Basic Automatic Generation of White Box Test Inputs from Ada Source Code using Mika

Midoan Software Engineering Solutions Ltd.

October 2010

Abstract

Our main objective, in this short white paper, is to present the basic usage of our tool for automatic test inputs generation from source code. In particular, we demonstrate Mika's ability to generate test inputs to help achieve full white box coverage of the code under test.

1 Introduction

At the most basic level, Mika constructs test inputs that will achieve full white box coverage of a given Ada subprogram. Mika achieves this by analysing the code under consideration and does not generate any redundant test inputs: every constructed new test input improves the level of coverage achieved so far. However, whilst this is often the case, and to keep the analysis tractable, the test inputs generated are not guaranteed to represent a minimal test suite for the coverage concerned (this is particularly relevant for MC/DC coverage).

In the presence of infeasible paths, or infeasible logical conditions combinations, 100% coverage may not be achieved for the chosen coverage criterion. In this case, the test inputs generated by Mika will achieve the highest possible coverage and report the uncovered features (branch, decision, or condition) for further investigation.

Mika can generate test inputs for branch, decision or masking MC/DC coverage [1]. This can be performed for the given subprogram only, for the given subprogram and its subsequent local subprograms call tree, or for the given subprogram and its entire subsequent subprograms call tree [1]. Finally, the user may specify whether the initialisations performed during the elaboration phase can be ignored (i.e. overwritten) by the generated test inputs or not [1]. These basic features of Mika will be illustrated in subsequent white papers.

For now, we illustrate test inputs generation to achieve branch coverage of a given subprogram only, where the elaboration context is ignored.

2 The Example

This short example manipulates dates of type `date_t`.

```
package array_date is
  type name_t is (mon, tue, wed, thu, fri, sat, sun);
  type year_t is range 1900 .. 3000;
  type day_t is range 1..31;
  type month_t is (jan, feb, mar, apr, may, jun, jul, aug, sep,
                   oct, nov, dec);
  type date_t is
    record
      name : name_t;
      day : day_t;
      month : month_t;
      year : year_t;
    end record;
  type index is range 1..50;
  type list is array(index) of date_t;
  procedure InsertionSort(L: in out list);
  procedure Mika_Test_Point(Test_number : in Integer);
end array_date;
```

In this white paper we are only interested in one of the procedure: given its *in* parameter the procedure `tomorrow` returns the next date in its *out* parameter. A typical test case based on black box test cases generation is: input `Monday 20th October 1974`, expected output `Tuesday 21st October 1974`.

```
package body array_date is

function is_leap(y : year_t) return boolean is
begin
  return  (y mod 4 = 0 and y mod 100 /= 0) or y mod 400 = 0;
end is_leap;

procedure tomorrow(date: in date_t; next_date: out date_t) is
begin
  next_date := date;
  if date.name = name_t'last then
    next_date.name := name_t'first;
  else
    next_date.name := name_t'succ(date.name);
  end if;
  if (date.month = dec and date.day = 31) then
    next_date.day := 1;
    next_date.month := jan;
```

```
    next_date.year := date.year + 1;
  elsif (date.day = 28 and date.month = feb) then
    if is_leap(next_date.year) then
      next_date.day := 29;
    else
      next_date.day := 1;
      next_date.month := mar;
    end if;
  elsif (date.day = 31 or (date.day = 29 and date.month = feb) or
         (date.day = 30 and (date.month = apr or date.month = jun or
                             date.month = sep or date.month = nov)
         )
        ) then
    next_date.day := 1;
    next_date.month := month_t'succ(date.month);
  else
    next_date.day := date.day+1;
  end if;
end tomorrow;

function preceeds(d1, d2 : date_t) return boolean
is
  ... -- not relevant
end preceeds;

procedure InsertionSort(L: in out list) is
  ... -- not relevant
end InsertionSort;

procedure Mika_Test_Point(Test_number : in Integer) is separate;

begin
  null;
end array_date;
```

Note the necessity of introducing the `Mika_Test_Point` procedure to allow for automatic test execution within Mika as detailed in Mika's user manual[1]; the ability to execute the generated test inputs from within Mika is optional and not relevant to this white paper.

```
separate (array_date)
procedure Mika_Test_Point(Test_number : in Integer) is
begin
  Null;
end Mika_Test_Point;
```

3 Generating White Box Test Inputs Automatically

Using Mika, we can generate a set of test inputs that will by construction traverse every branch in the `tomorrow` procedure during execution. This is a two step process that can be performed from the GUI or from the command line.

The initial parsing step allows the subsequent generation of test inputs for all the subprogram in the package under consideration. The second step generates test inputs for a specific subprogram.

For example, below are the test inputs generated for the `tomorrow` subprogram given previously. As indicated in the generated report, these tests achieve 100% branch coverage when executed and can form the basis for test cases and be used in subsequent regression testing.

```
------------------------------------------------------------
--                MIKA TEST DATA GENERATOR                --
-- Copyright Midoan Software Engineering Solutions Ltd. --
--                 http://www.midoan.com/                 --
------------------------------------------------------------
-- Original Directory:  XXXX
-- Target Directory:  XXXX/array_date_mika/tomorrow_8_XXXX/
-- Package:    array_date
-- Subprogram: tomorrow
-- Strategy:   branch
-- Context:    ignored
-- Coverage Depth:  Subprogram Only
-- Time Stamp: XXXX
------------------------------------------------------------
TEST NUMBER 1
PREDICTED COVERED BRANCHES :
Number 2, outcome true, in file array_date.adb, on line 16 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5

BRANCHES COVERAGE INCREASED BY BRANCHES:
Number 2, outcome true, in file array_date.adb, on line 16 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5

CONSTRUCTED TEST INPUT
date = (sun, 31, dec, 2800)

CODE BEHAVIOUR
next_date = (mon, 1, jan, 2801)
------------------------------------------------------------
TEST NUMBER 2
PREDICTED COVERED BRANCHES :
Number 5, outcome false, in file array_date.adb, on line 27 and column 8
```

```
Number 3, outcome false, in file array_date.adb, on line 20 and column 8
Number 2, outcome false, in file array_date.adb, on line 16 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5

BRANCHES COVERAGE INCREASED BY BRANCHES:
Number 5, outcome false, in file array_date.adb, on line 27 and column 8
Number 3, outcome false, in file array_date.adb, on line 20 and column 8
Number 2, outcome false, in file array_date.adb, on line 16 and column 5

CONSTRUCTED TEST INPUT
date = (sun, 28, dec, 2223)

CODE BEHAVIOUR
next_date = (mon, 29, dec, 2223)
------------------------------------------------------------
TEST NUMBER 3
PREDICTED COVERED BRANCHES :
Number 5, outcome true, in file array_date.adb, on line 27 and column 8
Number 3, outcome false, in file array_date.adb, on line 20 and column 8
Number 2, outcome false, in file array_date.adb, on line 16 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5

BRANCHES COVERAGE INCREASED BY BRANCHES:
Number 5, outcome true, in file array_date.adb, on line 27 and column 8

CONSTRUCTED TEST INPUT
date = (sun, 31, feb, 2842)

CODE BEHAVIOUR
next_date = (mon, 1, mar, 2842)
------------------------------------------------------------
TEST NUMBER 4
PREDICTED COVERED BRANCHES :
Number 4, outcome true, in file array_date.adb, on line 21 and column 7
Number 3, outcome true, in file array_date.adb, on line 20 and column 8
Number 2, outcome false, in file array_date.adb, on line 16 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5

BRANCHES COVERAGE INCREASED BY BRANCHES:
Number 4, outcome true, in file array_date.adb, on line 21 and column 7
Number 3, outcome true, in file array_date.adb, on line 20 and column 8

CONSTRUCTED TEST INPUT
date = (sun, 28, feb, 2204)

CODE BEHAVIOUR
```

```
next_date = (mon, 29, feb, 2204)
------------------------------------------------------------
TEST NUMBER 5
PREDICTED COVERED BRANCHES :
Number 4, outcome false, in file array_date.adb, on line 21 and column 7
Number 3, outcome true, in file array_date.adb, on line 20 and column 8
Number 2, outcome false, in file array_date.adb, on line 16 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5

BRANCHES COVERAGE INCREASED BY BRANCHES:
Number 4, outcome false, in file array_date.adb, on line 21 and column 7

CONSTRUCTED TEST INPUT
date = (sun, 28, feb, 1900)

CODE BEHAVIOUR
next_date = (mon, 1, mar, 1900)
------------------------------------------------------------
TEST NUMBER 6
PREDICTED COVERED BRANCHES :
Number 2, outcome true, in file array_date.adb, on line 16 and column 5
Number 1, outcome false, in file array_date.adb, on line 11 and column 5

BRANCHES COVERAGE INCREASED BY BRANCHES:
Number 1, outcome false, in file array_date.adb, on line 11 and column 5

CONSTRUCTED TEST INPUT
date = (mon, 31, dec, 2863)

CODE BEHAVIOUR
next_date = (tue, 1, jan, 2864)
------------------------------------------------------------
FINAL REPORT
   100% branch OVERALL COVERAGE PREDICTED.
6 tests generated (paths followed)
0 paths fully attempted but for which tests could not be generated
----------------------END OF REPORT-----------------------
```

The 6 test inputs generated will, during execution, fully exercise the 5 branches, for true and false outcomes, contained in the `tomorrow` subprogram. Mika indicates the branches covered by each test inputs as well a justification, by way of the newly covered branches, for the necessity of each test inputs.

If a branch cannot be covered, Mika indicates so. For example we if modify the subprogram `is_leap` above to:

```
function is_leap(y : year_t) return boolean is
```

```
begin
  return  true;
end is_leap;
```

Then the final report, in addition to the normal generated test inputs, contains:

```
FINAL REPORT
90% branch OVERALL COVERAGE PREDICTED.
9 BRANCHES PREDICTED COVERED:
Number 1, outcome false, in file array_date.adb, on line 11 and column 5
Number 1, outcome true, in file array_date.adb, on line 11 and column 5
Number 2, outcome false, in file array_date.adb, on line 16 and column 5
Number 2, outcome true, in file array_date.adb, on line 16 and column 5
Number 3, outcome false, in file array_date.adb, on line 20 and column 8
Number 3, outcome true, in file array_date.adb, on line 20 and column 8
Number 4, outcome true, in file array_date.adb, on line 21 and column 7
Number 5, outcome false, in file array_date.adb, on line 27 and column 8
Number 5, outcome true, in file array_date.adb, on line 27 and column 8
1 BRANCHES PREDICTED REMAINING TO BE COVERED:
Number 4, outcome false, in file array_date.adb, on line 21 and column 7
5 tests generated (paths followed)
0 paths fully attempted but for which tests could not be generated
----------------------END OF REPORT-----------------------
```

Thus the report clearly indicates that, for the modified code, branch number 4 on line 21 and column 7 in the file `array_date.adb` (i.e. `if is_leap(next_date.year) then`) cannot be traversed during execution for the false outcome and that only 90% branch coverage can be achieved due to an infeasible path.

4 Conclusions

In this white paper we have illustrated the basic automatic test inputs generation functionality of Mika. Subsequent white papers address other aspects of Mika such as:

- other coverage criteria (notably masking MC/DC test inputs generation);
- the automatic execution of the test inputs generated;
- the support available from within Mika for the generation of test cases;
- regression testing;
- the integration of pre-existing test inputs or test cases with Mika.

As always do not hesitate to contact us with your queries via our website's [2] feedback mechanism at `http://www.midoan.com/request.html` which can also be accessed directly from within Mika's GUI.

5 Bibliography

References

[1] Midoan Software Engineering Ltd. Mika user manual. Technical report, 2010.

[2] Midoan Software Engineering Ltd. Midoan website. `http://www.midoan.com/`.