tools in Windows is included in the archive `vn-bin.zip`.
  - Install an ANSI C compiler.
  - Modify the configuration file `config.cfg` to reflect the locations of the programs.
- The installation will create the following subdirectories: `docs` for the documentation; `vn` for the source files; `bin` for the libraries and executables for SPIN, DOT and JFLAP; `txts` for the text files (help, about and copyright); and `examples` for example programs.
- To run VN, execute the command `javaw -jar vn.jar`. An optional argument names the `jff` file from JFLAP to be opened initially. A batch file `run.bat` is supplied which contains this command.
- Configuration data for VN is given in the source file `Config.java`, as well as in the file `config.cfg`. The latter is reset to its default values if it is erased.
- To rebuild VN, execute `build.bat`, which will compile all the source files and create the file `vn.jar` with the manifest file.



3 Interacting with VN

The display of the VN program is divided into three scrollable panes. Messages from VN are displayed in the text area at the bottom of the screen. Graphs of automata are displayed in the left-hand pane and graphs of the paths are displayed in the right-hand pane. The size of the panes is adjustable and the small triangles on the dividers can be used to maximize a pane. Above the panes is another text area called the path area where execution paths of the automaton are displayed.

Interaction with VN is through a toolbar. All toolbar buttons have mnemonics (Alt-character).

Open Brings up a file selector; select a `jff` file and the automaton described in the file will be displayed.

Edit This invokes JFLAP on the current automaton. **The modified `jff` file must be re-opened in VN after it is saved in JFLAP.**

String or length Enter an input string for the automaton in this text field. See below on entering a length.

Generate Once an automaton has be opened and an input string entered in the text field, selecting Generate will create a program to execute the automaton.

Random This will execute the program for the automaton with a random resolution of nondeterminism. The sequence of states will appear in the text area below the toolbar, along with an indication if the input string was accepted or rejected.

Create This will execute the program for the automaton, resolving nondeterminism interactively. Determinisitic choices will be made automatically; if a choice cannot be taken, it will be displayed in square brackets and cannot be selected. Quit terminates the execution. Keyboard shortcuts: Tab moves between buttons and Space or Enter selects the highlighted button.

Find This searches for an accepting computation of the automaton.

Next This searches for the next accepting computation of the automaton. When no more accepting computations exist, the number of accepting computations is displayed in the path area.

DFA See below.

Options Displays a diaglog for choosing the size of the graphs (small, medium, large) and whether the highlighting of the paths will be in color or bold. When you select OK the changes will be saved, along with the directory of the current open file. (The options Interactive and Choose concern generating interactive programs as discussed below.)

Help Displays a short help file.

About Displays copyright information about the software. The version number appears in the title bar.

Exit Exit the application.

Once a computation has been found , the path itself is displayed in the right-hand pane and highlighted in the automaton in the left-hand pane. The difference is that the "path" includes multiple visits to the same state, while the highlighted path will, of course, include just one state for all occurences.

Searching for all inputs that are accepted

If you an integer value in the text field String or length, each selection of Next searches for an accepting computation for *any* input string of this length. When no more accepting computations are found, the set of strings that are accepted is displayed in path area. For the automaton `ndfa.jff` supplied in the archive and with input length six, there are seven accepting computations for the four inputs: `aaaaaa, aaaabb, aaabbb, aabbbb`.

Equivalence classes of deterministic finite automata

If the automaton is deterministic, it is possible to obtain the partition of a set of input strings of a given length into the equivalence classes associated with each state.[1] Enter a length in the text field and select Generate. Now select DFA repeatedly; for each state, starting from the first state, the set of input strings "accepted" by that state is displayed. When DFA has been selected for all states, the set of equivalence classes is displayed in the right-hand pane.[2]

For the DFA in the `examples` directory, the equivalence classes of length 4 are:

```
q0: [bbaa, baba, abba, aaaa]
q1: [bbab, babb, abbb, aaab]
q2: [bbba, baaa, abaa, aaba]
q3: [bbbb, baab, abab, aabb]
```

Generating interactive programs

VN can generate standalone interactive programs from a FA. Select a language from Options/Interactive: PROMELA, JAVA, or JELIOT for the Jeliot program animation system (`http://cs.joensuu.fi/jeliot/`). The Jeliot version is in Java, but contains non-standard constructs that simplify the animation.

Nondeterminism makes sense only for PROMELA programs; Java programs generated for NDFAs resolve nondeterminism in a fixed order. Select Options/Choose NDFA to generate a program whose input is a *transition number* rather than an input symbol. As can be seen from the generated source code, the transitions are given in lexicographic order.

This feature can also be invoked from the command line:

```
java -jar vn.jar file-name [promela|java|jeliot] [choose]
```

[1] This functionality was inspired by ProofChecker:
`http://research.csc.ncsu.edu/accessibility/ProofChecker/index.html`.

[2] Generate can be selected at any time to reset to the first state.

4 Files

The different phases of processing VN communicate through files. Here is a list of the extensions of these files:

`jff` JFLAP XML description of an automaton

`pml` Promela source generated from the automaton

`pth` Path resulting from running SPIN on the `pml` file

`dot` Graphics file describing automaton and path

`png` PNG graphics file after layout by DOT

5 Software structure

`VN` is the main class and contains declarations of the GUI components and the event handler for all the buttons.

`Config` contains constants and properties.

`Options` displays a dialog for changing the size and highlight of the graphs.

`DisplayFile` displays the files for About and Help in a `JFrame`.

`JFFFileFilter` is used with a `JFileChooser` to open a `jff` file.

`ImagePanel` displays the PNG files.

`GenerateSpin` generates the PROMELA program from the automaton.

`ReadXML` reads the XML description of the automaton from the `jff` file and stores the data in `states` and `transitions`. These types are defined in classes `State` and `Transition`.

`ReadPath` reads the path data printed by the PROMELA program and stores it in `pathStates` and `pathTransitions`.

`WriteGraph` writes the automaton and paths in DOT format and forks a process to run DOT to layout the graph.

`RunSpin` forks a process to execute SPIN. In interactive mode, it reads the PROMELA program so that the `JOptionPane` used for selecting among nondeterministic choices can display actual source code. For Find and Next processes are forked to run the C compiler and `pan`. For Next, `pan` is run with the `-c` argument.

A Release notes

2.3 As of Version 6.3, JFLAP as distributed can be invoked from another program so a modified version is no longer needed.

2.2 Generation of interactive programs.

2.1 Self-installing archive and relative path names.

2.0 Invocation of JFLAP.

1.8 DFA function added. Improved error messages.

1.7 In interactive mode (Create) the incremental path is displayed graphically as well as in the path area.

1.6 The input length can be entered instead of a string; VN searches for all inputs of this length that are accepted. The user interface has been slightly modified to support this feature.

1.5 Next is implemented. The paths are now shown automatically. The Show command can be restored by changing a constant in `Config.java`.

1.4 Configuration file added. Bug fix in display of path.

1.3 Improvement in generated PROMELA program's treatment of `timeout`.

1.2 Added Find.

1.1 Bug fixes and Options.

1.0 Initial public release.