A methodology and software environment for testing process model's sequential predictions with protocols

Frank E. Ritter

20 December 1992
CMU-CS-93-101

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted to the Carnegie-Mellon University Department of Psychology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Psychology in the AI and Psychology program

This research was partially sponsored by a training grant from the Air Force Office of Scientific Research, Bolling AFB, DC; in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597, and in part by the School of Computer Science, Carnegie-Mellon University. The research was also supported in part by Digital Equipment Corporation through an equipment grant. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of U. S. Government or Digital Equipment Corporation.

Table of Contents

I Introduction to TBPA 1
1. Testing process models through protocol analysis 2
1.1 The need for routinely testing process models' sequential predictions 3
1.1.1 The potential benefits of routinely testing process models' sequential predictions 3
1.1.2 The difficulty of testing sequential predictions 4
1.2 The steps of testing process models' sequential predictions with protocol data 5
1.3 Developing a methodology for routinely testing process models' sequential predictions 6
1.3.1 A detailed specification of what is necessary for routine testing of process models with protocol data 6
1.3.2 An environment to support the needs of routine testing of process models 7
1.3.2.1 A tool supporting the interpretation and alignment of the data with respect to the model's predictions 7
1.3.2.2 A measurement system for telling where a model needs improvement 7
1.3.2.3 An interface for tracing, understanding, and modifying models 7
1.3.3 Documentation of the utility of the environment and methodology 8
1.3.4 Testing and extending the sequentiality assumption of verbal protocol generation 8
2. Testing process models with protocol data: Review of past work 9
2.1 The possible relationships between process models and protocols 9
2.2 Review of creating and testing models with protocol data 13
2.2.1 Exploratory analysis leading to process models 13
2.2.2 General testing of process models 16
2.2.3 Trace based protocol analysis 17
2.2.4 Summary of important data features 19
2.3 Tools related to process model testing 21
2.3.1 Tools for building models from protocols 21
2.3.1.1 Declarative knowledge coding tools 21
2.3.1.2 Exploratory protocol analysis tools 21
2.3.2 Model testing tools 23
2.3.2.1 Strategy classification tools based on process models 23
2.3.2.2 Model tracing modules within intelligent tutoring systems 23
2.3.2.3 Tools for aligning the sequential predictions with data 24
2.3.3 Tools for building and understanding models 26
2.3.3.1 Process model induction tools 26
2.3.3.2 Tools for understanding and building symbolic cognitive models 26
2.3.3.3 Knowledge acquisition tools 28
2.3.4 Summary of useful tool features 29
2.4 Measures of model to data comparison 31
2.4.1 Using criteria to develop a set of measurements 33
2.4.2 Description of measurement inputs 35
2.4.3 Non-numeric descriptive measures 39
2.4.4 Simple numeric measures 41
2.4.5 Measures of component utility 44
2.4.6 Inferential measures 45
2.4.7 A unified view: Criterion based model evaluation 47
2.4.8 Summary of measures 48
2.5 Previous models of process model testing 50
2.6 Summary of lessons for process model testing methodology and tools 53

Appendix to Chapter 2: Review of the Card model alignment algorithm 55

3. Requirements for testing process models using trace based protocol analysis 57

3.1 Definition of trace based protocol analysis (TBPA) 57

3.1.1 The inputs to TBPA 57

3.1.1.1 A 0^{th} order functional model 57

3.1.1.2 Transcribed protocol data 58

3.1.2 The TBPA loop and its requirements 58

3.1.2.1 Step 1: Run the model to create predictions 59

3.1.2.2 Step 2: Use the predictions to interpret the data 62

3.1.2.3 Step 3: Analyze the results of the comparison 63

3.1.2.4 Step 4: Revise the model to reduce the discrepancies 64

3.2 Supporting TBPA with an integrated computer environment 66

3.2.1 Why an integrated environment is needed 66

3.2.2 The environment must automate what it can 67

3.2.3 The environment must support the user for the rest 67

3.3 The role of an intelligent architecture in the testing process 68

3.3.1 Soar: The architecture used in this environment 68

3.3.2 Making functional models examinable 70

3.3.3 Using the architecture to automate the analysis 72

3.4 Summary of requirements and description of the environment's design 72

II Supporting the TBPA methodology: A description of the Soar/MT environment 76

4. A spreadsheet for comparing the model's predictions with the data 77

4.1 Displaying and editing the correspondences 78

4.2 Automatically aligning unambiguous segments 81

4.3 Interpreting ambiguous actions 83

4.4 Supporting the global requirements 84

4.4.1 Providing an integrated system 84

4.4.2 Automating what it can 84

4.4.3 Providing a uniform interface including a path to expertise 84

4.4.4 Providing general tools and a macro language 85

4.4.5 Displaying and manipulating large amounts of data 85

4.5 Summary 85

5. Visual, analytic measures of the predictions' fit to the data 87

5.1 Creating the operator support display automatically 87

5.2 Understanding the relative processing rate 89

5.2.1 A display for comparing the relative processing rate 89

5.2.2 Using the relative processing display to test the sequentiality assumption of verbal protocol production 94

5.3 Creating additional displays 95

5.3.1 S: An architecture for creating displays 95

5.3.2 S-mode: An integrated, structured editor for S 96

5.4 Supporting the global requirements 97

5.4.1 Providing an integrated system 97

5.4.2 Automating what it can 97

5.4.3 Providing a uniform interface including a path to expertise 97

5.4.4 Providing general tools and a macro language 97

5.4.5 Displaying and manipulating large amounts of data 98

5.5 Summary of measures and recommendations for use 98

6. The model manipulation tool -- the Developmental Soar Interface (DSI) 100

6.1 Providing the model's predictions in forms useful for later comparisons and 101

analysis
6.1.1 Providing predictions for comparison with the data 102
6.1.2 Aggregating the model's performance 103
6.2 Displaying the model so that it can be understood 104
6.2.1 Normative displays of the model 106
6.2.2 Descriptive displays of the model's performance 109
6.2.3 The working memory walker 111
6.2.4 A pop-up menu and dialog boxes to drive the display 112
6.3 Creating and modifying the model 114
6.3.1 Soar-mode: An integrated, structured editor for Soar 114
6.3.2 Taql-mode: An integrated, structured editor for TAQL 115
6.3.3 The Soar Command Interpreter 115
6.4 Supporting the requirements based on the whole process and its size 116
6.4.1 Providing consistent representations and functionality 116
6.4.2 Automating what it can: Keystroke savings 117
6.4.3 Providing a uniform interface including a path to expertise 117
6.4.4 Providing a set of general tools and a macro language 118
6.4.5 Displaying and manipulating large amounts of information 119
6.5 Lessons learned from the DSI 119
6.5.1 The relatively large size of the TAQL grammar 119
6.5.2 Behavior in Soar models is not just search *in* problem spaces 119
6.5.3 Soar models do not have explicit operators 122
6.6 Summary 123
III Performance demonstrations of Soar/MT and Conclusions 125
7. Performance demonstration I: Analyzing the Browser-Soar model faster and more deeply 126
7.1 Description of Browser-Soar and its data 126
7.2 Producing richer analyses more quickly 133
7.2.1 The interpretation of data with respect to the model trace done faster and tighter 133
7.2.2 Operator support displays created automatically -- as a set they highlight periodicity in behavior 134
7.3 Where the model and subject process at different rates shown clearly 137
7.3.1 Processing rate display based on decision cycles shows that the quality of fit is high 137
7.3.2 The processing rate display can be based on other measures of the model's effort 140
7.4 High level features of the Browser-Soar model made apparent 141
7.4.1 Browser-Soar as routine behavior is made directly visible 141
7.4.2 Noting Browser-Soar's large goal depth 142
7.4.3 Modifying Browser-Soar 142
7.4.4 Testing the modified Browser-Soar 143
7.5 Testing and extending the sequentiality assumptions of protocol generation theory 144
7.5.1 Are verbalizations generated sequentially? 148
7.5.2 Are mouse actions generated sequentially? 148
7.5.3 Does the sequentiality assumption hold across verbalizations and mouse actions? 148
7.6 Conclusions about Browser-Soar and the TBPA methodology 150
7.6.1 Some conclusions about Browser-Soar 150
7.6.2 Some conclusions about the methodology 151
Appendixes to Chapter 7 152

1 Alignment of the Write episode of Browser-Soar 152
2 Displays of each analytical measure for each episode of Browser-Soar 158
8. Performance demonstration II: Use of Soar/MT components by others 164
8.1 Usage of the Developmental Soar Interface to develop Soar models 164
8.2 Usage of S-mode to create functions in S 166
Appendix to Chapter 8: Survey distributed to Soar users 168
9. Contributions and steps toward the vision of routine automatic model testing 172
9.1 A methodology for testing the sequential predictions of process models 173
9.2 Each step in the methodology was supported in a software environment 174
9.2.1 Interpreting and aligning the model's predictions and the data 174
9.2.2 Analyzing the results of the testing process 175
9.2.3 Steps related to manipulating the model: Prediction generation and modification 175
9.2.4 The synergy from integration 176
9.3 Validated and extended the sequentiality assumption of protocol generation theory 176
9.4 Progress toward the vision of routine applied theoretically guided protocol analysis 177
9.5 Concluding remarks 178
References 179
I. How to obtain the software described in this thesis 193