Automatic Testing of Student Programs using Token Patterns
運用單詞樣式以自動測試學生的電腦程序習作

Submitted to
Department of Computer Science
在 Partial Fulfillment of the Requirements
for the Degree of Master of Philosophy

by

Tang Chung Man
鄧頌民

June 2011
二零一一年六月


**Abstract**

Assessing students’ programming exercises manually can be tedious and error-prone. To relieve these problems, many universities have developed software systems to automatically assess students’ programs, usually by executing them to test their functional correctness, typically through simple matching of the program output with expected output. This approach requires that the precise and detailed form of the output be given in the specification of the exercise, because even a small and insignificant deviation (such as an extra whitespace) from the expected output will cause the matching to fail. An undesirable pedagogical consequence of this technical limitation is that it may unnecessarily confine the way of implementation, perhaps inhibit creativity, and distract students from the essentials of the exercise. To handle slightly deviated but acceptable program outputs, some automated systems employ tailored output comparison techniques that match certain kinds of output variants. However, existing techniques are mostly ad hoc and limited in capabilities. There is a need for more sophisticated techniques that can tolerate reasonably acceptable output variants. Towards this goal, we propose a novel approach, based on token patterns, that splits the output texts into smaller units for more fine-grained output comparison. As a result, the program output comparison can be more flexible and the accompanying undesirable side effects on teaching and learning can be reduced. An experimental prototype was initially built to evaluate our proposed approach. The prototype was then further integrated into a production system used in a programming course for empirical evaluation. The evaluation results are encouraging, showing that our proposed new approach is feasible and promising in improving student satisfaction in the use of the automated system, as well as producing better outcomes in terms of teaching and learning.
# Table of Contents

Abstract .......................................................................................................................... i

Acknowledgement .......................................................................................................... ii

Papers Published .............................................................................................................. iii

Table of Contents ........................................................................................................... iv

List of Figures .................................................................................................................. vii

List of Tables ................................................................................................................... ix

List of Sample Programming Exercises ......................................................................... x

List of Appendices .......................................................................................................... xii

1. Introduction .................................................................................................................. 1
   1.1 Background ................................................................................................................ 1
   1.2 Problems with manual assessment of student programs ........................................... 1
   1.3 Automated assessment of student programs .............................................................. 2
   1.4 Testing to determine program correctness ................................................................. 3
   1.5 Problems with existing methods of output comparison .............................................. 4
   1.6 Thesis organization .................................................................................................... 6

2. Literature Review ......................................................................................................... 8
   2.1 Supporting software tools for programming courses .................................................. 8
   2.2 Some automated program assessment systems ......................................................... 12
       2.2.1 TRY .................................................................................................................. 13
       2.2.2 ASSYST ............................................................................................................ 13
       2.2.3 Ceilidh/CourseMarker ....................................................................................... 14
       2.2.4 BOSS ............................................................................................................... 16
       2.2.5 Curator/Web-CAT ............................................................................................ 17
       2.2.6 PASS ............................................................................................................... 18
       2.2.7 Online Judge .................................................................................................... 20
       2.2.8 Summary .......................................................................................................... 21
   2.3 Assessment of student programs ............................................................................... 22
       2.3.1 Static assessment ............................................................................................... 22
       2.3.2 Dynamic assessment ......................................................................................... 25
2.4 Strategies for output correctness determination ........................................ 27
  2.4.1 Basic character matching ................................................................. 29
  2.4.2 Character matching with filtering ...................................................... 30
  2.4.3 Prescribing highly detailed specifications .......................................... 30
  2.4.4 Instructor-provided stub .................................................................... 33
  2.4.5 Instructor-designed driver ................................................................. 34
  2.4.6 Unit testing framework ....................................................................... 35
  2.4.7 Use of regular expressions ................................................................. 36
  2.4.8 Use of parser tools ............................................................................. 38
  2.4.9 Summary ............................................................................................ 38

2.5 Problems with existing output comparison strategies ............................ 40

2.6 Properties of a good automated assessment approach ........................... 41

2.7 Other requirements for an improved approach ........................................ 45

3. Analysis of Programming Exercises and Output Variants ............... 47
  3.1 Exercises from textbooks and courses .................................................. 47
  3.2 Exercise analysis .................................................................................... 50
    3.2.1 An ordinary programming exercise ................................................. 50
    3.2.2 A sample auto-assessed programming exercise .............................. 51
    3.2.3 Unsatisfactory consequence of overly detailed specifications ...... 53
  3.3 Program output variants ........................................................................ 55
  3.4 A case study on output variants ............................................................ 56
    3.4.1 Subjectivity and objectivity in correctness determination .............. 57
    3.4.2 Types of admissible variants .......................................................... 58

4. Token Pattern Approach .......................................................................... 64
  4.1 Tokenization ......................................................................................... 65
  4.2 Tagging ................................................................................................. 66
    4.2.1 Matching rule hierarchy ................................................................. 66
    4.2.2 Matching rules ............................................................................... 67
  4.3 Comparison of tokens ........................................................................... 69
  4.4 Comparison involving groups of tokens ................................................. 71
  4.5 Summary ............................................................................................... 74

5. Experimental Prototype .......................................................................... 76
  5.1 Functions of the prototype .................................................................... 76
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Implementation</td>
<td>78</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Tokenization</td>
<td>80</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Tagging</td>
<td>81</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Comparison</td>
<td>83</td>
</tr>
<tr>
<td>5.2.4</td>
<td>User interfaces</td>
<td>85</td>
</tr>
<tr>
<td>5.3</td>
<td>Preliminary findings</td>
<td>90</td>
</tr>
<tr>
<td>5.4</td>
<td>Adoption of programming exercises</td>
<td>94</td>
</tr>
<tr>
<td>6.</td>
<td><strong>Empirical Evaluation</strong></td>
<td>101</td>
</tr>
<tr>
<td>6.1</td>
<td>Research questions</td>
<td>102</td>
</tr>
<tr>
<td>6.2</td>
<td>PASS-ToP: An integrated APAS</td>
<td>102</td>
</tr>
<tr>
<td>6.3</td>
<td>Programming exercises</td>
<td>106</td>
</tr>
<tr>
<td>6.4</td>
<td>Test case design</td>
<td>110</td>
</tr>
<tr>
<td>6.5</td>
<td>Conducting the experiment</td>
<td>113</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Participants</td>
<td>113</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Schedule</td>
<td>114</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Logistics of programming problems and submissions</td>
<td>115</td>
</tr>
<tr>
<td>6.6</td>
<td>Analysis</td>
<td>116</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Type I and Type II errors</td>
<td>118</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Acceptance of admissible variants</td>
<td>120</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Program submission attempts</td>
<td>123</td>
</tr>
<tr>
<td>6.6.4</td>
<td>Feedback from students</td>
<td>125</td>
</tr>
<tr>
<td>6.6.5</td>
<td>Feedback from instructors</td>
<td>129</td>
</tr>
<tr>
<td>6.7</td>
<td>Summary of results</td>
<td>130</td>
</tr>
<tr>
<td>7.</td>
<td><strong>Conclusion</strong></td>
<td>132</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary and contributions</td>
<td>132</td>
</tr>
<tr>
<td>7.2</td>
<td>Limitations</td>
<td>134</td>
</tr>
<tr>
<td>7.3</td>
<td>Future work</td>
<td>135</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>140</td>
</tr>
<tr>
<td>Web References</td>
<td></td>
<td>143</td>
</tr>
<tr>
<td>Appendices</td>
<td></td>
<td>144</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1. Jeliot 3: A Program Visualization System (Jeliot 3, 2011). ............................................ 10
Figure 2. The assessment flow of ASSYST (Jackson and Usher, 1997). ...................................... 14
Figure 3. Ceilidh’s command line interface for accessing the exercises (Foxley, 1999). ........... 15
Figure 4. A sample screenshot of CourseMarker (CourseMarker, 2010). ................................. 16
Figure 5. A sample screenshot of BOSS (BOSS, 2010). ................................................................. 17
Figure 6. A sample screenshot of Web-CAT (Edwards, 2003). ...................................................... 18
Figure 7. The architecture of PASS version 3 (PASS, 2010). ....................................................... 20
Figure 8. Actual outputs of some programs for Sample Exercise 1............................................. 29
Figure 9. A stub program to be invoked by students to produce output .................................... 34
Figure 10. A student program that invokes the stub in Figure 3 to produce output .................... 34
Figure 11. A JUnit test method for testing a program solution for Sample Exercise 1. ............ 36
Figure 12. Regular expressions for matching the outputs for Sample Exercise 1. ..................... 37
Figure 13. A sample lex script for matching the last output line for Sample Exercise 1 ............. 38
Figure 14. Properties of a good automatic assessment approach (Jackson, 1991, p.138). ....... 42
Figure 15. An approach of output comparison based on token pattern. ..................................... 65
Figure 16. The tokens obtained after tokenization ........................................................................ 66
Figure 17. The hierarchy of matching rules ...................................................................................... 67
Figure 18. A token pattern composed of tokens tagged with matching rules ................................. 68
Figure 19. Matching a token pattern to actual output tokens ....................................................... 70
Figure 20. The main menu of the prototype for instructors ............................................................. 77
Figure 21. The Submission Marker module implements the token pattern approach ............. 79
Figure 22. The detailed flow of the tokenization process ................................................................. 81
Figure 23. The detailed flow of the tagging process ........................................................................ 82
Figure 24. The detailed flow of the comparison process ................................................................. 83
Figure 25. The matching process of a token pair ........................................................................... 84
Figure 26. Comparison options of PASS (PASS, 2010) ................................................................. 85
Figure 27. The prototype’s user interface for system login .............................................................. 86
Figure 28. User interface for creating programming problems ..................................................... 87
Figure 29. User interface for defining test cases ........................................................................... 88
Figure 30. User interface for tagging matching rules ..................................................................... 88
Figure 31. User interface for program submission ......................................................................... 89
Figure 32. Sample comparison results of the prototype..........................................................90
Figure 33. Acceptance in admissible variants by category......................................................94
Figure 34. User interface for programming problem creation................................................104
Figure 35. Part of the user interface for matching rule specification.......................................105
Figure 36. User interface for showing the matching result......................................................106
Figure 37. Sample input/output of Revision Exercise 1..........................................................108
Figure 38. Sample input/output of Revision Exercise 2..........................................................110
Figure 39. A test case for revision exercise 1 in plain text format (for CMFA).........................111
Figure 40. A test case fragment for revision exercise 1 in XML format (for TPA)...............112
Figure 41. Output deviations accepted by TPA for Revision Exercise 1.................................122
Figure 42. Output deviations accepted by TPA for Revision Exercise 2.................................122
Figure 43. Number of attempts until achieved best result for Revision Exercise 1.................125
Figure 44. Number of attempts until achieved best result for Revision Exercise 2.................125
List of Tables

Table 1. Sample exercises of various programming topics from different sources. .................. 48
Table 2. A comparison between ordinary exercises and auto-assessed exercises .................. 49
Table 3. Correctness of 111 output variants determined by two instructors ......................... 57
Table 4. Types of tokens implemented in the prototype ......................................................... 81
Table 5. Matching rules applicable to each token type .......................................................... 82
Table 6. The reassessment result of archived program submissions .................................... 91
Table 7. Marking decisions by human and TPA ................................................................. 91
Table 8. Marking decisions by human and CMFA ........................................................... 92
Table 9. Adoption of exercises for assessment using CMFA and TPA ................................. 96
Table 10. Adoption of exercises ....................................................................................... 97
Table 11. Allocation of student groups for the experiment ............................................... 114
Table 12. Schedule of the experiment ................................................................................ 115
Table 13. Number of submissions received among different student groups .................... 117
Table 14. Number of submissions by exercise and by assessment approach .................... 118
Table 15. Type I and Type II errors of correctness determination for Revision Exercise 1 ... 119
Table 16. Type I and Type II errors of correctness determination for Revision Exercise 2 ... 120
Table 17. Acceptance of admissible variants by TPA by category ...................................... 121
Table 18. Statistics on the number of attempts until first achieved best result (AFABER) .... 124
Table 19. Scoring of survey responses .............................................................................. 127
List of Sample Programming Exercises

Sample Exercise 1. An example programming exercise.......................................................... 29
Sample Exercise 2. An exercise with unnecessarily detailed specifications.......................... 33
Sample Exercise 3. An ordinary exercise (Deitel & Deitel, 2005) p.67, Ex.2.16 .................. 51
Sample Exercise 4. NUS (2010) Course CS1101C, Lab 0, Task 1 ...................................... 52
Sample Exercise 5. Revision Exercise 1 for the experiment .................................................. 108
Sample Exercise 6. Revision Exercise 2 for the experiment ................................................. 109
Sample Exercise 11. MIT (2010) Course 6.096, Assignment 1, Problem 7 ........................ 144
Sample Exercise 12. NUS (2010) Course CS1101C, Lab 1, Task 1 ..................................... 145
Sample Exercise 17. Deitel & Deitel (2005) Page 22, Exercise 5.6 .................................... 147
Sample Exercise 19. NUS (2010) Course CS1101C, Lab 5, Task 1 ..................................... 148
Sample Exercise 20. CityU-HK (2010) Course CS2363, Tutorial 5, Ex.1 ........................... 149
Sample Exercise 26. NUS (2010) Course CS1101C, Lab 7, Task 3 ..................................... 152
Sample Exercise 32. MIT (2010) Course 6.00, Assignment 3, Problem 1. .........................155
Sample Exercise 33. CityU-HK (2010) Course CS2363, Tutorial 9, Exercise 3 .................155
Sample Exercise 42. Etter (2005) Page 278, Exercise 35 ........................................158
Sample Exercise 44. MIT (2010) Course 6.096, Assignment 4, Problem 1 ...............159
Sample Exercise 45. NUS (2010) Course CS1101C, Lab 9, Task 1 .........................161
Sample Exercise 46. CityU-HK (2010) Course CS2363, Tutorial 7, Exercise 1 .......162
List of Appendices

Appendix 1. Programming exercises from textbooks and courses ............................................. 144
Appendix 2. Questionnaire on the acceptance of output variants ............................................. 163
Appendix 3. An assignment from CS2363 Computer Programming ........................................ 170
Appendix 4. Revision exercises for experiment (automated assessment) ................................. 173
Appendix 5. Revision exercises for experiment (manual assessment) .................................... 175
Appendix 6. Test cases for Revision Exercise 1 (CMFA) ......................................................... 177
Appendix 7. Test cases for Revision Exercise 2 (CMFA) ......................................................... 179
Appendix 8. Model program solution for Revision Exercise 1 ............................................. 182
Appendix 9. Model program solution for Revision Exercise 2 ............................................... 183
Appendix 10. Questionnaire for gathering feedback from students .................................. 184
Appendix 11. Written feedback from students .................................................................... 186