Writing a test file for MMAlpha

Patrice Quinton

Version of December 26, 2008

1 Introduction

Testing programs is recommended by everybody... and done only by a few people. MMAlpha is no exception...

The purpose of this document is to explain what policy is to be followed when one wants to develop a new package for MMAlpha.

A companion document, located in `$MMALPHA/doc/Tests` explains how to test the MMAlpha.

In a few words, here are the recommendations:

1. All packages must be provided with a Mathematica test file. Let `myPack.m` by a brand new MMAlpha package (the same policy applies to C program). Then, a MMAlpha program, `TestmyPack.m` must be written, and placed in the directory `$MMAlpha/tests` in a new directory `TestmyPack`. If you follow these recommendations, your test program will be called when evaluating the MMAlpha expression `tests["myPack"]`.

2. Your test program, `TextmyPack.m` must return a value `True` if all tests have been passed, `False` otherwise.

3. Test your test program! A good policy is to incorporate all examples you have used when developing your code. Remember that your tests are going to be run among several thousands, so make sure that your tests are correct.

In what follows, we present quickly facilities provided for testing by MMAlpha, and we give an example.

2 Functions for testing

Two functions are used: `tests`, and `testFunction`. These functions were improved in the V2 version of MMAlpha.

2.1 The `tests` function

This function is part of the `Alpha.m` package. Evaluating `tests[]` runs *all test files which are in the test directory*. This may take a few hours... We use it only to check a new version[1].

Evaluating `tests["myPack"]` runs the test program for package `myPack`. You can therefore run tests for various MMAlpha packages, for example, `tests["Alpha"]` (try it!). Notice that Mathematica error messages may happen during the test; this does not mean that the test has failed. Indeed, some test programs test that MMAlpha detects abnormal situations.

2.2 The `testFunction`

A function, called, `testFunction`, is provided in the `Alpha.m` package, as a means to organize tests in a systematic way. Using this function is recommended. Its operation is as follows.

```
testFunction[exp, result, testID]
```

compares the evaluation of `exp` to `result`. It returns `True` if they are equal, `False` and emits a message if they are different. `testID` is a string which identifies the test.

This function can be used in many different ways. For example, here is a test of the `Alpha.m` package, which checks the MMAlpha parser:

```
testFunction[load["Test1.alpha"];$result,<<"RTest1","Alpha 1"]
```

The evaluation of this expression loads the file `Test1.alpha` which is supposed to be in the test directory, and compares the value of `$result`, the MMAlpha variable which contains the current AST, with the content of the file `RTest1`, in the same directory. This test is identified as `"Alpha 1"`.

[1] This may change...

3 An example

Fig. 1 gives an example of test file.

```
Module[{dir, res1, testresult},

  dir = Directory[];  (* Save current directory *)

  (* Go in test directory *)
  SetDirectory[Environment["MMALPHA"]<>"/tests/TestAlpha/"];
  Print["Test for Alpha.m"];

  (* Build a list of boolean values *)
  res1=
  {
    testFunction[load["Test1.alpha"];$result,<<"RTest1","Alpha 1"],
    testFunction[load[22],Null,"Alpha 2"],
    testFunction[Check[show[],$Failed],Null,"Alpha 3"],
    testFunction[show[22],Null,"Alpha 4"],
    (* ... *)
    ReadList["MV1.c",Record,RecordSeparators->{}]===
      ReadList["MV1.c",Record,RecordSeparators->{}],True,"writeC 1"],
    DeleteFile["MV1.c"];True
  }

  (* And the list *)
  testResult = Apply[And,res1];

  (* Back to initial dir *)
  SetDirectory[dir];

  (* Diagnostic *)
  If[ testResult,
    Print["**** Test OK for Alpha.m "],
    Print["**** Something was wrong for Alpha.m"]];

  (* Do not forget to return the result *)
  testResult
] (* End module *)
```

Figure 1: Example of test, adapted from the file `Alpha.m`