# JUMBL Cheat Sheet

26th September 2005

This document contains some basic information about using the JUMBL usage modeling tool.

## 1 Help

### 1.1 On-Line Help

The JUMBL has an extensive on-line help system. To get a list of all of the supported JUMBL commands type:

```
jumbl
```

This will generate a list of all of the JUMBL commands:

```
UNIX / Linux JUMBL Wrapper Options:
  --mem=[num] Increase the memory limit for the virtual machine to

  --time      Time the execution of the specified action.

JUMBL: Specify command.
The following commands are understood:
      About
      Aggregate
      Analyze
      Check
      CraftTest
      Flatten
      GenTest
      ManageTest
      Options
      Prune
      Random
      RecordResults
      Write
```

To get help about a specific command X from the above list, type `jumbl X`. For example, to get more information about the Check command type:

```
jumbl Check
```

This gives you information about the JUMBL model checking functionality:

```
Usage: Check [models...]

Check the structure of the specified models. This detects states
which cannot be reached from the source and states from which the
sink cannot be reached. It also reports some statistics about the
model.

  --gui ..................Use a GUI to select the files to check and
                           also to display the results. When using the
                          GUI, model references are not checked.
  -h or
  --help .................Describe this command.
  -O [path] or
  --object_path=[path]...Specify the object search path.
```

### 1.2 User's Manuals

The JUMBL also has an extensive user's manual. The manual is available in `/ash/homes/cs340/jumbl_user.pdf`.

Additional information about the JUMBL is available at `http://www.cs.utk.edu/sqrl/esp/jumbl4/index.html`.

## 2 Checking Model Structure

After creating your model in TML, you will need to check its structure to ensure that it is a valid model. A valid model has the following characteristics:

- It has a single start state called the *source*.
- It has a single ending state called the *sink*.
- All states in the model are reachable from the source, i.e., there are no *unreachable* states.
- The sink can be reached from all states, i.e., there are no *trap* states.

To check the model cc1a.tml, type the following command:

```
jumbl Check cc1a.tml
```

# 3 Analyzing the Model

Once you have checked the model to ensure that it has the proper structure, the JUMBL can be used to perform a statistical analysis of the usage behavior described by the model. To analyze the model cc1a.tml, enter the following command:

```
jumbl Analyze cc1a.tml
```

This produces an analysis report in the file cc1a_ma.html, which can be read in a web browser. The model analysis results are used to review the model to ensure that it correctly specifies the use of the software being tested. The basic model analysis results are as follows.

- Expected Test Case Length - The length of a typical test case.
- Undirected Graph Cyclomatic Number - A measure of the complexity of the model.
- arc Occupancy - The amount of time, in the long run, that you will spend testing a state.
- Node Probability of Occurrence - The probability of a state appearing in a random test case.
- Node Mean Occurrence - The average number of times a state will appear in a random test case.
- Node Mean First Passage - The number of random test cases you will need to run, on average, before testing a state for the first time.
- Arc Occupancy - The amount of time, in the long run, that you will spend testing an arc.
- Arc Probability of Occurrence - The probability of an arc appearing in a random test case.
- Arc Mean Occurrence - The average number of times an arc will appear in a random test case.
- Arc Mean First Passage - The number of random test cases you will need to run, on average, before testing an arc for the first time.

The table headings in the HTML model analysis report are all live links. Clicking on them will provide you will more information about the various analysis results.

# 4 Generating Test Cases

Once the model is determined to correctly describe the use of the software under test, test cases can be generated from the model. Two possible ways to generate tests are randomly and via the Chinese Post man's algorithm (the JUMBL supports other test generation methods. Read the manual for more information about test generation).

### 4.1 Random Test Cases

To generate 10 test cases from the cc1a.tml usage model via random walks of the model, type the command:

```
jumbl GenTest -n 10 cc1a.tml
```

This generates 10 test random cases and saves them in a test record called CycleComputer.str (since the model in cc1a.tml is named CycleComputer). To export the test cases to files usable by a tester, type the command:

```
jumbl ManageTest export CycleComputer.str
```

The JUMBL will export each test case into a separate file.

### 4.2 Model Coverage Test Cases

To generate a non-random set of test cases that cover all arcs in the model with a minimum number of test steps, issue the following command:

```
jumbl GenTest -m cc1a.tml
```

## 5 Entering Test Results

Once you have run a test case, the results of running that test case can be added to the test record. To add test case two to the test record for usage model cc1a.tml as passed (i.e., the test was executed with no failures), enter the following command:

```
jumbl RecordResults CycleComputer.str@2
```

To add test case three to the test record as failed, with test failures observed on test steps one and 3, enter the following command:

```
jumbl RecordResults CycleComputer.str@3,1,3
```

To print out a summary of the contents of a test record, enter the following command:

```
jumbl ManageTest list CycleComputer.str
```

## 6 Analyzing Test Results

Once tests have been executed and the results of running the test cases have been stored in the JUMBL test record, the testing can be analyzed. To analyze a test record issue the following command:

```
jumbl Analyze CycleComputer.str
```

This will create a test analysis report in the file CycleComputer_ta.html. Some of the basic results in this file are:

Nodes Executed - The number of states covered in the executed test cases.

Arcs Executed - The number of arcs covered in the executed test cases.

Arc Reliability - The estimated probability of executing an arc in a test cases successfully.

Single Event Reliability - The estimated probability that a randomly selected arc can be executed successfully in a test case.

Single Use Reliability - The estimated probability of executing a test case without any failures.

Optimum Reliability - The estimated reliability if all **generated** test cases were executed successfully.

Relative Kullback Discriminant - A measure of how close the performed testing matches the software use as described in the usage model. The lower the number, the closer testing matches the expected use of the software.

As with the model analysis report, the table headings in the test analysis report are all live links. Clicking on them will provide you will more information about the various analysis results.