INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Semantic and pragmatic parsing based on systemic grammar and layered abduction

Hartigan, Julie Ann, Ph.D.
The Ohio State University, 1994

Copyright ©1994 by Hartigan, Julie Ann. All rights reserved.
SEMANTIC AND PRAGMATIC PARSING
BASED ON SYSTEMIC GRAMMAR AND
LAYERED ABDUCTION

DISSERTATION
Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By

Julie Ann Hartigan, B.S., M.S.

The Ohio State University

1994

Dissertation Committee:
B. Chandrasekaran
J. Josephson
R. Kasper
D. Wang

Approved by

Adviser
Co-adviser
Department of Computer and
Information Science
For my family, including Cindy
ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to several people who have made an impact on my life and my education over these past several years. I would like to thank Drs. B. Chandrasekaran and John Josephson for inspiring me to pursue my studies in artificial intelligence. Thanks also goes to the other members of my advisory committee, Drs. Deliang Wang and Robert Kasper, for their suggestions and comments. I would like to also thank Dr. Terry Patten for all the time and energy he spent encouraging me to follow my dreams and sculpting my knowledge in computational linguistics.

Though I have made many friends along this journey and I thank them all, I would like to give special thanks to a few. Thanks to Barbara Becker for getting me started in this line of research. Thanks to Mahendra Ramachandran for his constant friendship, through the good times and the bad. Thanks to Manas Mandal for all the good advice on how to reach and attain graduation. Thanks to Dr. Richard Fox who, though very far away, lent as much guidance as he could. And special thanks to Dr. Jun Lim, who was always there to listen to my problems and gripes, to which he suggested entertaining solutions.

Thanks to my dog, Cindy, for her unconditional love. When I thought my life was going to pieces and I just could not take it anymore, she would remind me of how lucky I truly was. I thank her for surviving all four of my graduations and never complaining about the late dinners and the long, lonely nights.
Last but not least, with deepest gratitude I thank my family, whose sacrifices and understanding can never be repayed. It has been a long and arduous journey that I am sure all of us are delighted to see come to an end.
August 26, 1964 ......................... Born - Columbus, Ohio

1986 ................................. B.S. Computer Science and Mathematics, Heidelberg College, Tiffin, Ohio

1986 - 1990 ......................... Researcher, Information Systems Department, Battelle Memorial Institute, Columbus, Ohio

1989 ................................. M.S. Computer and Information Science, The Ohio State University, Columbus, Ohio

1990 - 1991 ......................... Graduate Teaching Associate, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio

1991 - 1992 ......................... Graduate Research Associate, Laboratory for Artificial Intelligence Research, The Ohio State University, Columbus, Ohio

1992 - present ......................... Graduate Teaching Associate and Introductory Course Coordinator, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio
PUBLICATIONS


FIELDS OF STUDY

Major Field: Computer and Information Science

Studies in:

Artificial Intelligence  Prof. B. Chandrasekaran
Linguistics  Prof. P. Culicover
Computer Architecture  Prof. P. Sadayappan
# Table of Contents

**DEDICATION** ........................................................................................................................... ii  
**ACKNOWLEDGEMENTS** ......................................................................................................... iii  
**VITA** ........................................................................................................................................ v  
**LIST OF TABLES** ................................................................................................................... xii  
**LIST OF FIGURES** .................................................................................................................. xiii  

## CHAPTER

I. Introduction ............................................................................................................... 1  

1.1 Brief System Description ........................................................................ 4  
1.1.1 Semantic and Pragmatic Interpretation System ............................... 5  
1.1.1.1 Knowledge Representation ..................................................... 5  
1.1.1.2 Control Strategy ...................................................................... 5  
1.1.1.3 Top-Down Guidance .............................................................. 9  
1.1.1.4 Evaluation of “Understanding” ............................................. 11  
1.2 Research Objectives ................................................................................. 12  
1.3 Outline ...................................................................................................... 13  

II. Classification, Abduction and Systemic Grammar ............................................. 16  

2.1 Introduction ................................................................................................ 16  
2.2 Representation and Algorithm ............................................................... 17  
2.2.1 Classification ................................................................................... 17  
2.2.2 Abduction .......................................................................................... 18  
2.3 Top-Down Guidance ................................................................................. 24  
2.4 Systemic Grammar ................................................................................... 27  
2.4.1 Basic Concepts ................................................................................. 28  
2.4.2 The Strata ......................................................................................... 37  
2.5 High-Level Explanation of Operation ................................................... 39  
2.6 Example .................................................................................................... 41  
2.7 Conclusion ............................................................................................... 45  

vii
III. A Survey of Related Works .................................................................47

3.1 Introduction .........................................................................................47

3.2 NLU Systems .....................................................................................47
  3.2.1 Systemic Grammar Parsing - Kasper ...........................................48
  3.2.2 TACITUS - Hobbs .......................................................................49
  3.2.3 Wimp - Charniak .......................................................................51
  3.2.4 Implementation of Context in NLU Systems ................................53

3.3 Abduction systems .............................................................................56
  3.3.1 Tractable Abduction - Josephson et al. .........................................57
  3.3.2 General Set Covering - Peng and Reggia .....................................60
  3.3.3 Belief Nets - Pearl .......................................................................60
  3.3.4 Neural Networks - Thagard .........................................................62
  3.3.5 Parallelizing Abduction ..............................................................64

3.4 Conclusion .........................................................................................65

IV. SAPI - A Semantic And Pragmatic Interpretation System .................66

4.1 Introduction .........................................................................................66

4.2 Input Sentences ..................................................................................67

4.3 When an Essential may not be Essential ........................................69

4.4 Logicogrammatical Analysis ...............................................................70
  4.4.1 Input and Output ..........................................................................71
  4.4.2 Highly-Likely Phase .....................................................................72
    4.4.2.1 Strictly Syntactic Processing .................................................72
    4.4.2.2 Anticipatory Top-Down Guidance .......................................74
    4.4.2.3 Run-time Top-Down Guidance ...........................................75
  4.4.3 Likely Phase ..................................................................................77
    4.4.3.1 Traversing a Hierarchy .........................................................78
    4.4.3.2 Firing Realization Rules .......................................................80
    4.4.3.3 Conflicting Composite .........................................................82

4.5 Pragmatic Analysis ............................................................................88
  4.5.1 Input and Output ..........................................................................89
  4.5.2 Threshold-Controlled Abduction .................................................89

4.6 Implementation Level ........................................................................94

4.7 Conclusion .........................................................................................104
V. Experiment Results of SAPI ................................................................. 105

5.1 Introduction ....................................................................................... 105
5.2 Inputs, Outputs, and Processing ....................................................... 105
5.3 Expected Results ............................................................................. 108
5.4 Sentence #1 - "What time can you leave?" ...................................... 108
  5.4.1 Effects of Anticipatory Guidance .............................................. 112
  5.4.2 Run-Time Guidance Effects ..................................................... 114
5.5 Sentence #2 - "I can put you on that 10pm flight." .......................... 117
  5.5.1 Effects of Anticipatory Guidance .............................................. 120
  5.5.2 Run-Time Guidance Effects ..................................................... 121
5.6 Sentence #3 - "How many seats are you looking for?" .................... 124
  5.6.1 Effects of Anticipatory Guidance .............................................. 127
  5.6.2 Run-Time Guidance Effects ..................................................... 128
5.7 The Effect of Anticipatory Guidance on Sentences 1-3 ...................... 130
5.8 Sentence #4 - "I'm not finding anything then." ............................... 133
  5.8.1 Effects of Anticipatory Guidance .............................................. 133
  5.8.2 Run-Time Guidance Effects ..................................................... 135
5.9 Sentence #5 - "Can you leave earlier?" .......................................... 137
  5.9.1 Effects of Anticipatory Guidance .............................................. 137
  5.9.2 Run-Time Guidance Effects ..................................................... 139
5.10 The Effect of Anticipatory Guidance on Sentences 4 and 5 .......... 141
5.11 Sentence #6 - "Let me look." ........................................................ 144
  5.11.1 Effects of Anticipatory Guidance .............................................. 144
  5.11.2 Run-Time Guidance Effects ..................................................... 146
5.12 Sentence #7 - "Well, all I'm showing is a 10pm flight." ................. 147
  5.12.1 Effects of Anticipatory Guidance .............................................. 147
  5.12.2 Run-Time Guidance Effects ..................................................... 149
5.13 Sentence #8 - "I've got a seat on an 11:00 flight on Treetop
Airlines." .............................................................................................. 150
  5.13.1 Effects of Anticipatory Guidance .............................................. 150
  5.13.2 Run-Time Guidance Effects ..................................................... 151
5.14 The Effect of Anticipatory Guidance on Sentences 6-8 .................. 153
5.15 Sentence #9 - "When can I bring you back?" ............................... 156
  5.15.1 Run-Time Guidance Effects ..................................................... 156
5.16 Sentence #10 - "The round-trip fare will be $295." ....................... 157
  5.16.1 Run-Time Guidance Effects ..................................................... 157
5.17 Sentence #11 - "When can you leave?" .............................. 158
5.17.1 Run-Time Guidance Effects ....................................................... 158
5.18 Excuse me? Can you repeat that? ........................................................ 159
5.18.1 Correctness ................................................................................... 159
5.18.2 Completeness ............................................................................... 162
5.19 Conclusion ........................................................................................... 167

VI. Discussion of Results ................................................................. 169
6.1 Introduction .............................................................................................. 169
6.2 Expected Results .................................................................................... 170
6.3 Results ...................................................................................................... 171
6.3.1 Results of Anticipatory Guidance ............................................... 172
6.3.2 Results of Run-Time Guidance ................................................... 175
6.4 Interpretation of Results ........................................................................ 176
6.4.1 Bogus Input? ................................................................................. 176
6.4.2 Anticipatory Guidance without Run-Time Guidance ............... 177
6.4.3 Anticipatory Guidance with Run-Time Guidance ..................... 179
6.4.4 Meta-Knowledge .......................................................................... 182
6.5 Conclusion .............................................................................................. 186

VII. Future Work and Further Implications .................................................. 188
7.1 Introduction .............................................................................................. 188
7.2 Future Work ........................................................................................... 189
7.2.1 Expand the Strata ............................................................................ 189
7.2.1.1 Pragmatic Stratum ..................................................................... 189
7.2.1.2 Logicogrammatical Stratum ............................................... 191
7.2.1.3 Additional Strata ................................................................ 192
7.2.1.4 Parallel Processing .............................................................. 195
7.3 Generalizing the Concepts .................................................................... 197
7.3.1 Computational Complexity ......................................................... 198
7.3.1.1 Semantic Interpreter ........................................................... 199
7.3.1.2 Pragmatic Interpreter .......................................................... 201
7.3.2 Scalability ..................................................................................... 204
7.3.3 Increased Coverage, Same Domain ............................................ 204
7.3.4 Additional Domains ................................................................. 204
7.3.5 Broader Applicability ............................................................... 205
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Effects of anticipatory guidance on sentence 1</td>
<td>112</td>
</tr>
<tr>
<td>2. Run-time guidance effects on sentence 1</td>
<td>115</td>
</tr>
<tr>
<td>3. Effects of anticipatory guidance on sentence 2</td>
<td>120</td>
</tr>
<tr>
<td>4. Run-time guidance effects on sentence 2</td>
<td>123</td>
</tr>
<tr>
<td>5. Effects of anticipatory guidance on sentence 3</td>
<td>127</td>
</tr>
<tr>
<td>6. Run-time guidance effects on sentence 3</td>
<td>129</td>
</tr>
<tr>
<td>7. Speedup caused by increasing levels of anticipation for sentences 1-3</td>
<td>131</td>
</tr>
<tr>
<td>8. Effects of anticipatory guidance on sentence 4</td>
<td>133</td>
</tr>
<tr>
<td>9. Run-time guidance effects on sentence 4</td>
<td>136</td>
</tr>
<tr>
<td>10. Effects of anticipatory guidance on sentence 5</td>
<td>138</td>
</tr>
<tr>
<td>11. Run-time guidance effects on sentence 5</td>
<td>140</td>
</tr>
<tr>
<td>12. Speedup caused by increasing levels of anticipation for sentences 4 and 5</td>
<td>142</td>
</tr>
<tr>
<td>13. Effects of anticipatory guidance on sentence 6</td>
<td>144</td>
</tr>
<tr>
<td>14. Run-time guidance effects on sentence 6</td>
<td>146</td>
</tr>
<tr>
<td>15. Effects of anticipatory guidance on sentence 7</td>
<td>147</td>
</tr>
<tr>
<td>16. Run-time guidance effects on sentence 7</td>
<td>149</td>
</tr>
<tr>
<td>17. Effects of anticipatory guidance on sentence 8</td>
<td>150</td>
</tr>
<tr>
<td>18. Run-time guidance effects on sentence 8</td>
<td>152</td>
</tr>
<tr>
<td>19. Speedup caused by increasing levels of anticipation for sentences 6-8</td>
<td>154</td>
</tr>
<tr>
<td>20. Run-time guidance effects on sentence 9</td>
<td>156</td>
</tr>
<tr>
<td>21. Run-time guidance effects on sentence 10</td>
<td>157</td>
</tr>
<tr>
<td>22. Run-time guidance effects on sentence 11</td>
<td>158</td>
</tr>
<tr>
<td>23. Results of the incompleteness test on “What time can you leave?”</td>
<td>163</td>
</tr>
<tr>
<td>24. Results of the incompleteness test on “I can put you on that 10pm flight.”</td>
<td>166</td>
</tr>
<tr>
<td>25. Input sentences</td>
<td>171</td>
</tr>
<tr>
<td>26. Processing time in average number of seconds of eleven sentences with varying levels of anticipatory guidance, with and without run-time guidance</td>
<td>172</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>1. Abduction versus layered abduction.</td>
<td>8</td>
</tr>
<tr>
<td>2. Graphical view of layered abduction and top-down guidance for semantic and pragmatic interpretation.</td>
<td>11</td>
</tr>
<tr>
<td>3. Abduction.</td>
<td>19</td>
</tr>
<tr>
<td>4. Layered abduction.</td>
<td>22</td>
</tr>
<tr>
<td>5. Top-down guidance in layered abduction.</td>
<td>26</td>
</tr>
<tr>
<td>6. System networks of the Deicticity and Primary-Tense systems.</td>
<td>29</td>
</tr>
<tr>
<td>7. The various system structures.</td>
<td>31</td>
</tr>
<tr>
<td>8. Gate feature structure.</td>
<td>32</td>
</tr>
<tr>
<td>9. Functional analyses of a clause.</td>
<td>34</td>
</tr>
<tr>
<td>10. Systemic grammar, a stratified approach.</td>
<td>39</td>
</tr>
<tr>
<td>11. A small portion of a sample pragmatic stratum.</td>
<td>44</td>
</tr>
<tr>
<td>12. A small portion of a sample logicogrammatical stratum.</td>
<td>45</td>
</tr>
<tr>
<td>13. Examples of the Elman, Jordan and Mozer connectionist architectures.</td>
<td>56</td>
</tr>
<tr>
<td>14. Sample network for ECHO.</td>
<td>63</td>
</tr>
<tr>
<td>15. The conversation.</td>
<td>68</td>
</tr>
<tr>
<td>16. Base conflation.</td>
<td>85</td>
</tr>
<tr>
<td>17. Three function conflation.</td>
<td>86</td>
</tr>
<tr>
<td>18. Four or more function conflation.</td>
<td>87</td>
</tr>
<tr>
<td>19. Example of pragmatic interpretation after step one.</td>
<td>90</td>
</tr>
<tr>
<td>20. Example of pragmatic interpretation after step three.</td>
<td>91</td>
</tr>
<tr>
<td>21. Example of pragmatic interpretation after step four.</td>
<td>92</td>
</tr>
<tr>
<td>22. Example of pragmatic interpretation after step five.</td>
<td>93</td>
</tr>
<tr>
<td>23. Ending configuration of the example of pragmatic interpretation.</td>
<td>94</td>
</tr>
<tr>
<td>24. Portion of a sample syntactic parse tree.</td>
<td>95</td>
</tr>
<tr>
<td>25. Sample input to the semantic interpreter.</td>
<td>99</td>
</tr>
<tr>
<td>26. Logicogrammatical features as output from the semantic interpreter.</td>
<td>100</td>
</tr>
<tr>
<td>27. Logicogrammatical functions as output from the semantic interpreter.</td>
<td>101</td>
</tr>
</tbody>
</table>
CHAPTER I

Introduction

The abilities to understand and generate natural language are fundamental to human existence; it is what enables us to communicate with one another. This communication is vital for the dissemination of ideas, concepts, fantasies, and stories, as well as the propagation of news and information. How people manage to encode and decode the intended information is a difficult process to describe, let alone program a computer to perform.

Natural language processing is such a vast topic that a thorough discussion of it is not possible. Therefore, this discussion will concentrate on the natural language understanding (NLU) aspect of the task and in particular, semantic and pragmatic interpretation. Some of the difficulties encountered in describing NLU is defining what exactly the task is, determining what constitutes a successful understanding, and identifying what computational strategies and knowledge representations to use to perform the task.

What do we mean by “natural language understanding?” It is the understanding of natural language, where natural language is any language spoken among a group of people to convey information. The “understanding” portion of the phrase is really a subjective term, but in general a sentence is said to be understood if it creates the desired effect on the
recipient's actions and beliefs. To create an appropriate understanding, the speaker must be aware of the current beliefs and knowledge possessed by the intended receiver, as well as the degree to which the recipient "reads between the lines" or assumes alternative interpretations. Knowing these things will help the speaker generate a statement which will then be understood appropriately by the listener.

Once the speaker knows what information is to be conveyed, the speaker produces an auditory and/or visual stimulus which encodes the information. The recipient interprets this stimulus into some internal representation of meaning. To make this transformation from auditory and/or visual stimulus to an internal representation of meaning, some form of inference is used. Philosophers and psychologists have long suggested that perception relies on some form of inference, perhaps deduction, induction, or simple recognition [Josephson and Josephson, 1994]. Recently researchers have suggested that the task is actually performed with abduction.

Abduction is a form of inference that translates input data that describes something into an explanatory hypothesis that best explains the data. Abduction is discussed in greater depth in Chapter II. My work is based on the claim that the task of natural language understanding can be viewed and discussed as being an abductive task. The abductive hypotheses may have been acquired through deduction, induction or abduction, but the task itself is abductive. The sentence "I'm not finding anything then." has multiple interpretations at multiple levels (e.g. phonetic, syntactic, semantic, pragmatic). In this case, the sentence pragmatically can be explained as one person informally telling another that the current search has been unsuccessful.
Computationally assigning a pragmatic interpretation directly to a sentence is difficult to justify. The abductive interpretation would seem more believable, or at least easier to justify, if there were several smaller abductions created that would lead up to the final explanation. This being the case, it may be beneficial to appeal to the idea held by many linguists that NLU tends to be a layered process.

Language understanding is often viewed as an orderly progression of layers. The base layer is the input received by some sensory organ such as the ear receiving acoustic input. This layer creates an explanation of its input. This conclusion is then explained by the next layer which may be the syllable layer. This in turn is explained by words, then syntax, then semantics, and then finally pragmatics. This ordering of layers is just an example, many possible orders and layers exist. The point is that the order of the layers progress from a sensory input to a more abstract cognitive category description. This order is considered a bottom-up approach, where each layer forms a best explanation of the conclusion abduced at the less abstract layer. An example of a layered approach to natural language is systemic grammar as proposed by Halliday (1985) which is further explained in Chapter II.

Though researchers agree that NLU is primarily a bottom-up process, it can not be denied that top-down guidance may occur. Top-down guidance is where the more abstract cognitive description guides a lower level by suggesting or predicting the existence of certain hypotheses. People benefit from top-down guidance to help in the processing of language. A person possesses general knowledge and common sense, and at some level of comprehension, grasps the context of the situation which will help predict what will be said and will occur. For example, imagine a person who wishes to make an airline reservation.
If the person often makes airline reservations, then he/she will have great expectations of how the conversation will ensue. If the person has never made an airline reservation but has made several cruise reservations through a travel agent, then he/she will still have some level of expectation, but to a lesser degree. Then again, perhaps the person has never dealt with a travel agent, but has experience making dinner reservations. This still provides the person with some level of familiarity with the act of making reservations, just the context is a bit further removed and therefore the expectations will be fewer or less powerful. In the final case, albeit unlikely, the person has absolutely no idea of what to expect, has never made a reservation and has never conversed with someone in the service industry. In this case, the person will be forced to process the conversation in a predominantly bottom-up manner, without the aid of top-down guidance.

Having presented the notion put forth by linguists that NLU occurs in layers, and also having presented the notion of many researchers that the transformation from sensory input to an internal representation of meaning is performed abductively, it is then necessary to propose an abductive strategy that can operate on layers. This strategy is referred to as layered abduction. Layered abduction is the use of multiple abductive processes, where the conclusions produced by one abductive process become the data to the next. Layered abduction is further discussed in Chapter II. Using layered abduction for NLU is proposed in Josephson and Josephson (1994).

1.1 Brief System Description

Many researchers have attempted to program various aspects of NLU utilizing many different computational strategies and representations with a variety of specialized
linguistic representations. The computational model implemented as part of this research uses a systemic grammar in a layered abductive framework for semantic and pragmatic interpretation of natural language. This section gives a brief overview of the system, its control strategy, processing technique, and method of evaluation.

### 1.1.1 Semantic and Pragmatic Interpretation System

The Semantic And Pragmatic Interpretation system (SAPI) is a system designed to perform semantic and pragmatic interpretation\(^1\). SAPI operates in the domain of a travel agent making a reservation. It accepts as input a syntactic parse and produces as output a semantic parse which it uses to create a pragmatic parse.

#### 1.1.1.1 Knowledge Representation

SAPI uses systemic grammar (further discussed in Chapter II) to represent the linguistic knowledge. It organizes the knowledge into two layers or strata; the logicogrammatical stratum which is used to create a semantic interpretation, and the pragmatic stratum which is used to provide top-down guidance, to determine if the semantic interpretation is correct, and to create a pragmatic interpretation.

#### 1.1.1.2 Control Strategy

The control strategy used in SAPI is layered abduction. To explain layered abduction it is necessary to first understand abduction. Both abduction and layered abduction are fully described in Chapter II.

---

1. The pragmatic interpreter portion is based on the pragmatic abducer created by Becker (1994).
Abduction

Abduction is a form of inference, as are induction and deduction. Briefly stated, deduction is an infallible form of inference. Given true premises, the conclusion of a deduction must be true. Deductions are truth preserving since their conclusions must be true if the premises are true.

Induction is a fallible form of inference. The predominant type of induction is "inductive generalization" which determines from some sample set of cases some characteristic to apply to a larger population. An example of an inductive argument is: All observed dogs bark, therefore all dogs bark. This inductive argument is based on experience, which may be incomplete. (A basenji is a breed of dog that does not bark.) Induction is an example of ampliative inference since the conclusion may represent more information than what was presented in the premises.

Abduction is fallible and is ampliative, like induction. Abduction infers a best explanation for an experience. Given a set of data, the hypotheses that can best account for, or explain, the data are concluded. In abduction, the conclusion explains the premise. In the above example, it has been inductively proven that all dogs bark, however all dogs barking does not explain why all the observed dogs barked. The goal of abduction would be to explain this. Abduction seems to be an ubiquitous form of processing [Josephson and Josephson, 1994].

An abductive task is one that has inputs that need to be explained and the output represents the best explanation of the input. The steps taken to arrive at the output, given the input, is the abductive process. A task that can be couched in these terms, such as natural
language understanding, can be called an abductive task. In NLU, the input is a set of data, such as an acoustic signal or syntactic parse, and the goal is to process these data to create a description that captures the semantics and pragmatics of the input. The composite explanation is comprised of the concluded hypotheses that best explain the data.

The abductive process used in NLU can be realized by several different abductive techniques. The choice of technique depends on the form of the input, the required form of the output, the knowledge available to convert the input to the output, the computational resources available, etc. Regardless of the technique chosen, they all abduce a best explanation from the input.

**Layered Abduction**

Layered abduction has an abductive mechanism for each problem-solving level and the output at one level becomes the input to the next. Where regular abduction has a single mapping from findings to hypotheses, layered abduction can come to a conclusion (or partial conclusion) at one level of inferencing, and make more abstract inferences based on that conclusion at another level. Abductive conclusions at one level are passed up to the next level and are used as findings to be explained. This is bottom-up or data-driven processing. When a hypothesis at a higher level, or more abstract type, is found to explain the less abstract hypothesis, it may offer expectations to aid the lower level inferences or prompt the lower level to refine a hypothesis [Josephson, 1988]. This is top-down or hypothesis-driven processing. Figure 1 graphically depicts the difference between abduction and layered abduction.
SAPI is a layered abduction system where the logicogrammatical stratum creates partial conclusions to be explained by the pragmatic stratum and from this information the pragmatic stratum predicts additional information in the logicogrammatical stratum. SAPI creates a “best” explanation of the syntactic input where “best” is defined as the first conclusion created that represents a complete and consistent path in the logicogrammatical stratum. The logicogrammatical stratum is in the form of a classification hierarchy and a complete path is one that, for every node that is part of the conclusion, if it has children,
then at least one of its children are in the composite. A consistent path is one that for all the nodes that are part of the conclusion, the structural and semantic constraints associated with the nodes do not conflict. A detailed explanation of how SAPI performs layered abduction is described in Chapter IV.

The definition of "best" explanation being the first explanation encountered is acceptable for this problem. The reason is that the knowledge representation (systemic grammar) is highly constrained, and finding any solution or complete and consistent path is a difficult problem. When one is found, then that one is probably the only one possible. If another exists, it is a less likely conclusion. Other definitions of "best" will be discussed in Chapter III.

1.1.1.3 Top-Down Guidance

A system that only operates in a bottom-up manner is limited as it does not take full advantage of the knowledge available. It is wise to incorporate top-down processing in order to generate expectations at the lower level, to cast doubt on problematic data, or to resolve conflicts. With a system that has both top-down and bottom-up processing, a decision must be made as to how to integrate the two processing directions. The current available knowledge or the status of the problem-solving may determine the best direction [Punch, 1989], or the system could be constructed to integrate the two at specified times. "By using bottom-up and top-down processing and a flexible control strategy, a layered abduction mechanism can generate very plausible partial conclusions at one level, and use these as islands of certainty to leverage further problem-solving, both at the same level of inference and at higher and lower levels," [Fox, 1992, page 11].
Top-down guidance exists in SAPI from the pragmatic stratum to the logicogrammatical stratum. The pragmatic stratum makes predictions about what logicogrammatical hypotheses will be part of the semantic conclusion. In this research, top-down guidance is manifested in two forms; anticipatory top-down guidance and run-time top-down guidance.

Anticipatory top-down guidance represents what the listener expects to hear, what he/she anticipates will be said. In SAPI, this form of guidance is dependent on the familiarity the listener has with the context of the conversation. Anticipatory guidance is realized by initially seeding the composite to be formed at the logicogrammatical stratum with those hypotheses that were determined by a domain expert to be representative of what a listener of a certain level of familiarity would expect.

Run-time top-down guidance occurs during processing as a result of the problem-solving. It is realized by a passing of control between the pragmatic and the semantic interpreters to further identify or predict logicogrammatical features. Run-time guidance is terminated when the pragmatic stratum can offer no additional information to the semantic interpreter or when a complete and consistent path has been created in the logicogrammatical stratum.

The various strata, anticipatory top-down guidance, run-time top-down guidance, and the bottom-up processing used in SAPI are graphically depicted in Figure 2.
1.1.1.4 Evaluation of "Understanding"

When SAPI creates an explanation of the syntactic input, how is it determined that the semantic and the pragmatic interpretation accurately represent an understanding of the sentence? Since understanding is subjective and is dependent on many factors, a method was devised to determine when the input sentence had been understood. The method used was to generate a sentence from the pragmatic interpretation. If the sentence generated was the same as the one that created the syntactic input, then it was assumed that a basic understanding had been achieved. The system that performed the generation is called A Natural-language Interface for a Travel Agent (ANITA) and is discussed in [Patten et al., 1992].
1.2 Research Objectives

Much work in the field of natural language understanding (NLU) has been to prove the usefulness of a certain theory, to negate a criticism to a theory, or to fortify various claims that were being touted. Rather than merely stating the objectives of this work, perhaps it would be best to identify claims that are not being made.

This work is not trying to add to any linguistic theory. It utilizes the theory of systemic grammar (discussed in Chapter II), which is an existing theory, and purposefully does not tamper with the theory, but utilizes it as the linguistic knowledge representation. This work uses the explicit systemic grammar that was the basis for a text generation system [Patten, 1988].

This work is not making any neurological suppositions. How, or where, humans represent grammar in their brains, or if it is represented, is not considered in this work. No claim is being made that this method is neurologically feasible, neither in the representation nor in the algorithm. Neuroscience is its own field and it is left to the neuroscientists to determine how the human brain works.

This work does not present a new problem-solving strategy. It utilizes an established abductive technique. Since the concluded hypotheses at one level are presented to the next level as data, it is a layered abduction system. The technique is enhanced by making use of top-down knowledge in the form of anticipatory and run-time top-down guidance to help guide the processing.
The surface objective of this work was to unify the concepts of systemic grammar and layered abduction into a NLU system. This unification mandated many implementation decisions. These issues and their solutions are discussed in Chapter IV.

Once the surface objective was met, two deeper objectives were explored. One such objective was to analyze the computational implications of utilizing top-down guidance. SAPI utilized two forms of top-down guidance, anticipatory and run-time guidance, to aid the processing at the logicogrammatical layer. The goal was to show that top-down guidance improves the efficiency of the understanding by reducing the number of hypotheses to be considered.

Lastly, a goal of this research was to show that by choosing abduction as the form of inference to create the understanding, coping with noisy (incomplete and incorrect) data is more easily facilitated. Inherent in SAPI's abductive processing is its ability to cope, to some degree, with incomplete syntactic input. With very few minor changes, SAPI can also cope with conflicting input, where either the anticipated information or the syntactic parse is incorrect. These issues are discussed in Chapter V.

1.3 Outline
This research examines the results of using contextual knowledge in the form of anticipatory and run-time top-down guidance for natural language understanding. The research discovers and explores relevant issues surrounding the use of such knowledge. It also explores the ability of a layered abductive system to overcome shortcomings in the input; in essence, coming to a conclusion when given compromised input. In order to accomplish the purposes of the research, a system was implemented. This system presents
a method for semantic and pragmatic interpretation based on systemic grammar and layered abduction.

The purpose of Chapter II is to describe each of the major components of the system; classification, abduction and systemic grammar. When these items are used in conjunction with each other they can be used to perform semantic and pragmatic interpretation. This chapter can be viewed as a generic, high-level view of the representation and algorithms used in the system; a later chapter discusses the specific representations and algorithms used in the implementation. Chapter II provides the background knowledge and terminology necessary to understand later chapters.

Chapter III presents the work of other researchers in the area of abduction and natural language understanding, trying to identify the positive aspects as well as the possible pitfalls of each approach. This research effort is compared with the other approaches to help identify why this approach is a more appropriate and beneficial methodology to explore.

Chapter IV presents a more detailed discussion of the representations and the algorithms implemented in the system. It presents more clearly the terminology and the interesting operational aspects of this implementation of layered abduction.

Chapter V presents the CPU time required to understand the eleven sentences under several different circumstances to analyze the benefits of anticipatory and run-time top-down guidance. The chapter concludes with a discussion of how SAPI can cope with incorrect input and the results of SAPI coping with incomplete input.
Chapter VI discusses the experiment results in concrete terms. It takes an objective view of the computational advantages of including run-time guidance as well as reviewing the benefits that can be reaped by the inclusion of pragmatic expectations in the form of anticipatory guidance.

Finally, Chapter VII summarizes the contributions of this research and discusses future enhancements. It takes a more abstract view of the results of utilizing layered abduction on classification hierarchies and using context to guide processing. It explores the generality of the concepts presented.

The goal of this research was to make progress in the task of computerized natural language understanding. Although the work presented here is only a small piece of the information processing puzzle, it is hoped that the experimental results and insightful conclusions have contributed to the field of natural language understanding as well as all processing tasks that operate within a context and are abductive in nature.
CHAPTER II

Classification, Abduction and Systemic Grammar

2.1 Introduction

Many researchers have proposed that NLU involves some form of abduction or explanation-based reasoning [Charniak and McDermott, 1985; Josephson and Josephson, 1994; Charniak, 1986; Dasigi, 1988; Hobbs et al., 1988]. Therefore, NLU can be described as an abductive task. People perform this abductive task swiftly, yet engineered attempts to solve it have yet to achieve the same timely results. One possible reason is that the world is knowledge-rich and people absorb environmental information, organizing it and using it in order to enable efficient processing. Conceivably, the relatively poor performance of NLU systems can be attributed to their knowledge-starved operation and inadequate knowledge organization and representation schemes. The former leads to the inability to understand while the latter leads to inefficient knowledge access and evaluation.

This research presents a strategy to improve performance. It utilizes a grammar representation, called systemic grammar, and the domain independent computational techniques of classification and abduction. Classification and abduction are domain independent computational techniques for which generic control strategies and knowledge representations can be specified. Classification's knowledge organization enables large-scale pruning of a search space. Systemic grammar lends itself to such an organization. The
goal of abduction is to produce a conclusion that explains the given data. The task of natural language understanding can be viewed as a combination of these two techniques into a problem-solving strategy called *classificatory layered abduction*.

In this chapter, classification, abduction and layered classificatory abduction will be briefly explained. This is followed by a definition of top-down guidance and a discussion of systemic grammar. The chapter concludes with an example of semantic interpretation as layered classificatory abduction using both anticipatory and run-time top-down guidance.

### 2.2 Representation and Algorithm

The system developed as the computational model of NLU utilizes an approach that can be aptly described as *layered classificatory abduction*. It combines the knowledge organization and plausible hypothesis generation ability of classification with the problem-solving ability of abduction to create a system that can efficiently arrive at a conclusion. This computational framework has been used successfully in the domain of medical diagnosis [Josephson et al., 1985], but has been relatively unexplored in the domain of natural language understanding. To better understand this approach, it is necessary to analyze its various subcomponents.

#### 2.2.1 Classification

Classification is the process of finding the categories within a classification hierarchy that apply to the situation being analyzed [Bylander and Mittal, 1986]. Each node in the hierarchy represents a category, with the more general categories closer to the root node, and the most specific categories as leaves. During processing, the various categories are hypothesized as being able to account for the data. The goal is to output the most specific
and plausible hypothesis or hypotheses as the conclusion given the input data. The hierarchical structure enables class-subclass knowledge to be easily expressed. A generic control strategy for classification is Establish-Refine [ibid.] where, if a hypothesis (node in the hierarchy) is plausible, it establishes. Once established, it tries to refine itself by establishing any of its children. A node which finds its hypothesis to be implausible rules itself out which causes the entire subtree beneath it to also be ruled out.

### 2.2.2 Abduction

Abduction is defined as inference to the best explanation. Given a set of data, abduction infers a hypothesis, or set of hypotheses called a composite, that best accounts for the data (Figure 3) [Josephson et al., 1987]. What is meant by best is that the composite is more plausible than any other possible composite and has no superfluous parts. Other criteria to determine “bestness” include: how completely the composite accounts for the set of data, how consistent the hypotheses in the composite are, and if the composite has the least number of hypotheses of any similarly complete explanation. Other factors that influence the confidence in the composite are: how good the composite hypothesis is by itself, how much better it is than the alternatives, how much confidence there is in the accuracy of the data, and how thorough the search was for an alternate explanation.
A system developed based on a definition of abduction that requires the concluded explanation to be the "best" as in optimal, would seem to be problematic. To be able to conclude that a composite is the optimal explanation, it would have to be shown to be better than all other possible composites. In order to determine this, all other composites would need to be generated from the set of hypotheses and scored. This generation of all the subsets of hypotheses is intractable. (If there are $n$ hypotheses, then the number of composites to generate and score would be $2^n$. This would take exponential time.)

A related problem is that abductive conclusions are constructed at run-time since it is generally not possible to store all possible conclusions and simply retrieve them from memory. This time-consuming construction process requires examining the alternative hypotheses and putting the hypotheses together in a composite. Before the composite can be accepted, there needs to be some level of confidence in the composite; its explanation must be significantly better than the alternative composites. Thus, it would seem that all other composites must also be generated, which brings us back to the first problem.

Many methods of performing abduction are intractable. Generating a "best" explanation is intractable as it requires generating all possible explanations and comparing
them. However, if the term "best explanation" were to mean "the explanation that can account for as much of the data as is confidently and practically possible," then the intractability\(^1\) can be avoided. Furthermore, there are several strategies that can be used to aid in the composing and critiquing of the composite:

- Generate the best "local" explanation for a single datum. These local "best" hypotheses will be combined later into a global explanation.

- Delay hard decisions. If there are two or more competing hypotheses, and none is clearly better than the others, do not put any in the composite. This has a threefold effect. Time is not spent trying to make a hard decision, the confidence in the conclusion is not weakened by including a less confident hypotheses, and by stalling on the decision, the best of the competing hypotheses might be determined later by other problem-solving.

- Allow hypothesis interactions to help drive the decision making. Take the hypotheses that have been concluded, or that have a high level of confidence. Use these hypotheses as leverage against (or for) other hypotheses. This leverage can take the form of ruling out the hypotheses that are incompatible with the accepted set, raising the plausibility of those that the accepted set are related to, or seeking to refine the accepted set to attain a more detailed explanation.

- Accept a "best, but less than optimal" solution that meets some criteria of being "good enough."

Therefore, the "best" explanation will maximize explanatory coverage consistent with maintaining a high standard of confidence [Josephson and Josephson, 1994]. Complete coverage of the data is not necessary, nor does the conclusion have to be optimal. By searching for the clearly best local explanations, the composite explanation will explain as much of the data as possible with a low likelihood of error. Furthermore, the confidence

---

\(^1\) This definition and the composition strategies that follow remove the intractability from some abductive problems while others remain intractable. For a discussion on the types of intractable abductions see Bylander et al. (1991) and Allemang et al. (1987).
in the composite will not lessen with the inclusion of a hypothesis that is part of a hard decision.

Many processing strategies exist that accomplish the goals of abduction. In general, the strategy is: the hypothesis that explains a datum that no other hypothesis can explain must be included in the composite; it is essential to the explanation. Next, the hypothesis that explains a datum better than other hypotheses and with greater plausibility than a certain threshold value are included in the composite. Then, the composite must be checked for consistency (one hypothesis in the composite must not be incompatible with another) and made parsimonious. These steps are continued until all the data can be explained or until there is sufficient explanatory coverage. This strategy is just one that produces an abductive conclusion. The various strategies differ along several dimensions. To name a few of these dimensions: they differ according to their interpretation of the term "best" explanation, the amount of coverage mandated, the amount of time spent deciding which hypotheses are part of the composite and should be concluded, the hardware available, the representation of the hypothesis space, and the existence and type of interactions between hypotheses.

Layered Abduction

Layered abduction is the use of multiple abductive processes, where the conclusions produced by one abductive process become the data for the next. The passing of conclusions between the levels need not be strictly after a composite has been accepted, but can be done as each hypothesis is added to the composite, as a subset of hypotheses reaches a certain level of confidence, at predetermined time intervals, or when processing at one level stalls and would benefit from some outside guidance. This issue, though important, is
not part of this research and will not be addressed. The justification schema for layered abduction is illustrated in Figure 4.

Besides having bottom-up processing from lower level hypotheses to those at a higher level, top-down processing from a higher level to a lower level can occur. This downward flow of information can generate expectations to aid the lower level abductions, question ambiguous data, and help determine the level of plausibility of the higher level hypothesis. By combining bottom-up and top-down processing, a layered abduction mechanism can generate a partial conclusion at one level to leverage further problem-solving at the same level of inference as well as the neighboring levels.

\[
\begin{align*}
D_1 & \text{ is a collection of data} \\
H_1 & \text{ is the best explanation of } D_1 \\
\text{Let } D_2 &= H_1 \\
D_2 & \text{ is a collection of data} \\
H_2 & \text{ is the best explanation of } D_2 \\
\cdots & \\
\cdots & \\
\text{Let } D_n &= H_{n-1} \\
D_n & \text{ is a collection of data} \\
H_n & \text{ is the best explanation of } D_n \\
\end{align*}
\]

Figure 4: Layered abduction.

To perform layered abduction, the knowledge necessary to solve the problem must be organized such that the results at one layer can be used as input to another layer. For
example, imagine a medical expert system that determines the cause of a disease based on the symptoms of a patient. It might have a pathological stratum (a hierarchy of diseases and their processes) and an etiological stratum (a hierarchy of the cause of diseases). The input could be the symptoms of the patient which would then cause some disease or set of diseases to be hypothesized. This composite explains the symptoms of the patient in terms of their illness. The disease(s) and perhaps patient information is then the input (data) to the etiological stratum. This stratum explains why the patient contracted the disease by considering the patient’s diseases along with the patient’s medical history and other relevant patient information. In summary, from the symptoms, the disease composite was created, and based on this diagnosis, the cause of the disease was ascertained.

Classificatory Abduction

Classificatory abduction combines the knowledge organization and plausible hypothesis generation ability of the classification strategy with the problem-solving methods of abduction to attain a system that can efficiently arrive at a conclusion. This computational framework has been used successfully in the domain of medical diagnosis [Josephson and Josephson, 1994], but has been relatively unexplored in the domain of natural language understanding.

Layered Classificatory Abduction

Layered classificatory abduction is layered abduction where the hypotheses have been organized into classification hierarchies. To employ layered classificatory abduction in a domain requires the partitioning of the domain knowledge into multiple classification
hierarchies and a means for evaluating the plausibility of the hypotheses represented in the hierarchies.

Systemic grammar is one such example of a classificatory description of natural language as will be seen in Section 2.4 on page 27.

2.3 Top-Down Guidance

People constantly receive environmental input and become accustomed to the occurrence of various items. It is natural to become so accustomed to a situation that those various aspects are expected. The level at which a person is familiar with a situation will determine the amount of expectations he/she will have. This familiarity will lead to the expectation of certain ideas to be conveyed, events to occur, objects to be viewed, etc. For example, driving a car in a downtown city, a person expects to see cars, buildings, pedestrians, bikes, etc. If a person were to see a raccoon, he/she might do a double-take to make sure it was really a raccoon, not a figment of his/her imagination or a dog. A raccoon downtown is unexpected while the plausibility of a dog being downtown is greater. Just as people encounter visual expectations, they also encounter expectations when understanding natural language. Auditory expectations occur when people are understanding speech and letter or word expectations occur when people are reading text. An example of auditory expectations is discussed in a study by Bruce (1958). He had people listen to sentences presented in a noisy situation. Before they heard the sentence, they were told that it would be in some context. This study showed that people tend to interpret sentences in the terms of the suggested context. For example, if the input sentence referred to food, but the person was told it referred to sports, the person would repeat the sentence they thought they had
heard, and it would match the sports context. Therefore, given a context, the expectations of words and concepts of what would be heard guided the understanding process.

This knowledge of expectations is difficult to include in a NLU system. One reason for this is that the representation of the linguistic knowledge does not encourage the incorporation and use of expectation knowledge. By using systemic grammar, which is designed with a mapping between the pragmatic and semantic strata, this notion of expectations and top-down guidance is readily available.

The current system implements layered classificatory abduction with an option of having the pragmatic stratum provide both anticipatory guidance before processing begins and run-time guidance during processing. Pragmatic features, whether anticipated or determined at run-time, presume the existence of various semantic features. By knowing a pragmatic feature exists, it is also known that multiple semantic features exist. The basic computational advantage of including top-down guidance is that it can reduce the time spent traversing the systemic grammar hierarchy. It enables some of the natural language processing to occur in a top-down manner in conjunction with bottom-up processing. In essence, more knowledge is brought to bear on the problem in order to decrease the abductive processing and hence increase the efficiency.

Figure 5 represents an algorithm of how to provide top-down guidance in an abductive framework consisting of two layers. This figure combines the justification schema of layered abduction with the processing necessary for top-down guidance. The top-down guidance is represented by the italicized portions, where the first line in italics represents anticipatory top-down guidance and the rest of the italicized portion represents run-time top-down guidance. The basic idea of how to incorporate top-down guidance in
layered abduction between any two levels is to begin by placing those features that are expected or anticipated to be part of the conclusion into the composite, \( H_i \). Next, the input \( D_i \) is explained by a subset of all the hypotheses (\( H_{i, \text{all}} \)) available at level \( i \). This explanation is the best explanation composite \( H_i \). The hypotheses generated at level \( i \), \( H_i \), become the data at level \( i+1 \), \( D_{i+1} \). The system then generates a set of hypotheses at level \( i+1 \), called \( H_{i+1} \), to account for \( D_{i+1} \). These hypotheses will give downward guidance to level \( i \) in terms of features that \( H_{i+1} \) expects to be a part of \( D_{i+1} \). A feature expected as a part of \( D_{i+1} \) must also then be part of \( H_i \) (since they are the same thing). \( H_i \) is checked for consistency and the effects are propagated, accounting for more data and adding other hypotheses to \( H_i \). \( H_i \) is repetitively passed as \( D_{i+1} \) to level \( i+1 \) until either \( H_i \) accounts for all of \( D_i \), or until \( H_{i+1} \) can give no further guidance (can not add any more hypotheses to \( H_i \)).

---

**Figure 5:** Top-down guidance in layered abduction.

\[
\text{Seed } H_1 \text{ with anticipated hypotheses}
\]

- \( D_1 \) is a collection of data
- \( H_{1, \text{all}} \) is the set of all hypotheses at level \( 1 \)
- \( H_1 \subseteq H_{1, \text{all}} \) is the best explanation of \( D_1 \), but may be incomplete

**REPEAT**

- Let \( D_2 = H_1 \)
- \( D_2 \) is a collection of data
- \( H_2 \) is the best explanation of \( D_2 \)

\[
\text{For each } h \in H_2, \ h \text{ preselects } x \text{ where } x \in H_{1, \text{all}} \\
\text{let } H_1 = H_1 \cup x
\]

**Check \( H_1 \) for consistency and propagate the effects**

**UNTIL** \(((H_1 \text{ is complete}) \text{ or } (H_2 \text{ is stable}))\)
2.4 Systemic Grammar

The origin of systemic grammar is quite different from that of more familiar generative linguistic theories. Its roots are:

... in anthropology and sociology, not in mathematics or formal logic. The questions that motivated its development were not those of grammaticality or the acquisition of linguistic competence, but those of language as a social activity: What are the social functions of language? How does language fulfill these social functions? How does language work? [Winograd, 1983, page 273]

The original concepts of systemic grammar can be traced back to Bronislaw Malinowski (1884 - 1942), an anthropologist. In Malinowski's study of language, he proposed that language use depends on its social and cultural context, or the context of the situation. He also presented the concept that language is used to perform a function. In other words, aspects of language such as word choice and sentence structure reflect the effect that the statement is to have on the listener.

J. R. Firth (1890 - 1960) and Louis Hjelmslev (1899 - 1965), both linguists, took the concept of Malinowski and better incorporated them into linguistic theory. Firth's contributions were to develop a linguistic theory that links language and society and to devise a way to divide context into multiple levels such as phonological and grammatical. Each of these levels had "systems," or linguistic choices in a specific linguistic context. Hjelmslev's contribution asserts that a language can be realized at multiple levels, where the description provided at one level is further refined, given more definition at the lower level, and can be described in more abstract terms at a higher level. The various levels need not reflect the organizational criteria used at the other levels.

M. A. K. Halliday (1925 - ), a student of Firth's, revised the concept of system by making the choice of a system related to other choices, and these choices were functionally
related. He also developed a way to relate the various levels suggested by Hjelmslev by systematically relating the description statement of one functional level to the next.

To understand systemic grammar, and the implementation of the computational model that uses it, it is necessary to understand its basic concepts and terminology.

2.4.1 Basic Concepts

This section is meant to provide a quick explanation of the linguistic theory of systemic grammar. A more complete overview can be found in [Winograd, 1983].

**Feature**

The basic unit in a systemic grammar is the *feature*. A feature is a descriptive term for the name of a class, for example, declarative, interrogative, and finite are features. Since systemic grammar is described in terms of a classification hierarchy, the features are the nodes of the hierarchy. These nodes are arranged with the more general terms closer to the root and more specific terms closer to the leaves. The generality of the term determines its "delicacy" or specificity; delicacy increases from root to leaves.

Some other features of the grammar are future, singular-subject, negative, positive, and so on. The features are not independent of each other. A clause can not be positive and negative, or past tense and future tense. These features are mutually exclusive. To capture this organization, the features are arranged in "systems."
System and System network

A system represents the "gateway" to a set of mutually exclusive features, it is the parent of the mutually exclusive sibling features. For example, the features future, present, and past are part of the same system, Primary-Tense (see Figure 6). Since these features are related by a system, only one of them may be part of the description of the clause. It is possible that none of the Primary-Tense system choices will be part of the clause description. This is possible because in order to have traversed the grammar and arrived at the Primary-Tense system, the clause has to be temporal (see Figure 6). A sibling to temporal is modal, and they are both part of the system Deicticity. Since features of a system are mutually exclusive, if the clause is modal, then it will not be future, present or past, since it cannot be temporal.

To graphically display the relationship between features, a system network is used. The system network that matches the above discussion about Deicticity and Primary-Tense is shown in Figure 6.

```
Deicticity ——> temporal — Primary-Tense — future
              
modal — present

past
```

Figure 6: System networks of the Deicticity and Primary-Tense systems.

Primary-Tense and Deicticity are "systems." Primary-Tense is the "gateway" to the mutually exclusive features of future, present and past and Deicticity is the gateway to the
mutually exclusive features temporal and modal. The right side of a system is always represented by a sideways “T,” referred to as an XOR (exclusive-or) branch, where the features to the right of the “T” are mutually exclusive. To further refine a feature (increase specificity), a system must be traversed and a child feature must be established. Generally in system networks, system names exist but are not explicitly identified as they are in Figure 6. This is because all XOR branches are proceeded by a system making the labelling of the system immaterial; it is accepted knowledge that a system exists. During traversal of the network, the system names may be used to guide the processing, but are still not generally included in a diagram.

To enter a system, the previous or parent feature (or features) must be accepted as true. The parent feature(s) are referred to as the entry conditions to the system. Several different entry-condition/system network configurations exist. There may be only one feature as the entry condition for one system (Case A, Figure 7). In this case, only the feature (temporal) being true is necessary for a decision to be made between the competing siblings of the system (Primary-Tense). Another system network is represented in Case B. This case has one feature as the entry condition to multiple systems. If the feature (indicative) is true, then each of the systems (Deicticity and Indicative-Type) related to it represent a choice to be made between their own competing mutually exclusive children features. This type of system network is referred to as an AND branch and is graphically represented by “{.” A third type of configuration is represented in Case C. Case C has multiple features as entry conditions to a system. In order for a system choice to be made, all of the entry-conditions must be true. In the example, in order to have the choice to make between the children of Indicative-Other-Subject, both of the parent features of other-
subject and indicative must be true. This case is referred to as an AND join and is graphically represented by the "\)." The last case is represented in Case D. This case has multiple entry conditions (noun-head and substitute-head) for a system (Quantification), but only one of the entry conditions must be true in order for the system choice to be made. This system network is referred to as an OR join and represented by the "T." Any combination of these system networks is permissible.

Case A:

\[
\begin{align*}
\text{feature} & \quad \text{System} \\
\quad & \quad \text{Entry Condition} \\
\end{align*}
\]

\[
\begin{align*}
\text{e.g.} & \quad \text{temporal} & \quad \text{Primary-Tense} \\
\end{align*}
\]

Case B:

\[
\begin{align*}
\text{feature} & \quad \begin{cases} 
\text{System1} \\
\text{System2} 
\end{cases} \\
\quad & \quad \text{Entry Condition} \\
\end{align*}
\]

\[
\begin{align*}
\text{e.g.} & \quad \begin{cases} 
\text{indicative} \quad \begin{cases}
\text{Deicticity} \\
\text{Indicative-Type}
\end{cases} \\
\text{AND branch}
\end{cases} \\
\end{align*}
\]

Case C:

\[
\begin{align*}
\text{feature1} & \quad \begin{cases} 
\text{System} \\
\text{feature2} 
\end{cases} \\
\quad & \quad \text{Entry Conditions} \\
\end{align*}
\]

\[
\begin{align*}
\text{e.g.} & \quad \begin{cases} 
\text{other-subject} \quad \text{Indicative-Other-Subject} \\
\text{indicative} \\
\text{AND join}
\end{cases} \\
\end{align*}
\]

Case D:

\[
\begin{align*}
\text{feature1} & \quad \begin{cases} 
\text{System} \\
\text{feature2} 
\end{cases} \\
\quad & \quad \text{Entry Conditions} \\
\end{align*}
\]

\[
\begin{align*}
\text{e.g.} & \quad \begin{cases} 
\text{noun-head} \quad \text{Quantification} \\
\text{substitute-head} \\
\text{OR join}
\end{cases} \\
\end{align*}
\]

Figure 7: The various system structures.
The features discussed so far are referred to as system features since they are chosen within a system in opposition to other features. Another type of feature exists, called the *gate feature*. A gate feature is still a feature, however it results from any boolean combination of features and not from the choice of a system. In other words, the entry conditions to a gate feature are system features where all the boolean combinations of system features must be true in order for the gate feature to exist. There is no system choice involved. The graphical representation of a gate feature is in Figure 8. Notice the similarity between a gate feature and Case C in Figure 7. The difference is that a gate feature does not present a choice to be made between a set of features, as a system does. The gate feature may be an entry condition for other gate features, but there will never be an XOR branch to the right of a gate feature as will always appear to the right of a system.

![Figure 8: Gate feature structure.](image)

**Functional Analysis**

Systemic grammar is based on the idea that form follows function, hence the concept of "function" is vital to any discussion of systemic grammar. The sentence's purpose, significance, or intended effect on the recipient is its function, and this function will determine how the sentence will be constructed. A functional analysis of a sentence determines the relevance of the meaning to the social context. The grammar is designed and
developed with multiple functional dimensions which enables multiple functional analyses to be explored simultaneously. The purpose of having multiple functional dimensions is that each attempts to explain different properties of the sentence. For example, the goal of the functional dimension of TRANSITIVE is to create a logical representation of the event; the functional dimension THEME focuses on the goals of the speaker; and MOOD gives a syntactic functional analysis, explaining the constituent and word order. The analyses in SAPI are along the functional dimensions of MOOD, TRANSITIVITY, THEME and ERGATIVITY. Each of these analyses use a set of functions or roles in their explanation: MOOD has the roles Subject, Predicator, Residue, Mood, Residual, Finite, and Predicator; TRANSITIVITY has the roles Actor, Process, Senser, Phenomenon, Beta, Attribute, and Carrier; THEME has Theme, Rheme, Interpersonal and Topical; and ERGATIVITY has the roles Agent, Medium and Process. None of these lists of roles are exhaustive. A major portion of grammatical processing revolves around labelling the various constituents with roles and relating the various roles within a functional dimension to each other, as well as to roles of other dimensions. The various roles are related interdimensionally by “conflations” (unifications) and intradimensionally by “expansions” and “adjacencies” (these terms are explained in the section on Realization Rules). Hence, each linguistic constituent is actually being analyzed on multiple facets, so there will be multiple analyses per linguistic item.

An example of the multi-dimensional functional analysis is depicted in Figure 9. The clause “this dissertation was written by Julie Hartigan” is analyzed along the dimensions of MOOD, TRANSITIVITY and THEME\(^2\). The roles of Subject, Predicator

\(^2\) This analysis is based on Winograd’s (1983) analysis of “this gazebo was built by Sir Christopher Wren.”
and Adjunct all belong to the functional dimension MOOD; Goal, Action and Actor are part of the TRANSITIVITY dimension; and Theme and Rheme are the functions represented in the THEME functional dimension. The noun phrase "this dissertation" maps to the roles Subject, Goal and Theme, where each function is a member of a different dimension. Hence, the constituent as well as the clause has been analyzed in multiple functional dimensions.

<table>
<thead>
<tr>
<th></th>
<th>Subject</th>
<th>Predicator</th>
<th>Adjunct</th>
</tr>
</thead>
<tbody>
<tr>
<td>mood</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transitivity</td>
<td>Goal</td>
<td>Action</td>
<td>Actor</td>
</tr>
<tr>
<td>theme</td>
<td>Theme</td>
<td>Rheme</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 9: Functional analyses of a clause.*

**Realization Rules**

Up to this point, the discussion of systemic grammar has been strictly functional, but there needs to be a way to relate the functional specifications determined by the grammar to the structure of the sentence or clause. This is accomplished by realization rules. Realization rules are attached to features and specify the structural implications of that feature. The various types of realization rules used in this work are conflation, expansion, adjacency, lexification, and preselection.

*Conflation* is the term used in systemic grammar that means unification. A conflation occurs when two functions from different functional dimensions represent the
exact same linguistic item. When two functions are conflated, the features and subfunctions associated with each of the members of the conflation are combined. A resulting data structure of a conflation would be the union of the data structures of the conflated functions. It is this rule that enables a linguistic item to be related to more than one function. In Figure 9, the functions Subject, Goal and Theme (each from a different dimension) have been conflated and represent the functions of the linguistic item “this dissertation.”

*Expansion* is the rule that allows one function to be divided into multiple functions, or to look at it another way, multiple functions to be identified as one super-function. An expansion only occurs within a single functional dimension. For a specified dimension of analysis, one function may be realized by multiple other functions, all related to the same analysis. For example, in the MOOD dimension analysis, the function of Mood can be realized as the functions Subject and Finite. In the sentence, “The man is happy,” the Mood is “the man is.” Mood can then expand into the Subject “the man” and the Finite “is.” In summary, Mood expands into the functions Subject and Finite, all of which belong to the same functional analysis.

*Adjacency* is the rule that specifies that two functions are next to each other in the structure. A function may also be adjacent to the bounds of the sentence, or adjacent to the bounds of an expanded function. Adjacencies are limited to operating in only one functional dimension, not across dimensions. For example, from Figure 9, it is known that the Subject is adjacent to the Predicator, and the Predicator is adjacent to the Adjunct. Furthermore, it is known that the start of the sentence is adjacent to Subject and the Adjunct is adjacent to the end of the sentence. This way the bounds of the sentence have been determined as well as the order of the pieces in-between those bounds. It would not be
proper to say that the Subject is adjacent to the Action since they are from different functional dimensions.

*Lexification* enables a function to be related to a lexical word. It is said that the function lexifies to “x,” where “x” is a word in the sentence.

To best discuss the rule of *preselection*, further knowledge of the structure of the logigrammatical stratum is necessary. The entire logigrammatical stratum is not presented in one hierarchy, but rather a set of hierarchies. The clause, nominal, verb, adjunct, adjunct-phrase, conjunct, determiner, noun, prepositional phrase, preposition, and quantifier hierarchies are all part of this stratum. A hierarchy represents the description of a constituent. Preselections are used to relate the various hierarchies, or constituents, in terms of restrictions on their subconstituents. A feature in the description of one constituent may have a realization rule which has a function that preselects a feature for a subconstituent. A result of a preselection is the further definition of a subconstituent. An example of a preselection from the systemic grammar used in SAPI is: the feature “past” at the clause level has a realization rule that implies that for the clause to be past tense, the Finite must have as part of its description the feature “!past” from the verb hierarchy. In other words, Finite preselects !past. The feature (!past) for the function (Finite) is chosen from a hierarchy (verb) at a lower level of classification. This form of preselection is from one level to the next. Preselections can occur across multiple hierarchy levels. For example, the feature “pronominal-subject” has a realization rule that says the Head of the Subject must have as part of its description the feature “!third”, representing third person. (The case of first and second person pronominal subjects are handled by other features.) When the preselection is across multiple levels, as in this example, the feature that has the
preselection rule (pronominal-subject) is actually specifying a feature (\textit{third}) for a function (Head) at a lower rank, not a feature itself at the lower rank (no feature is being selected at the rank of Subject).

### 2.4.2 The Strata

Classification hierarchies – in the form of systemic grammars – are used to encode hierarchies of linguistic knowledge. The system that performs classificatory abduction employs a stratified hierarchical approach, assigning a stratum to represent socio-pragmatics information (pragmatic and sociolinguistic information) and another stratum to represent the logicogrammatical information (semantics, grammar, and some discourse information). Both strata are arranged as classification hierarchies (or sets of hierarchies in the case of the logicogrammatical stratum) and the realization rules define a mapping between them [Winograd, 1983]. The socio-pragmatic information is in the pragmatic stratum and the concluded features of the stratum will outline the information that the sentence was meant to convey, the significance of the sentence. The logicogrammatical information is in the logicogrammatical stratum (also referred to as the semantic stratum) and the concluded features of this stratum will outline the literal meaning of the sentence. The layer of form represents the syntactic structure of the sentence. This syntactic parse, for the purpose of the system developed, is assumed to have already taken place and the results of that analysis is the input to the system. The stratified approach is displayed in Figure 10.

The basic computational advantage of including the socio-pragmatic information is that a person’s expectations of what may be said can provide additional guidance through the semantic stratum. Processing occurs in both a bottom-up and top-down manner in order
to decrease the abductive processing and hence increase the efficiency. The system also has the computational benefit arising from the hypotheses being arranged hierarchically; the consequence being that the number of hypotheses being considered at any one time can be drastically reduced.

The arrows that point from an upper stratum to a lower stratum in Figure 10 represent "interstratal preselections." An interstratal preselection is identical to the preselections previously defined, but they reach from one stratum to another and link the social context to the language functioning in that context. A preselection from the pragmatic stratum can preselect any feature in the logicogrammatical stratum, in any hierarchy. In this way the strata are dependent on each other, they are linked by these preselections. Each strata has its own system networks and hierarchies that are suitable for its own level of analysis of language. The actual pragmatic stratum used by the pragmatic interpreter to provide top-down guidance, to verify the conclusions of the semantic interpreter, and to determine the significance of the sentence can be found in Appendix A on page 220.
2.5 High-Level Explanation of Operation

The basic operation of the system falls in the family of strategies that can be called confidence-controlled abduction. This family is so called because the order and termination of processing are presumably always determined by confidence levels. A confidence level represents the degree to which the hypothesis is believed to be part of the explanation. One member of this family is the "essentials first, leveraging incompatibilities" strategy which is discussed in Section 3.3.1 on page 57. The basic strategy is to account for all the input data by adding to the composite the relevant hypotheses in order of greatest likelihood.
The input to the system is the systemic grammar, a structural analysis of the sentence, and any anticipated logicogrammatical features. These are all assumed to be correct. The anticipated features are assigned the highest possible confidence value of highly-likely\(^3\), as they are highly-likely to be part of the explanation or understanding of the sentence. Next, the features in the logicogrammatical stratum that explain the structural data are assigned the highest possible confidence level of highly-likely. The status of highly-likely is propagated up through the parent nodes in the systemic classification hierarchy. This is possible since confidence values can only increase or stay the same as specificity decreases, or in other words, a parent feature must have a confidence value no less than any of its children. Also, any constraint imposed by a highly-likely feature on a subconstituent (e.g. by a clause feature on a noun phrase constituent) is highly-likely. All highly-likely features are included in the composite. If run-time guidance is enabled, the next step is to determine if any socio-pragmatic feature can account for a highly-likely logicogrammatical feature. If any such socio-pragmatic feature is found, all of its preselections of logicogrammatical features are included in the composite. These new highly-likely features have their status' propagated up through the hierarchy and the pragmatic stratum is again evaluated. When no socio-pragmatic feature can add to the explanation, processing returns to the logicogrammatical stratum.

At this point, all the highly-likely features have been determined. The next step is to make decisions between competing siblings. If one child is a default feature, it is put in the composite first. Default features are those that are assumed to be likely until proven otherwise. Whenever a node is put in the composite, its realization rules are posted as

---

3. All other strategies in the confidence-controlled abduction family have the highest possible confidence score being essential. The rationale for not having essential features is discussed in "When an Essential may not be Essential" on page 69.
constraints. Realization rules define the structure of the sentence in terms of surface and deep roles and are represented in Figures 11 and 12 by the shaded boxes. When an inconsistency occurs among these constraints, backtracking takes place. The current system uses dependency-directed backtracking if it can determine exactly which feature caused the inconsistency, otherwise it resorts to chronological backtracking. Additional hypotheses are placed in the composite until a complete and consistent path is made through the logicogrammatical stratum, where completeness implies that a choice has been made for each system traversed and consistent implies that no realization rules conflict.

2.6 Example

A system, SAPI, has been developed that uses a confidence-controlled abduction strategy to perform semantic and pragmatic interpretation on sentences of varying complexity and varying levels of anticipation. This section gives an actual example of the system performing layered classificatory abduction, with anticipatory and run-time guidance. The relevant socio-pragmatic information is described in Patten et al. (1992). Figure 11 is a fragment of the pragmatic stratum and Figure 12 is a minute fragment of the logicogrammatical stratum used in the implementation. The bold-type pragmatic features aided in the top-down guidance and the bold-type logicogrammatical features represent the complete and consistent path of features in the conclusion.

Assume a scenario where a person has called a travel agent and asked for a seat on a flight. It is reasonable to assume that the person is expecting the travel agent to respond to his request with a report of the search status ("I'm still looking...", "I found...", or "I have a..."). This expectation can be summarized as the person expecting that the next
sentence uttered by the travel agent to be agent-oriented. Agent-oriented in the pragmatic stratum preselects speaker-subject in the logicogrammatical stratum. The feature speaker-subject is assigned the confidence value of highly-likely and is added to the composite. The ancestor of agent-oriented is service which does not give any additional top-down guidance. The predecessors of speaker-subject, which are interactant-subject, indicative, and finite-clause, become highly-likely and are added to the composite. The travel agent says, “I am not finding anything then.” Assume the syntactic parser provides the information that the sentence is declarative and negative, making these features highly-likely and part of the composite. The predecessors of these features are already part of the composite, so inheritance does not add anything to the composite. At this point, the composite receives run-time top-down guidance. The words “finding” and “anything” can only be accounted for by the socio-pragmatic feature of search-failure. The predecessors of search-failure are unsuccessful-so-far, search-in-progress, search and service, which are all assigned the confidence value of highly-likely. The first two of these pragmatic features have preselections into the logicogrammatical stratum; negative, present and unmarked-declarative-theme. These logicogrammatical features along with any of their predecessors not already included in the composite (temporal) are added to the composite. The realization rules are posted to check for inconsistencies, but none occur. The pragmatic stratum can give no more guidance, but the logicogrammatical stratum still needs to be completed. The AND joins are evaluated, which adds neg-ind-finite to the composite. At this point, nothing more can be determined to be highly-likely with the available knowledge. No more run-time top-down guidance is possible and the logicogrammatical stratum is incomplete. Since a complete path is required for termination, the likely-phase is started. To complete the logicogrammatical stratum, a decision must be made as to whether
the sentence is marked- or unmarked-negative. Since the unmarked-negative is the default feature, it is included in the composite. The composite is checked for inconsistencies, and none exist. The AND joins are evaluated, which adds reduced-negfinite to the composite. The realization rules are checked for any inconsistencies, and none exist. Therefore, a complete and consistent path has been created through the logicogrammatical stratum and the semantic interpretation is complete.

A complete path has been abduced using the structural analysis as well as the anticipatory and run-time top-down guidance from the pragmatic stratum. If the pragmatic stratum had not been used to guide the processing, then the determination of the subject, tense, and type of declarative theme of the sentence would not have been so readily determinable and would have required additional processing time, hence decreasing the overall system efficiency. Also, by using the classification hierarchy, putting a feature into the composite ruled out that feature's siblings and all of their children.
Figure 11: A small portion of a sample pragmatic stratum.
Figure 12: A small portion of a sample logicogrammatical stratum.

2.7 Conclusion

Systemic grammar and layered classificatory abduction are promising representations and strategies for performing NLU. Efficiency is attained by using systemic grammar as the knowledge representation because there is a hierarchy of hypotheses. In a hierarchy, ruling out one hypothesis rules out all of that hypothesis' descendants, hence pruning the search space. Further efficiency is attained by mapping contextual information directly to features in the systemic grammar, hence reducing the number of hypotheses needing to be
considered. Furthermore, the socio-pragmatic knowledge available in the context makes the system sensitive to social and contextual issues.

Systemic grammar and classificatory abduction are just one of many representations and algorithms proposed to perform NLU. The next chapter will discuss the approaches to NLU and abduction that other researchers have utilized.
CHAPTER III

A Survey of Related Works

3.1 Introduction

The task of natural language understanding can be decomposed into many subtasks. The system implemented as part of this research performs semantic and pragmatic interpretation by way of an abductive strategy using context as a guide. This chapter reviews some previous work in natural language understanding and analyzes different techniques and methods used to represent context as part of the processing. The chapter concludes with a discussion of some different abduction techniques. The ones discussed were chosen because of their differing interpretations of the term “best” and their different degrees of deliberative and explicit processing to generate hypotheses and synthesize composites. The degree of deliberative processing reflects the amount of serial, conscious problem-solving that occurred. The degree of explicit processing reflects the extent to which alternate composites were created and considered.

3.2 NLU Systems

Attempts to develop natural language processing systems dates back to the 1940s. At that time, computer scientists thought it would be a simple task to perform machine translation, “...that the principal obstacle would be incorporating a full dictionary of the two languages,” [Aikin et al., 1982, page 234]. Though the dictionary would not solve all the
problems of machine translation, the remaining obstacles seemed trivial. These additional problems included the syntactic structure variation among languages, idiomatic expressions, one word having multiple meanings according to context of use, and some words in the source language having no semantically equivalent word in the target language. These problems still plague the task of natural language understanding and all of its subareas, however progress has been made to solve or reduce their significance. This section first discusses a few systems that perform some aspect of natural language understanding. The systems selected for inclusion in this discussion either used systemic grammar to represent their linguistic knowledge or used an abductive strategy to arrive at an explanation. The last part of this section discusses different techniques that have been used to allow context to influence processing.

3.2.1 Systemic Grammar Parsing - Kasper

Robert Kasper (1988) developed a “...general parsing method using a declarative representation of SG [systemic grammar]...” whose purpose was “... to determine to what extent a grammar that is adequate for text generation can also be used for text analysis.” This parser was developed and tested using Nigel, a grammar used in a text generation system [Mann and Matthiessen, 1983].

The conceptual design of the parser was based on Functional Unification Grammar (FUG) [Kay, 1979]. FUG is a notation that enables system feature choices of the systemic grammar to drive the structure realizations. Thus, the grammatical description is a consequence of the feature choices. Each linguistic object is represented by a functional description (FD) which contains a list of attribute/value pairs. FDs can be unified (conflated) to combine the information represented by two FDs, as long as they are
compatible. FDs can also contain disjuncts. Nigel's systemic grammar was compiled into FDs to accommodate text analysis. The result is that Kasper's parser operated on a grammar representation that was only based on the systemic grammar that Nigel used for generation.

The implementation of the parser was based on the PATR-II system [Sheiber, 1984]. PATR-II is basically a two stage parser where the first stage takes the input sentence and develops a constituent structure by using a chart parser and a phrase structure grammar. This structure is the input to the second phase which develops a functional structure based on the unification of feature structures.

To summarize, the parser developed by Kasper uses the knowledge representation scheme of FUG and the processing strategy of PATR-II. Kasper (1988) concludes by posing the question, “How much context needs to be encoded in order to perform effective analysis of sentences?” This question can be posed at any level of NLU and the research presented here proposes that at the level of semantics, even a minimal amount of context is beneficial.

3.2.2 TACITUS - Hobbs

Jerry Hobbs developed a system called The Abductive Commonsense Inference Text Understanding System (TACITUS) [Hobbs et al., 1990]. TACITUS is a system for interpreting natural language texts, and in particular it interprets malfunction reports. It is intended to be a domain- and application-independent system.

TACITUS performs its task by producing logical forms in first-order predicate calculus and backward-chaining across them. It creates an abductive conclusion or explanation by determining the best subset of predicates that can explain the sentence. A backward-chaining algorithm results in an abductive conclusion composed of the least
costly predicates that were assumed in order to generate a proof corresponding to the "explanation."

In this system, "best" is determined by the sum of the costs that are associated with each predicate. This cost should be minimized. The cost is based on the specificity of the predicate (the more specific the predicate, the higher the cost) and what semantic contribution the predicate offers to the overall proof. In creating the interpretation of a sentence, it is important to weigh the specificity of a single predicate against the semantic contribution made by the combination of several less specific predicates. It is possible that a single, highly specific predicate is cheaper to conclude than some number of less specific predicates that explain the same text.

The system makes three assumptions:

1. Goal expressions are assumable and can be stated as predicates, at varying costs.
2. Assumptions can be made at any level of specificity. The greater the specificity, the greater the cost of the assumption, the greater its explanatory power, and the harder it is to conclude.
3. Natural redundancies found in natural language can be exploited for more economical proofs, using them to leverage against ambiguities.

The basic operation of TACITUS is to accept the input text and express it in its logical form. The sentence is then interpreted with respect to a knowledge-base that reflects the shared knowledge of the speaker and hearer, or in the case of the system, the knowledge that can be assumed from previously interpreted sentences. The system attempts to resolve
four pragmatic problems: syntactic ambiguity, metonymy, references, and compound nominal interpretation. Assumptions are made as necessary. The system's goal is to prove some subset of predicates that represent the text and that which is not proven is assumed to be "new" information.

The differences between TACITUS and SAPI are that TACITUS has no notion of a predicate explaining a datum, but strives to create a set of predicates that together explain the entire sentence. SAPI tries to account for each datum and the local "best" hypotheses are combined to create a global explanation (assuming they are consistent). SAPI's strategy makes composite creation more efficient because the overall abductive problem is divided into smaller problems which are individually conquered and the individual explanations unite to create a global explanation. This approach seems more orderly and efficient than tackling the entire problem at one time. Another difference is that TACITUS does not make a distinct separation between its semantic and pragmatic knowledge.

3.2.3 Wimp - Charniak

Eugene Charniak (1986) created a system called Wholly Integrated Marker Passer (WIMP) which is a language comprehension program. This program creates an explanation of a story, where explanation is defined as assigning motivations to the characters based on their actions. It performs this task with a technique called marker passing or spreading activation. This is basically a "...breadth first search in an associative network to find connections between concepts, in the hope that such connections will suggest explanations." The "important" words in the text (the nouns and verbs) appear as nodes in the network and are connected by links represented in first order predicate calculus. Of the many existing network indexing method, WIMP's method is to index an activity by the
objects that are used in the activity. For example, the activities of “eating cereal” and “drinking” are both related to the object “milk.” Finding a path in the network between two words from the story represents the backbone of the proofs. The idea is that these paths lead to “... the kind of inferences which are normally thought to characterize understanding.”

WIMP can accept an unparsed sentence as input, but for the purposes of discussion assume the sentence has been parsed. The “marker passer” component creates the paths between the nodes representing the words by passing markers between them. Markers are 5-tuples containing the: (1) mark’s origin, (2) last node visited, (3) last link visited, (4) creation “date”, and (5) strength (called “zorch”). The zorch of a mark decreases as it is passed, and when it falls below one, it is no longer propagated. When an intersection at a node occurs, the paths from the two origins are recreated and are all assigned a “path zorch” based on a set formula. A path between two words represents an abductive hypothesis and is considered an outline or suggestion of how the words in the story may be conceptually related. More than one path may exist between two words.

WIMP next evaluates each path in three stages in order to determine which is “best” and should be believed. Some paths are immediately thrown out of consideration, like those that connect objects of the same class. The first stage is to determine if an abductive assumption associated with the path is internally consistent as well as consistent with all that is already known. The second stage is to verify that the path’s assumptions have predictive power. This value is roughly the number of true predictions minus the number of assumptions, and must be greater than or equal to zero for the path to be believed. The third stage is to verify that no other path of equal predictive power exists that makes incompatible assumptions. If this occurs, their assumptions and predictions are compared.
If their assumptions are the same, the choice of the path to believe is arbitrary. If the assumptions are different, the one with the greatest predictive power is believed. If there is a tie for greatest predictive power, the decision is delayed until more evidence is found to enable the decision.

The problem with WIMP is that its scalability is questionable. It uses a small database of words and even with such a limited database, the marker passer component creates an average of 40 paths, of which half are eliminated because they explain nothing worthwhile and only one-tenth of the remaining paths are any good. This proliferation of paths will increase exponentially as the database size and connections between nodes increase.

3.2.4 Implementation of Context in NLU Systems

The implementation of context in NLU systems has taken several forms. As early as 1949, Weaver had an idea to utilize context. His idea was to constantly have a “window” of words available for analysis. This window should be of size $2N+1$, where $N$ is some function of the number of words. Given a large enough $N$, the size of the window should allow enough context to be viewed to determine the correct translation of the word in the middle of the window [Aikin et al., 1982].

Since those early days and their ideas, more methods have been created to incorporate contextual knowledge. These methods can be basically divided into two groups: those that allow semantics to aid the syntactic processing and those that allow pragmatic knowledge to aid semantic processing. Though connectionist techniques
actually fit into both of these subdivisions, they will be discussed separately because of their different representation.

Examples of techniques that utilize context when parsing include case grammars and rule-by-rule parsing. A case grammar semantically relates the noun phrases to the verbs and adjectives in the sentence. It builds semantically relevant syntactic relationships represented by cases. The goal of case grammars is to extract the meaning of a sentence. Sentences with the same meaning but different surface structures should have the same case grammar structures. Case grammars aid in the parsing by applying selectional restrictions on the possible interpretations of the words.

Rule-by-rule semantic interpretation parsing allows the syntax and semantic components to maximally interleave their processing. In this form of parsing, the syntactic parser does not form whole constituents before the semantic analyzer is called to judge its semantic validity. Every syntactic rule is paired with a semantic rule, and when the syntactic rule is applied, the semantic rule is also applied. If the semantic rule can not create a semantic interpretation, the constituent is not completed and the syntactic parser backtracks, assuming that the syntactic parse passed to the semantic analyzer was incorrect. The semantic component works to resolve ambiguities after the application of each syntactic rule by bringing semantic knowledge to bear as guidance to the syntactic processing at each step.

Examples of methods to allow for contextual guidance from the pragmatic level to the semantic level includes scripts and register. Scripts describe a stereotypical sequence of events for a situation. These events can prime the system to expect their occurrence. The problems with scripts include determining which script to use, how to acquire the script
information, how to handle an atypical event, and knowing when to change scripts. *Register* [Halliday, 1978] is a specialized language that is dictated by the social context. The register of a situation is based on the relationship of speaker to hearer (tenor), the channel of communication (mode), and what is discussed (field). This register knowledge links the social context to the language functioning in that context. Register knowledge is the method used in SAPI to allow context to guide the semantic processing.

The connectionist camp has attempted to incorporate context in various ways. Elman (1988) proposed a three layer network. At level one are two separate sets of nodes: the input nodes and the context nodes. These nodes provide input to the hidden layer of nodes. The hidden layer's output is fed to the output layer and is copied to the context layer. This way the context nodes have the value of the hidden layer at the previous iteration. Jordan (1986) proposes a similar network, however the context layer receives input from itself and the output layer, rather than just the hidden layer. Mozer (1988) proposes a four layer network consisting of an input, context, hidden, and output layers. In this architecture, the context layer at time $t$ is a combination of the input at time $t$ and the context layer at time $t-1$. Basically, the difference in these three approaches is the input to the context layer; only from the hidden layer, from the output layer and the context layer of the previous iteration, or from the context layer of the previous iteration and the current input layer.
The purpose of all these approaches is to somehow retain the state information, i.e. save the relevant information from the history of the previous inputs. This way the context is constantly being maintained and updated.

3.3 Abduction systems

The information processing task of *abduction* is inference to the best explanation. Given a set of data, abduction infers a hypothesis, or set of hypotheses called a composite
hypothesis, that best accounts for the data [Josephson and Josephson, 1994]. Abduction is allegedly used in human story comprehension [Charniak and McDermott, 1985], perception [ibid.], medical diagnosis, word disambiguation [Dasigi, 1988], theory formation [Thagard, 1988] and theory decision making [Tanner et al., 1991]. Abduction can be seen implemented in several problem solving systems, for example, RED [Josephson et al., 1985], INTERNIST [Miller et al., 1982], ECHO [Thagard, 1989], QUAWDS [Weintraub, 1991] and ArtRec [Fox, 1992]. Several abduction processing strategies will be discussed briefly. These strategies were chosen for discussion because of their differing interpretations of the term “best” and their different degrees of deliberative and explicit processing to generate hypotheses and synthesize composites. The degree of deliberative processing reflects the amount of serial, conscious problem-solving that occurred. The degree of explicit processing reflects the extent to which alternate composites were created and considered.

3.3.1 Tractable Abduction - Josephson et al.

Several versions of an abductive strategy have been developed at The Ohio State University, with many of them having been implemented in a system to test out their design ideas. This family of versions I will refer to as Peirce. The latest, non-parallel version of Peirce is a form of tractable abduction and the strategy is called “essentials first leveraging incompatibilities” (EFLI) [Josephson and Fox, 1991] and is implemented in the tool called PEIRCE-IGTT [Josephson et al., 1991]. It is this version that will be described.

In EFLI, the definition of the “best” explanation is the one that is consistent, parsimonious, confident in what it explains, and has maximal explanatory coverage. A trade-off exists between the two concepts of maximal explanatory coverage and
confidence. Increasing the explanatory coverage of the composite can be accomplished at
the risk of lowering its confidence. A composite with a high degree of confidence may lack
explanatory coverage. Another factor influencing the amount of coverage is time; as
processing time increases, the explanatory coverage can increase. Time is costly, though.
These factors result in an overall goal of: in the minimum amount of time, maximize the
explanatory coverage while maintaining a high degree of confidence.

The basic processing strategy identifies islands of certainty (hypotheses of greatest
confidence) and then has the composite grow opportunistically, only adding to the
composite the hypotheses that have high confidence values. The strategy operates on a set
of data that needs to be explained and is also given hypotheses to explain the data. Each
hypothesis has an initial confidence score, a list of data that it can explain, and a list of
hypotheses with which it is incompatible. The first step is to find all the essential
hypotheses. These hypotheses are indispensable to the explanation of a datum. Place all
such hypotheses in the "believed" set. Eliminate from the set of hypotheses those that are
incompatible with the members of the believed set. This elimination may cause additional
hypotheses to be essential. Continue adding essential hypotheses to the believed set and
eliminating the incompatible hypotheses until no additional hypotheses are essential. At
this point, determine the explanatory coverage of the believed hypotheses.

The second step considers the unexplained data. If no unexplained data exist, then
the processing stops. Otherwise, the hypotheses that are clear-bests are added to the
believed set. A clear-best hypothesis has a confidence score greater than some threshold
and also greater, by some margin, than all of the confidence scores of the alternative
hypotheses that can explain the same datum. The confidence value of any hypothesis
incompatible with a clear-best is significantly lowered. More clear-bests may be identified as scores of explanatory rivals are downgraded. Continue adding clear-bests to the believed set and altering scores until no additional hypotheses are clear-bests. Now determine the explanatory coverage of the believed hypotheses.

At this point, adding any additional hypotheses to the set will lower its confidence level. If additional explanatory coverage is desired and the loss in confidence is acceptable, the third step is to identify the hypotheses that are weak-bests. Weak-bests can be viewed as intelligent guesses. They are the hypotheses that explain an otherwise unexplained datum better than any other hypothesis, but not well enough to be a clear-best. If two weak-best hypotheses are incompatible, than neither of them is to be believed.

The output of the process consists of the confident explanation (the essential and clear-best hypotheses), the less confident explanation for a portion of the remaining data (the weak-bests), and the unexplained data with the unresolved set of potential explainers. The believed set of hypotheses is consistent, parsimonious, explains its set of data better than any other possible composite, and has maximal explanatory coverage with a high degree of confidence.

A system utilizing EFLI will be an explicit and deliberative system, since it serially considers each hypothesis in turn, seeing if it can account for the data and to what extent. Some form of the Peirce algorithm was used in RED [Josephson et al., 1985] and [Smith et al., 1985], QUAWDS [Weintraub, 1991], TIPS [Punch, 1989], ArtRec [Fox, 1992] and PATHEX [Smith et al., 1987].
3.3.2 General Set Covering - Peng and Reggia

Peng and Reggia (1987) present another way to perform abduction which is also deliberative. Their approach is to use Bayesian probability and the General Set Covering model to perform medical diagnosis. The goal of this system is to arrive at a best explanation of the symptoms given the diseases, where "best" is defined as the composite hypothesis with the maximum likelihood that can account for all the symptoms.

To determine this best explanation, disease hypotheses are evoked by being causally related to the manifestations. This knowledge is explicitly represented in a causal network. The evoked diseases are then scored or instantiated by using probability calculus. The two probabilities required are the probability of the demographic epidemiology (the probability of a disease occurring given the a priori probability of the disease occurring in that region with no knowledge of the manifestations) and the probability of the disease given the manifestations. The goal of the system is to create a composite of disease hypotheses that is of maximum likelihood (greatest probability) that can account for all the symptoms.

This abductive algorithm is explicit since each hypothesis must be considered in turn in order to create the most probable composite hypothesis set. This is performed serially, hence this is a deliberative process. Furthermore, Peng and Reggia assume that the disease hypotheses are mutually independent which means the conclusion of one disease will not change the probability of another being hypothesized.

3.3.3 Belief Nets - Pearl

Another abductive technique is to represent the data and hypotheses in a belief net or causal network and compute the probabilities of the hypotheses with Bayesian probabilities. This
is the approach of Judea Pearl (1987) to perform medical diagnosis with abduction. All the data (manifestations) and hypotheses (diseases) are arranged in a belief net with arcs connecting the hypotheses with the data they can account for. Other arcs exist between hypotheses in order for the hypotheses to influence each other either in the form of inhibitions (mutually exclusive hypotheses) or excitations (causally associated hypotheses). Three forms of probabilities are used in the system: evidential probabilities (the probability of a symptom given a disease), causal probabilities (the probability of a disease given a symptom), and the probability of the disease itself.

The system operates by activating the existing manifestations. The nodes of the hypotheses connected to the manifestations are activated and compute their belief using the probabilities mentioned earlier. The hypotheses' nodes connected to the activated hypotheses via an excitation link also compute their belief. These belief probabilities are propagated through the network. Once the belief net reaches quiescence (which relies on their being no cycles in the network), all hypotheses with sufficiently high belief scores are considered true, all others are considered false.

This technique defines “best” as the set of hypotheses with a “high-enough” probability. By definition of the method, the hypotheses will have maximal likelihood, but may not necessarily account for all the data. This approach can be viewed as somewhat deliberative. It is not completely deliberative like the EFLI and the General Set Covering approaches because it does not consider each hypothesis in turn; all hypotheses are computing their probability simultaneously, and recomputing as necessary. The processing is explicit since it evaluates the probabilities and propagates them throughout the network; they are not precompiled.
3.3.4 Neural Networks - Thagard

A final abductive technique that will be discussed is that proposed by Paul Thagard (1989) and is embodied in ECHO. This version of abduction is a bit different in that the hypotheses are divided into various sets, where each set of hypotheses supports a certain theory. Each set can have hypotheses that explain the same data. The current systems implemented have been limited to two theories, hence two hypothesis sets.

Thagard's approach appears to be quite similar to Belief Nets. The difference between the two approaches is that Thagard creates nets using a connectionist network instead of a belief net and the links between the nodes are weights. In Thagard's connectionist networks, hypotheses and data nodes are both explicit and hypotheses are connected to the data which they can explain. Links also exist between hypotheses if the hypotheses are incompatible, which results in an inhibition link, or if the hypotheses are associated, which results in an excitation link. The weights on the links are determined by the level to which the hypothesis can cause the data or the degree of incompatibility and association between hypotheses. The network operates by propagating activations through the network until the network stabilizes. Any hypothesis that has a high enough score is part of the explanation and the rest are ruled out.
Belief nets are the abductive strategy used in ECHO. ECHO is a tool that has been used to perform theory formation to determine which theory is more reasonable given facts and findings and also in legal reasoning to determine the verdict of guilty or innocent.

This technique does not perform an "exhaustive search" for the plausible hypotheses to account for the data. This is a direct result of the way the knowledge is encoded. All the knowledge in the system is already available. Search is not performed, in the typical sense of the word, as exhaustive search usually means explicitly considering each node in turn and evaluating it rather than propagating weights through the network and selecting the active nodes. Therefore, this approach is far less explicit in its processing. Lastly, this approach is almost non-deliberative; the time in deliberation is the time it takes to pass the activations through the trained network.
3.3.5 Parallelizing Abduction

Though all the deliberative abductive techniques have been presented as serial operations, any of them could be parallelized in order to make the processing less deliberative and more efficient. For example, the tractable abduction of EFLI can be parallelized by having each hypothesis and datum assigned to its own processor. The assembly phase has three types of hypotheses generated, those that are essential, clear-bests, and weak-bests. Each of these generations must be performed in a serial fashion, with essential being determined before clear-bests, and clear-bests being determined before weak-bests. This ordering needs to be maintained to minimize backtracking. Though each hypothesis ranking level must be performed in a serial order, the computation of all of one rank can proceed in parallel. The parsimony phase can also be performed in parallel with the same distribution of hypotheses and data [Josephson and Josephson, 1994].

A high-level view of the basic algorithm to parallelize the EFLI strategy follows. First, have all the processors with a hypothesis send a message to the processors with a datum that the hypothesis can account for. If the data's processor only receives one message, then that hypothesis is essential and it sends to that hypothesis' processor an "essential" message, else it sends an acknowledgment to all the hypotheses from which it received a message. In one cycle (hypothesis processors communicating to the data processors and the data processors responding), all the essential hypotheses have been determined. In the next cycle, the hypothesis processors send messages to the processors with a datum that they can explain with a value that specifies to what extent the hypothesis can account for the datum. The datum's processor returns a message to all the hypotheses from which it received a message. If the datum has already been accounted for by an
essential, then all the hypotheses receive a message that they are superfluous. Otherwise, the hypothesis with the greatest plausibility receives a message that it is the clear-best while all the others receive a message that they are superfluous. In the final cycle, which is optional (including weak-bests in a composite may weaken the confidence in the composite), the processing proceeds in a similar fashion to the second cycle, however now a hypothesis is superfluous if the data it can account for has already been accounted for by an essential or a clear-best. (See Goel et al., 1988, for further discussion). This way the entire processing takes a maximum of three cycles to maximize the explanatory coverage while maintaining a high degree of confidence.

3.4 Conclusion

This chapter presented the work of various researchers in the field of natural language understanding and the field of abduction. Also discussed were various ways in which context has been represented and utilized. In each section discussing an abduction implementation was a statement of how deliberative and explicit the method was. This information provides background for Chapter VII where there is a rebuttal to a claim made by Noam Chomsky which was that natural language understanding cannot be abductive. Before being able to present the rebuttal, the system SAPI that performs natural language understanding with layered abduction and top-down guidance will be discussed in detail along with its results. These are the topics of the next three chapters.
CHAPTER IV

SAPI - A Semantic And Pragmatic Interpretation System

4.1 Introduction

Chapter II presented layered classificatory abduction, top-down guidance and systemic grammar at a high level. The purpose of this chapter is to produce a more detailed and specific explanation of how these items can operate together. SAPI tests the hypothesis that the layered classificatory hierarchies of systemic grammar and the processing strategy of layered abduction can be successfully combined and used to perform semantic and pragmatic interpretation. Furthermore, SAPI explores the issues surrounding the inclusion of context to aid in processing.

This chapter identifies the input, output and the abduction strategies that are used to perform both semantic and pragmatic interpretation. The utilization of run-time and anticipatory guidance is discussed. The explanation of the abductive strategy used in the semantic stratum is broken down into two ordered subtasks; the task of placing only the features with the highest confidence value into the composite and the task of placing additional features in the composite in order to have complete explanatory coverage.

The entire system is considered a layered abduction system because abduction occurs at both the semantic and pragmatic levels where the input (the data to be explained)
by the pragmatic level is the composite explanation generated at the semantic level. Each of these two layers utilizes a different abductive strategy, described in this chapter.

The final section of this chapter is at the level of implementation and shows the inputs and outputs of the two interpreters.

### 4.2 Input Sentences

The input sentences to this system are the same as those sentences that were generated in Patten et al. (1992) which were based on the actual utterances of a human travel agent. All the sentences were generated by the same grammar that was used to perform the understanding, without modification. The context is a travel agency; the setting is that it is December 23rd and the Client wishes to travel on the 26th of December, for which reservations are difficult to acquire. The conversation is presented in Figure 15.
A: May I help you?
C: Do you have any flights to Miami on the 26th?
A: How many seats are you looking for?
C: One.
A: What time can you leave?
C: Some time in the afternoon.
A: Let me look... I’m not finding anything then... Can you leave earlier?
C: If I have to.
A: I’ve got a seat on an 11:00 flight on Treetop Airlines.
C: That’ll be good.
A: When can I bring you back?
C: On the morning of the thirtieth.
A: Well, all I’m showing is a 10 p.m. flight.
C: Do you have anything the night before?
A: I can put you on that 10 p.m. flight.
C: That’ll be okay.
A: The round-trip fare will be $295.
C: Okay.

Figure 15: The conversation.

The system can be viewed as operating as if it were the client trying to understand the sentences produced by the agent. The agent’s first sentence is a typical opening line. As such, it is assumed that the system’s or client’s understanding would be uninteresting (the expectations of hearing “May I help you?” or some other phrase with the same intended meaning are so great, that the phrase is often ignored or parsed with highly-compiled knowledge). That leaves ten sentences for the system to understand. One additional sentence was included in the study that the grammar can generate, so it should be able to understand. Though the increase in the number of sentences seems trivial (from 10 to 11), it does represent a 10% increase. The more sentences understood, the more credible the system appears. The eleventh sentence is: “When can you leave?”
4.3 When an Essential may not be Essential

An essential hypothesis is one that is indispensable for explaining some datum. In general, an essential hypothesis is added to the composite from which it is never removed. It is essential to the explanation of the data to include that hypothesis in the composite, without it the composite's explanation will not have maximal coverage with the highest degree of confidence. In NLU, it may be difficult to claim that a certain feature is essential. So, why is there a problem with essentials? When is an apparently essential hypothesis denied an "essential" confidence value?

A problem for essentials occurs when the input is assumed to be correct. Any hypothesis that solely accounts for a datum must be essential. What if the input has conflicting data? This creates the problem of incompatible essential hypotheses which would be a serious anomaly and would cause processing to suspend\(^1\).

In NLU systems that utilize top-down guidance and bottom-up processing, conflicting data can easily occur. For example, if top-down guidance predicts \(h_1\) to exist at level(i), and the bottom-up processing requires \(h_2\) at the same level, and \(h_1\) is incompatible with \(h_2\), either the top-down or the bottom-up processing must be incorrect. If both forms of input are assumed to be correct, then \(h_1\) and \(h_2\) would be conflicting essential hypotheses. The system needs to have a means by which it can resolve these conflicts.

The solution used in SAPI to resolve the conflicting input problem is not to assume that the input is correct. With all the input being suspicious, no hypothesis can be

---

\(^1\) Other approaches to handling incompatible essentials are to continue the problem-solving process in hopes of either lowering the confidence values of the "essential" hypotheses or raising the values of the hypotheses that compete with the "essential" hypotheses so they are not so vastly better than the alternatives, or to test the data for correctness.
considered essential because any datum might be "noise" or be a misinterpretation. The composite will contain hypotheses of great plausibility, but never an essential hypothesis. The hypotheses of greatest plausibility are the islands of greatest certainty around which other hypotheses will be based. As a result of there not being any essential hypotheses, there can be no ruled-out hypotheses, for any hypothesis may eventually be part of the composite if the islands of certainty change. These restrictions result in SAPI assigning the confidence values of highly-likely, likely, and unlikely.

With this in mind, the rest of this research is based on the presumed unavailability of essential hypotheses and uses this fact to cope with input data conflicts.

4.4 Logicogrammatical Analysis

The purpose of a logicogrammatical analysis is to determine the literal meaning of the utterance in terms of syntactic, semantic and discourse features. These features will be referred to as logicogrammatical features [Patten et al., 1992]. The interpretation of the input takes place in two phases. The first phase creates a composite of hypotheses that are "highly-likely" (have the highest confidence values) since they can best account for a datum. These features in the composite become the islands of certainty and this phase is called the high-likely phase. The second phase is where more hypotheses are added to the composite until all the data is accounted for and a complete path through the logicogrammatical stratum is created. These hypotheses are considered "likely" as they have a lower confidence value than the highly-likely hypotheses. This phase is called the likely phase. Features that are a sibling of a likely or highly-likely feature are considered "unlikely", but are not ruled-out entirely. The logicogrammatical features that create the
path through the logicogrammatical stratum are members of the composite and provide a complete and consistent explanation or interpretation of the input data. While features are types of hypotheses, functions (roles) are the realization of the features and reflect the structural implications of the composite.

4.4.1 Input and Output

The input to the semantic analyzer is dependent upon whether any anticipatory guidance is to take place. In the absence of anticipatory guidance, the input consists entirely of syntactic features and functions that were generated by a syntactic parser. The input is the words in the sentence, how they are combined to create constituents, what syntactic functions identify with each constituent, and what syntactic features are associated with each function.

The input may also contain anticipated features. The number of anticipated features in the input depends on how familiar the "listener" is to the given situation. The example sentences were executed in this system with no anticipatory guidance, which is analogous to a person who is in a completely unknown situation, lacking any expectations, and is forced to do all understanding in a bottom-up manner, from the syntactic parse to the semantics to the pragmatics. Other levels of anticipatory guidance for these experiments is at the level of an average person's knowledge of the situation, a business person's knowledge, or a person with complete knowledge. Each of these levels of anticipation represents a person with an increasingly higher number of pragmatic expectations since each person is increasingly more familiar with the situation. The anticipated features were determined by a linguist at The Ohio State University, Michael Geis, as the plausible expectations for the various familiarity levels.
The output of the semantic interpreter is the composite created through abduction and is intended to represent the literal meaning of the utterance. This composite is comprised of logicogrammatical features that were generated as an explanation of the input sentence’s data. The functions associated with the composite features define the composite’s structural implications and are also part of the output.

**4.4.2 Highly-Likely Phase**

The highly-likely phase is the first phase of operation. The system initially assumes that the syntactic parse represented in the input is the correct parse for the sentence\(^2\). The data from the syntactic parse is explained in terms of highly-likely features and functions from the logicogrammatical stratum.

The highly-likely phase can operate with varying degrees of knowledge. The system can simply take the syntactic information and perform the semantic interpretation without any top-down guidance. Or it can include anticipated features with the syntactic information. Or it can incorporate run-time guidance during processing. Hence, there are really 4 modes of operation: strictly syntactic, syntactic plus anticipated features, syntactic with run-time guidance, and syntactic and anticipated features with run-time guidance.

**4.4.2.1 Strictly Syntactic Processing**

Strictly syntactic processing has no anticipated features in the input nor any run-time top-down guidance. It is simply a bottom-up processing technique. This section discusses its processing.

\(^2\) The problem of an incorrect parse in the input is discussed in Section 5.18.1 on page 159.
After reading in the output from the syntactic parser, each feature hypothesized by the input has its realization rules fired (see "Realization Rules" on page 34). This means that all the realization rules associated with the feature are instantiated. If the realization rule has as one of its arguments a function that does not already exist, it is created and added to the composite. During the highly-likely phase, all hypotheses are given a confidence value of highly-likely.

The next step is to determine which additional logicogrammatical features should be considered highly-likely since they alone can account for an input datum. For each word in the input, the lexicon is checked to see if it can be accounted for by only one logicogrammatical feature. If only one feature can account for that lexical item, then that feature is hypothesized and is considered to be highly-likely. Though this hypothesis is the only way to account for the word, it is not "essential." It is possible that, if this were a speech or text recognition system, the word in the data could be incorrect, i.e. could be noise or misinterpreted from the source of the data.

The next step is to make all the ancestors of highly-likely features highly-likely. This is possible because if any of the ancestors had been ranked as unlikely, then the highly-likely feature could never have existed. The stratum's knowledge representation is a classification hierarchy which by design enables pruning of entire subtrees. The subtree that had the unlikely feature as its root would have been entirely ignored. The situation of conflicting highly-likely features of different words will not arise. Each word is represented by its own hierarchical feature description, so a conflict within a hierarchy for multiple words is not possible. If a constituent has a highly-likely feature that preselects a feature for a subconstituent and the preselection conflicts with a highly-likely feature, then there is an
inconsistency in the logicogrammatical stratum or an error in the syntactic parser. In other words, this should never occur. The realization rules of the highly-likely features are instantiated.

The final step in the processing is to create the system choices. In the grammar, every node that is a parent of mutually exclusive children is connected to them via a system. This means that every group of siblings are related not only by having the same parent(s), but they are also a part of the same system. All systems whose parent feature is likely or highly-likely must establish one of its children. The processing of SAPI is driven by the need to make a choice for every system. A complete path through the hierarchy, which is what is necessary to terminate processing, is identifiable by a choice having been made for all traversed systems. Once all systems have been refined, the system is guaranteed to have a complete explanation.

4.4.2.2 Anticipatory Top-Down Guidance

Anticipatory guidance is realized by adding to the input those logicogrammatical features that a person expects but were not suggested by the syntactic parse. For this research, beyond the level of no anticipatory guidance, there are three levels of familiarity intended to correspond to that of an average person, a business person, and a domain expert. Each of these levels of expertise represent different amounts of expectations or anticipatory guidance. In the domain of making airline reservations, an average person is one who has made a reservation before, but not often. A business person is one who flies frequently and makes reservations a few times a month. A domain expert would be a travel agent. A travel agent making a reservation through another travel agent would have strong expectations of what will be said.
The processing of the highly-likely phase proceeds as it did without the anticipatory
guidance, the anticipated features are merely treated as additional input. It is possible,
however, that the anticipated logicogrammatical features and the syntactically parsed
features may not agree. This occurs in real life when the listener is expecting to hear one
utterance but another is said. This is remedied by having the system ask, at the beginning
of processing, whether more confidence should be placed on the syntactic features or the
anticipated features. If a conflict occurs, the features that are responsible for the conflict that
were produced by the less reliable source have their status changed to unlikely and the
effects of their realization rules are undone\(^3\).

4.4.2.3 Run-time Top-Down Guidance

The inclusion of run-time top-down guidance changes the operation of the system slightly.
The highly-likely phase proceeds as it did before with one difference. Once all the system
choices are made for the highly-likely features, which previously indicated the end of the
highly-likely phase, it is then that run-time guidance begins.

Run-time guidance performs a breadth-first traversal of the developing semantic
parse tree (which can be viewed as a hierarchical organization of the composite), looking
for highly-likely logicogrammatical features that can only be accounted for by one
pragmatic feature. Having only one pragmatic feature that can account for a
logicogrammatical feature is a necessary condition for the pragmatic feature to be highly-
likely, but it is not sufficient. Sufficiency is attained by making sure that:

3. The handling of conflicting features has not yet been fully implemented. The additional modifications to
SAPI to handle conflicting input is explained in Section 5.18.1 on page 159.
1. None of the other realization rules of the pragmatic feature being evaluated will cause a conflict with the other highly-likely logicogrammatical features.

2. None of the ancestors of the pragmatic feature being evaluated have a realization rule that will cause a conflict with the other highly-likely logicogrammatical features.

If both of the sufficiency conditions are met, then the pragmatic feature and all of its ancestors are made highly-likely, and their realization rules are instantiated. If a conflict is found, then the pragmatic feature that has the conflicting realization rule is given the confidence value of unlikely. This eliminates any possibility of it or any of its children being part of the explanation.

If run-time guidance added more highly-likely logicogrammatical features, the next step is to make the ancestors of all the new highly-likely features highly-likely, instantiate their realization rules, and create all the appropriate system choices. Once this is completed, or if no new logicogrammatical features were added, the breadth-first traversal of the semantic parse tree continues, in search of logicogrammatical features which are accounted for by only one pragmatic feature that can pass the necessary and sufficiency tests. Whenever run-time guidance creates new highly-likely logicogrammatical features, the control of the processing is returned to the semantic interpreter to propagate the effects through the logicogrammatical stratum. Processing continues bouncing between the pragmatic stratum and the logicogrammatical stratum until the entire semantic parse tree has been traversed.
4.4.3 Likely Phase

Having determined all the highly-likely features, the core of the composite explanation has been created. Processing, however, is not complete until the logigrammatical stratum has a choice made for every system. A system that does not have any of its children established represents an explanation that is incomplete.

An outline for the algorithm used in the likely phase follows. For a given hierarchy in the logigrammatical stratum, start at its root and begin adding logigrammatical features to the composite. These features are assigned a confidence value of likely. Instantiate any realization rules associated with the likely features. Determine if any gate feature results in being likely and instantiate its realization rules. For all the new features that have children, the system between parent and child must be traversed. Until a child is chosen, an empty system choice, representing that no choice has been made for the system, is part of the hierarchy. Continue traversing this hierarchy until no empty system choices exist or until a conflict in the composite arises. If a conflict occurs, then backtrack, removing features from the composite and adding others. When a choice has been determined for all the systems of this hierarchy, try to complete another hierarchy in the logigrammatical stratum. Processing continues until there are no empty system choices left in the entire logigrammatical stratum and the resultant composite is consistent and complete.

The CPU time that is required to compute a complete and consistent composite is dependent on the order in which features are added to the composite and the order in which the realization rules are instantiated. An efficient system needs to minimize these times. Section 4.4.3.1 discusses the order in which features are added to the composite and section
4.4.3.2 discusses the intelligent ordering of the firing of realization rules. The following sections expand on the various stages of processing in the likely phase.

**4.4.3.1 Traversing a Hierarchy**

Having determined all the features possible in a primarily bottom-up fashion during the highly-likely phase, with or without top-down guidance, the next step is to add features to the composite by traversing each hierarchy in a top-down manner from the root to the leaves. The order that the hierarchies are completed is from the highest "ranking" hierarchy to the lowest ones. The rank of a hierarchy is determined by the size of the constituent it is trying to explain. Any system that does not have a choice made for it yet can be chosen to be fulfilled. The system choice is made by selecting one of the logicogrammatical features associated with it and adding that feature to the composite. The child (feature) that is selected first is the one with the highest degree of plausibility. In many of the sibling sets, one sibling is more likely to occur than the other(s) and is referred to as a default feature. Default features are added to the composite before non-default features. This determination of which features should be default was decided prior to any processing and based on the following:

1. Non-occurrence of a feature suggests the occurrence of a sibling feature. If one of the siblings suggests a very specific sentence structure that should have been identifiable by the syntactic parser, and that structure is not part of the input, then it probably did not occur. For example, if a sentence is effective, it is more likely to be operative than receptive. The feature of receptive requires the predicator to have the feature !en. This feature is necessary for passive and perfec-
tive verbs such as "eaten" or "seen." If the verb is not of this type, then the sentence can not be receptive so it must be operative.

2. Occurrence of a feature will lead to a greater range of possible answers. For example, the system nominal is parent to the features pronoun-head, noun-head, substitute-head and generic-nominal. Noun-head is the feature that is considered most likely of the sibling set. The reason it is more likely than substitute-head and generic-nominal is because these two features lead to the lexification of only five different words, while noun-head is more general and can lead to the lexification of many nouns. Noun-head is preferred over pronoun-head because if a pronoun had existed in the sentence, the syntactic parser would have recognized it and put "pronoun-head" in the input. Since pronoun-head was not in the input, it can be assumed that there is not a pronoun in this phrase, and the best first choice would be noun-head.

For many siblings, there is no clear choice as to which is more likely. This is an example of a hard decision and the abductive strategy will delay making hard decisions until all easier choices have been made. To resolve a hard decision, a feature is randomly chosen from the sibling set and added to the composite. This reduces the confidence in the composite, and is likely to cause backtracking in the hierarchies.

As new features are added to the composite, their realization rules are instantiated.

The order that hypotheses are included in the composite is crucial to the efficiency of the system. The more likely the hypothesis, the earlier it is added to the composite in order to reduce the chance of inconsistencies. An inconsistent composite results in a time-consuming situation since it entails elimination of gates (discussed on page 83), undoing
realization rules (discussed on page 83), backtracking and choosing new features (discussed on page 87), re-evaluation of all gate features, and instantiating all the new features' realization rules. Therefore, it behooves the semantic interpreter to add logicogrammatical features of greatest plausibility first to avoid great computational expense.

4.4.3.2 Firing Realization Rules

In the section on “Realization Rules” on page 34, the five types of realization rules were identified and their meaning discussed. This section discusses how the realization rules materialize in the semantic interpreter and the extent to which the order the realization rule firing is important.

Before any realization rule can be instantiated, all the functions referenced in the rule must exist. If any function does not exist, the system creates it.

The ordering of the rules was determined by how easily they could be undone if a conflict (an inconsistency in the composite) was discovered. The easier to undo the rule, the earlier it should be performed. After each of the easy rules, a consistency check occurs to verify that there is no conflict. If a conflict does occur, it is easy to remove the effects of the fired rule since the more difficult rules have not yet been fired. Hence, the system is postponing hard actions in favor of those that require less processing. By the time processing reaches the more difficult rules, a number of the possible conflicts that could have occurred have not, thereby substantially reducing the range of possible conflicts.

The easiest rules to “unfire” are the rules that lexify, expand and specify adjacencies. The order of firing amongst these three rules is immaterial, however they
ought to be performed prior to preselections and conflations. These rules are considered “easy” because they alter existing data structures rather than adding features or creating structures. This makes them easy to undo. If an error is detected after any of these realization rules, further processing of outstanding rules is suspended, the outstanding rules eliminated, and immediate measures are taken to begin the correction of the composite. In this way, time has not been wasted performing the computationally expensive conflations or preselections.

The rules of preselection and conflation are more involved because they have a more global effect on the semantic parse tree. Preselections are performed before conflations because a preselection causes fewer changes to the semantic parse tree structure, though the number of features and functions added to the structure may be the same. The fewer the structural changes, the easier the rule is to undo. Another reason to perform preselections before conflations is that a preselection is more likely than a conflation to immediately cause an inconsistency. By performing preselections first, the preselection phase can be checked for errors before any conflations occur. If an error occurs at the preselection level, then the processing of conflations can be completely avoided. Furthermore, there are checks for preselection errors before the preselection starts altering the semantic parse tree (composite). These checks avoid creating obvious preselection errors and avoid wasting time performing the preselection.

The effect of a preselection is that it always adds logicogrammatical features to the composite. A function is created if the preselection is made by a function that does not yet exist. The preselected feature will add more logicogrammatical features to the composite since a likely feature must have all likely parents, who are also added to the composite. All
of these features must instantiate their realization rules. If a conflict occurs, such as the preselection of a feature that has already been deemed as unlikely, the system must backtrack amongst the affected hierarchies and un-instantiate all the realization rules of the offending features.

Conflations are performed last. They alter the structure of the semantic parse tree by conjoining two or more functions. When this occurs, all the information of the conflated functions must be combined. A non-destructive means for performing conflations was developed in order to better facilitate unconflations at the cost of having multiple copies of the same functions, their features, and their children's functions and features.

Some errors are only detectable after the conflations occur. Once all the realization rules of a feature are performed, a thorough error-checker is invoked to determine if the structural implications of the composite have remained consistent. If they have not, the composite must be fixed.

4.4.3.3 Conflicting Composite

The composite is checked for inconsistencies after each feature is added. Possible conflicts include, but are not limited to, lexification to a word not in the data, an existing system choice and a preselection choosing mutually exclusive siblings, conflation of two functions that represent conflicting information, etc. When any error occurs, the system must take steps to undo the effects of the alleged offending features before adding any new features to the composite. This process includes a combination of the subtasks of gate feature elimination, realization rule reversal, and backtracking to find new viable features.
Eliminate Gates

The first in the algorithm for handling inconsistencies is to eliminate the gate features from the composite. The most common strategy of gate elimination is to remove all the gate features from the composite and the effect of all of their realization rules. It may seem computationally more efficient to selectively remove gate features associated with system features that are being systematically removed from the composite and adding in other gate features as new system features become likely. However, this is not a prudent strategy. The problem is that the determination of which features are members of the composite is extremely fickle when conflicts arise; the composite becomes volatile. The addition and subtraction of gate features to the composite during the correction would only add to the confusion. It is simpler to remove all the gates, redetermine a set of likely features to join with the highly-likely features in the composite, and then re-evaluate all the gate features to determine which ones can be added to the composite. The strategy is based on the idea of reducing traffic in the composite assembly by first adding system features then gate features. When a conflict occurs, remove all the gate features, wait till the addition and subtraction of system features to the composite acquiesces, re-evaluate the gate features and add those that are likely to the composite.

Undo Realization Rules

Each realization rule of each feature removed from the composite must be undone. The only rule to the ordering of undoing rules is that conflations are undone last. This is so that the composite is as “clean” as possible before performing an involved unconfl ate. Before discussing how unconflates occur, the undoing of the other realization rules will be discussed.
To undo an adjacency realization rule, all information of the location of the functions in relation to other functions and the bounds of the sentence is removed. The reason that all functions are affected rather than just the functions involved in the conflict is that the ramification of nullifying one function’s location may indirectly create the need to nullify another function’s location. This function identification is difficult. It is easier and quicker to nullify the location information of all the functions. The adjacencies will be recreated eventually because the adjacency rules of the features in the composite are always active. As soon as the task of performing the outstanding adjacencies is invoked for the hierarchy, all the location information for the features will be adjusted.

The undoing of the results of an expand rule occurs at the same time as the removal of effects of adjacency rules. It is performed by removing all super-function relations from the functions.

Lexification can be undone by removing the lexical information from the function involved in the lexification rule. Though lexification only occurs at the gate level, the results of a lexification are propagated from the lexifying feature to the ancestors of the feature that explain the same constituent. The lexification will reoccur if and only if the gate that previously fired is still valid to fire after the composite finishes changing.

To undo a preselect, the rules of the preselected feature must be undone. Then, all the parents of the preselected feature that were added to the composite solely because of the preselected feature (they were inherited) must be removed from the composite and their realization rules undone.
The steps required to undo the effects of a conflation are dependent on how many functions are conflated together. The base case is where two functions are conflated.

Function A has child function a and function B has a child function b. A and B are conflated, creating C (which are really functions A and B) and the children of A and B are copied up to C. If a and b had represented the same function, then only one copy would have been moved up, but the union of the information that each provides would be present in the function that has C as a parent. To remove the conflation of A and B, function C is removed along with all of its and its children's (a and b) information.

Figure 17 is an example of three conflated functions. In this case, imagine that the unconflation of B and C is necessary. The top function F (which is really the conflated functions A, B, and C) is removed along with all of its children and associated information. B and C are still conflated, but now their unconflation has been reduced to the base case of conflation. To complete the unconflation, function E is removed with all of its associated information. Notice that on these two cases it has not been necessary to eliminate gates and undo the realization rules of the features associated with the top function. The reason is that
any realization rule actions that had occurred at the top function would have only affected the top function and its children. Since the top function and its children and their information are eliminated, the formal steps to undo the realization rules are not needed.

Figure 17: Three function conflation.

The final case is when there is more than three functions conflated as in Figure 18. In this case, it is not possible to remove the top, remove the base conflation and rebuild the top with the three remaining conflated functions. To explain the reason for this, imagine that A, B, and C are conflated first. This conflation results in H which is used in further processing. Then C conflates with D creating G and G joins the conflation at H. Some inconsistency is noticed and the unconflation of C and D must take place. If H is removed, then all the additional information that was determined since its creation, prior to the conflation of C and D, is lost. A better plan would be not to remove H, but remove all the information at H that is a result of the conflation of C and D. This selective removal is made possible by assigning to each feature as it is added to the composite an identification number representing the number of conflations that have occurred in its hierarchy. When that conflation is to be undone, all features of the composite with that identification number
are removed. This strategy could have been used on the previous two cases, but it is actually more efficient for those cases to just remove the top, as long as the top does not have to be rebuilt.

Figure 18: Four or more function conflation.

**Backtrack and Find New Feature**

Once the composite has had all the gate features removed and the realization rules undone, it is time to determine what new feature to add to the composite that will not cause an inconsistency.

In certain situations, the feature that caused the inconsistency is known and the backtracking can be directed at that feature. For example, if a feature was removed from the composite that had been preselected, then the feature that had performed the preselection should be removed from the composite as well. Or if the feature that caused the inconsistency was a gate feature, then at least one of the parents of the gate feature must be removed. In this case, the parent that is chosen for removal is the one with the lowest
confidence value. If both features have equal values, then the one added last to the composite is removed.

In other situations, the feature that caused the problem is unknown. The system picks the system feature that was added to the composite last and removes it from the composite. The reason that this is a valid form of backtracking is that the composite was consistent before the additional feature and the inconsistency only occurred after the feature was added. It is possible that the last added feature is actually correct and the composite has a different incorrect feature that was added earlier. The system will eventually correct the problem by repetitive backtracking until the faulty feature is removed. In the worst case, this could result in an exhaustive search. The ordering of the changes to the composite by confidence value should keep the search manageable though.

During the removal and addition of features to the composite, a situation can occur where all the siblings of a system have been added to the composite and each one has created a conflict. In this case, the semantic interpreter must consider ruling out the parent leading into that system, removing the parent from the composite, and adding a sibling of the parent to the composite. In this way, the systemic grammar is traversed.

4.5 Pragmatic Analysis

The pragmatic analysis determines the intended meaning or significance of the sentence. It determines the pragmatic features by utilizing a slightly different abductive algorithm than the logicogrammatical stratum, using the output of the semantic interpreter as input.
4.5.1 Input and Output

The input to the pragmatic interpreter is the output of the semantic interpreter. The output of the pragmatic interpreter is a list of pragmatic features that best accounts for the input, where best is defined as maximal explanatory coverage while maintaining a high degree of confidence. Though this strategy has the same definition of "best" as EFLI, its algorithm is a little different. The algorithm will be explained in the next section.

4.5.2 Threshold-Controlled Abduction

The translation from the semantic interpreter's output to pragmatic interpreter's output utilizes a more practical form of abduction than that used in the semantic interpretation. What makes this form of abduction different from the form used in the semantic interpreter is that the system only accounts for data that can be explained with the highest degree of confidence; it is threshold-controlled abduction. The reason that it differs from the abduction strategy of the semantic interpreter is that the output of the semantic interpreter includes features and functions that the pragmatic stratum does not need to explain. They are not pertinent to the pragmatic explanation; the significance of the sentence can be determined without explaining every piece of the semantic interpretation. The pragmatic interpreter produces a best explanation that accounts for as many parts of the input as possible. If an input datum can not be accounted for with a high degree of confidence, then it is left unaccounted for. This approach seems to best fit those situations where much of the incoming data may be noise or irrelevant facts and the lack of an explanation for the data is nonconsequential.
The strategy of threshold-controlled abduction has multiple steps. The goal of the strategy is to create a composite of highly-likely pragmatic features that explain the logicogrammatical features in the input. The steps are as follows:

1. All of the pragmatic features that can explain any logicogrammatical feature are placed into the Accounts-For group, along with a pointer to the logicogrammatical features that each pragmatic feature can explain. Some logicogrammatical features may be accounted for by multiple pragmatic features. For example, imagine that the semantic interpretation consists of data \( d_{1-5} \) and the hypothesized pragmatic features \( (h_{1-6}) \) are placed in the Accounts-For group, with each hypothesis "pointing" at the data it can explain. This is depicted in Figure 19. For this example, assume the relationships among the hypotheses are: \( h_4 \) and \( h_6 \) are mutually exclusive siblings, \( h_3 \) is a child of \( h_4 \), and \( h_1, h_2 \) and \( h_5 \) are unrelated to any other \( h_i \), where \( 1 \leq i \leq 6 \).

![Figure 19: Example of pragmatic interpretation after step one.](image)
2. The Accounts-For group of pragmatic features is then analyzed. If any pragmatic feature ($h_i$) of the Accounts-For group has a pointer to a logicogrammatical feature that does not exist in the input, then that pragmatic feature is removed from the group. From the example, $h_1$ is removed from the Accounts-For group as it accounts for "x" which was not in the input data. This step is based on the assumption that a fixed set of logicogrammatical features are pre-selected by each pragmatic feature.

3. Any logicogrammatical feature ($d_j$) that is pointed at by only one pragmatic feature is explainable by only one pragmatic feature. That pragmatic feature is highly-likely and is moved from the Accounts-For group to the composite. The data it accounts for are marked as not needing further explanation. In the example, the hypothesis $h_3$ is moved to the composite and the data $d_2$ and $d_3$ are accounted for (marked with an X). The result of these last two steps on the example is in Figure 20.

![Diagram](image-url)

Figure 20: Example of pragmatic interpretation after step three.
4. Add to the composite all the ancestors of the highly-likely pragmatic features since they must also be highly-likely. Mark as accounted for all the logico-grammatical features that are explained by the composite. This results in \( h_4 \) being added to the composite, since it is the parent to \( h_3 \), and \( d_4 \) being accounted for. The status of the example after this step is in Figure 21.

![Diagram of Accounts-For Group and Composite](image)

Figure 21: Example of pragmatic interpretation after step four.

5. Any pragmatic feature that is a sibling of a highly-likely pragmatic feature can be ranked as unlikely since siblings are mutually exclusive. If any of the unlikely pragmatic features are in the Accounts-For group, remove them from the group. In the example, \( h_6 \) is a sibling of \( h_4 \) which is in the composite, so \( h_6 \) is unlikely and removed from the Accounts-For group. The result of this step is depicted in Figure 22.
6. Remove from the Accounts-For group any hypothesis that can only account for data that has been already accounted for. This will result in $h_2$ being removed from the group.

7. Since pragmatic features have been removed from the Accounts-For group, steps 3-6 are repeated to see if any of the remaining pragmatic features are now highly-likely. Processing terminates when one complete pass through the algorithm results in no additional hypotheses being added to the composite. In the example, $h_5$ is now highly-likely since it is the sole way to account for $d_5$. The ending state of the example is depicted in Figure 23.

Figure 22: Example of pragmatic interpretation after step five.
In the example, the composite explanation is the set \{h_3, h_4, h_5\} that can explain \{d_2, d_3, d_4, d_5\}. Processing ceased with an empty Accounts-For group, but this is not a necessary condition. If there had existed in the original example a \(d_6\) such that \(h_7\) and \(h_8\) both explained it, the algorithm would have halted at this same point with a non-empty Accounts-For group. Also, notice that \(d_1\) was never accounted for, which is also permissible since this is threshold-controlled abduction where accounting for all the data is not necessary.

4.6 Implementation Level

The semantic and pragmatic interpretation abductive strategies have been discussed at a high-level. This section shows actual input and output from SAPI and identifies the major data structures in terms of the language of implementation which is OPS5.
SAPI begins with the information that identifies a syntactic parse tree. For example, Figure 24 shows a portion of the syntactic parse tree for the sentence "What time can you leave?" The syntactic functions are represented in upper case and the syntactic features are in lower case. The function of U is not a real function; it is only used to represent the root of the syntactic parse tree.

Figure 24: Portion of a sample syntactic parse tree.

To represent this structure to the semantic interpreter, several working memory elements (wme) are created. First, each syntactic function in the parse tree receives a working memory element called a HUB. A hub is a data structure that identifies a function

---

4. Working memory elements can be thought of as data structures or records, but in OPS5 they are called working memory elements.
and is so named since a function has features associated with it; it is the “hub” of activity.

An example of a HUB is:

\[
\text{HUB } \text{^OF RANGE } ^\text{IS G10 } ^\text{MOM G9 } ^\text{FROM 0 } ^\text{TO 2 } ^\text{STATUS H } ^\text{TYPE BASIC}
\]

This HUB corresponds to the function of RANGE in Figure 24. Each word preceded by a “^” is referred to as an attribute, and can be thought of as a field in a record. The information presented in this HUB is the role of the hub (RANGE), its identification number in the system (G10), its mother’s identification number (G9), its starting position in the string (0), its ending position (2), its status of highly-likely (H), and its type (BASIC). A HUB can also have the attributes of LEX, LEXIFIER, SUPER, NEEDED, and CNUM. If an attribute is not listed in the working memory element, then its value is nil. The meaning of each of the attributes is as follows:

- **OF** specifies the syntactic role of the hub.
- **IS** assigns the hub a unique identification number.
- **MOM** specifies the parent of the hub and is filled by its parent’s IS field.
- **FROM** represents the starting position of the hub’s constituent.
- **TO** represents the ending position of the constituent.
- **STATUS** refers to the plausibility that the hub should exist. STATUS can be H, for highly-likely, L for likely, or U for unlikely. All input hubs have a STATUS of H. Hubs created during processing assume the status of the feature that caused their creation.
- **TYPE** refers to the conflation status of the hub. The type of BASIC implies the hub has not been conflated with any others, CONFLATED implies that the hub is conflated with one other, and TOP implies that it is conflated with 2 or more other hubs.
- **LEX** is the word in the data that the HUB lexifies to.
- **LEXIFIER** is the feature that linked the lexification to the HUB.
• **SUPER** contains the identification number of the HUB that this hub is part of the expansion of.

• **NEEDED** determines if the HUB should still be considered for operations or if it should be ignored because a conflated version of it exists.

• **CNUM** is the value of the counter of the number of conflations that have occurred. Each HUB, when it is created, is given the current conflation number.

Each syntactic feature in the parse tree is represented by a wme called a TABLE. Since a feature is a hypothesis, a TABLE is explanatory hypothesis. An example of a TABLE is:

```
TABLE FEATURE DETERMINED ^VALUE H ^ID G10 ^STATUS INPUT
```

This TABLE wme corresponds to the feature "determined" in Figure 24. The information presented in this TABLE wme is the name of the feature it represents (DETERMINED), its value of highly-likely (H), its specification as to which hub it modifies (G10), and its reason for existing (it is from the INPUT). Other TABLE attributes include NUM and CNUM. The meaning of each of the attributes is as follows:

• **FEATURE** specifies which syntactic feature the TABLE represents.

• **VALUE** specifies the plausibility of this feature. This attribute actually tells whether the feature is highly-likely (H), likely (L), or unlikely (U). Input features are always highly-likely.

• **ID** specifies to which hierarchy (hub) the feature is associated.

• **STATUS** specifies why this feature exists. If it comes from the input, its status is INPUT. Other possible STATUSes are PRESELECTED, DEFAULT, and INHERITED.

• **NUM** is the value of the counter that keeps track of the number of table features added to the composite.

• **CNUM** specifies the identification number of the last conflation that occurred.
The third type of wme created is the DATA element. One DATA wme is created for each word in the input. An example of a DATA wme is:

DATA ^VALUE lwhatl ^ID G11 ^FROM 0 ^TO 1

This DATA wme specifies which word it is associated with (what), the HUB it is related to (G11) and its starting and ending positions in the string (0 and 1, respectively). The meaning of each of the attributes is as follows:

- VALUE which is the word that the DATA element represents.
- ID which specifies which net it belongs to. This number will agree with the HUB's IS attribute and the TABLE's ID attribute.
- FROM tells the starting position in the sentence of that word.
- TO tells the ending position in the sentence of that word.

The actual input to the semantic interpreter that corresponds with the information provided in Figure 24 on page 95 is shown in Figure 25.
If anticipated features are to be part of the input, they are included as TABLE wmes.

The output of the semantic interpreter is the input to the pragmatic interpreter. The next two figures represent sample output for the sentence “What time can you leave?” Figure 26 represents the list of logicogrammatical features and Figure 27 represents the logicogrammatical functions.
Figure 26: Logicogrammatical features as output from the semantic interpreter.
The numbers following each feature specifies the path of functions from the root (U) to the function with which the feature identifies. For example, the last feature of 
PLURAL is followed by nil, G8, and $$OPS_2562. To understand this, the list of functions 
(Figure 27) must be reviewed. $$OPS_2562 refers to the roles SUBJECT and AGENT, so 
the Subject and Agent of the sentence are plural. The parent of Subject and Agent is G8, 
which is the identification number for the function U which is the root to the tree. The “nil” 
is used by the pragmatic interpreter to identify the start of the functions’ identification 
numbers. This form of output enables the identification of which features belong with 
which function(s).

(translator U G8 )
(translator SUBJECT $$OPS_2562 )
(translator AGENT $$OPS_2562 )
(translator HEAD $$OPS_2563 )
(translator FINITE $$OPS_2568 )
(translator MODAL $$OPS_2568 )
(translator TOPICAL $$OPS_2577 )
(translator WH $$OPS_2577 )
(translator RANGE $$OPS_2578 )
(translator DEICTIC $$OPS_2579 )
(translator HEAD $$OPS_2581 )
(translator PROCESS $$OPS_2588 )
(translator PREDICATOR $$OPS_2588 )
(translator MODALSTEM $$OPS_2588 )

Figure 27: Logicogrammatical functions as output from the semantic interpreter.

The previous two figures represent a semantic parse tree, or the composite 
explanation. The portion of the output that corresponds to the portion of the input presented 
in Figure 25 on page 99 results in the semantic parse tree shown in Figure 28.
The output of the semantic interpreter in the form of logigrammatical features and functions represented in Figures 26 and 27 explain the sentence "What time can you leave?" When this output is the input to the pragmatic interpreter, the pragmatic interpreter builds an explanatory composite comprised of a set of pragmatic features. The pragmatic features for this sentence are shown in Figure 29.

( INFORMAL )
( DEPARTURE-DATE-KNOWN )
( CHANCES-NOT-GOOD )
( CLIENT-PLURAL )

Figure 29: Output from pragmatic interpreter
The pragmatic feature of INFORMAL identifies the formality of the sentence. For example, the sentence “What time can you leave?” is informal while the sentence “What time can you depart?” is formal. The feature DEPARTURE-DATE-KNOWN is a part of the pragmatic stratum that determines when the person is able to fly in terms of the time and date of their departure and return. Since the pragmatic interpreter is trying to determine flying times and the departure date is known, the time of the departure must be in question. CHANCES-NOT-GOOD is part of the output because the sentence had the word “can” in it as a modal. The possibility that the listener can get a flight at a certain time is not very good. The final feature in the list is CLIENT-PLURAL. This was chosen since the word “you” refers to the client and can be either singular or plural. The semantic interpreter lacked a conflict when it added the feature “plural” to the composite as a feature describing the subject of the sentence. Therefore, the plural feature was passed along to the pragmatic interpreter. That feature may be correct if the travel agent is actually making reservations for a group and is referring to the representative of the group as “you,” implying “all of you.” It may be that there is only one client in which case the feature could be CLIENT-SINGULAR. Both CLIENT-SINGULAR and CLIENT-PLURAL are children of CLIENT-ORIENTED which indicates that the listener is the subject of the sentence instead of the travel agent. The pragmatic interpreter determining that the sentence is CLIENT-ORIENTED is more important than figuring out the plurality since the designation of whether the client is singular or plural should be handled by a planning component in the conversation. Once the number of people interested in making a reservation (determinable by the number of seats being reserved) has been determined, then the appropriate feature would be selected. At this time, no planning component is connected to the semantic and pragmatic interpreters.
4.7 Conclusion

This chapter has discussed the abduction strategies used in the processing of the semantic and pragmatic interpreters as well as an implementation level description of the form and content of the input and output. The system operates with varying amounts of input where the amount is dependent on the level of anticipatory guidance. It is an example of layered classificatory abduction since it has multiple layers performing abduction where the layers are classification hierarchies and each abductive layer can “influence” the processing of its neighbors. The system also utilizes run-time guidance while processing. Several implementation issues have been addressed and the rationalization for the final decisions have been provided.

The next chapter presents the CPU processing time used by the system to comprehend eleven sentences under varying amounts of anticipatory guidance, with and without run-time-guidance.
CHAPTER V

Experiment Results of SAPI

5.1 Introduction

The purpose of this chapter is to present the results of the experiments. The chapter opens with a brief discussion of the input, output, and modes of processing, a definition of what constitutes the "processing time" in the results, and a discussion of the composition of the input of each sentence. The expected results of this research are briefly explained. Next, the amount of time required to "understand" all the sentences, with and without run-time guidance at each of the various levels of anticipation is presented. An analysis of the various results will be provided. The chapter concludes with a discussion of the various possible forms of incorrect input, the changes needed in SAPI to handle incorrect input, and the current ability of SAPI to cope with incomplete input.

5.2 Inputs, Outputs, and Processing

The input to the semantic interpreter depended on the level of anticipation. In the case of no anticipation, the input was completely determined by the syntactic parse. Additional syntactic information was provided for the anticipatory levels of average, business and complete. This additional information was determined by considering the pragmatic expectations a person of the level of familiarity should possess. Not all sentences have all levels of anticipation.
The order in which the sentences are discussed in this chapter was determined by the number of anticipatory levels available for that sentence. The first three sentences have input data for all four anticipatory levels, the next two have three levels, the following three have two levels, and the last three lack any anticipation. The basis for the existence of a level was the determination of whether or not a person of a certain familiarity level would have greater expectations than a person at the next less-informed level. For example, if a sentence has data for the situations of no anticipation, average and complete, and no data for the level of a business person, then this implies that a business person would have no more anticipations than an average person. Hence, in the cases of the groups of sentences having 2 or 3 anticipatory levels, the members of the group may not have had the exact same levels. They have been grouped together since the absence of a level implies that the less-informed level’s data applies to the higher (missing) level. For example, assume sentence one has none, business and complete anticipatory levels and sentence two has none, average and complete anticipatory levels. These two sentences can be grouped together since sentence two actually does have a business level, it is the same as its average level. Therefore, both sentences have the levels of none, business and complete. Where and when this occurs in the sentence groupings will be made explicit in this chapter.

The output of the semantic interpreter is the logicogrammatical functions and features that define the literal meaning of the utterance.

The processing is either strict layered abduction where the semantic interpreter completes its task before the pragmatic interpreter starts its task, or it is interleaved layered abduction where the pragmatic interpreter hypothesizes logicogrammatical features during
run-time. The first case will be referred to as "no run-time guidance," while the latter is "run-time guidance."

The processing times presented in the chapter are the actual number of seconds that the CPU spent processing the code to complete the match cycle (determining which rule should fire) and right-hand side actions. The processing times are broken down into two phases; the time to determine the highly-likely features (the highly-likely phase) and the time to determine the likely features (the likely phase). The sum of these two times is the time to complete the analysis of the sentence at the semantic level. The time to create a full pragmatic interpretation has been ignored since it is small (four to seven seconds) in comparison to the semantic interpretation and fairly consistent for all the sentences. The processing times reported are actually an average of the processing times of six different runs of the same sentence under the same conditions. In all cases, the standard deviation was less than ±0.6 seconds. All the runs were performed on a Sun 4/65 (SPARCstation 1+) with 16 MB main memory, 101 MB virtual memory, a 15.5 mips Sparc CPU, Sun4 application architecture, Sun4c kernel architecture, and a SunOS v4.1.3 operating system. The machine was connected to its server via a Thicknet ethernet cable through a Cabletron MR800 Multiport Transceiver. These experiments were run remotely during non-peak network time. The language used was CparaOPS5, version 5.3 P4.4. The number of production rules in the system was 555 and the number of two input nodes was 1120.

For each of the eleven test sentences, several pieces of information are provided. First is a table that gives the processing times for each of the levels of anticipation, with and without run-time guidance. This table gives the processing times of the highly-likely phase, the likely phase, and their sum which is the time necessary to complete processing. Next
are two graphs that depict this information without and with run-time guidance. Next is a table that shows the same processing times in a different format so that the speedup attained by incorporating run-time guidance is easily determined. This information is then presented graphically. After each group of sentences having the same anticipatory levels, a table and a chart are provided to better compare the processing times and the speedup caused by anticipatory guidance.

5.3 Expected Results

The expected results of the system are two-fold. The first has to do with anticipatory guidance. This expected result is based on the concept that the more knowledge available about a situation, the more expectations or anticipations exist of what will happen and what will be said. This should be reflected in the processing times consistently showing some speedup from one level of anticipation to the next. The more anticipatory guidance, the faster the literal meaning of the sentence can be determined.

The second expected result is based on the effect of run-time guidance. The idea is that run-time guidance should speedup overall processing because the pragmatic stratum is guiding the processing in the logicogrammatical stratum. Therefore, the time to arrive at a literal interpretation should be less with run-time guidance than without run-time guidance.

5.4 Sentence #1 - “What time can you leave?”

This sentence had four levels of anticipation under which to be tested; none, average, business and complete.
The case of no anticipation gives the basic syntactic parse. A graphical depiction of this syntactic parse is provided in Figure 30. The average level adds the features non-present-in and non-past-in, which are both considered default features. The business level adds some more default features, and also identifies the indicative interactant subject as “you,” “time” as a mass noun, the Finite “can” as a Modal, and the Predicator “leave” as the Process. The complete level further identifies that “you” is singular.

SAPI’s semantic interpretation of the input is shown in Figure 31. The reason some features appear in the input but not in the semantic interpretation is that they are either preselected features, in which case they are not included since having the preselector will guarantee having the preselectee, or they are an ancestor to a feature, in which case they would have been found through inheritance.
Figure 30: Syntactic input for sentence 1.
Figure 31: Semantic interpretation of sentence 1.
5.4.1 Effects of Anticipatory Guidance

Table 1 gives the processing times for sentence 1 to run under the various anticipatory levels.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>None</th>
<th>Average</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>9.67</td>
<td>10.22</td>
<td>21.33</td>
<td>21.89</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>134.16</td>
<td>125.33</td>
<td>68.70</td>
<td>58.33</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>14.97</td>
<td>15.59</td>
<td>25.64</td>
<td>26.21</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>107.32</td>
<td>98.78</td>
<td>69.37</td>
<td>58.86</td>
</tr>
</tbody>
</table>

This sentence displayed the expected results: as the semantic interpreter received more anticipatory information in the input, the highly-likely phase became longer, taking more time to compute what the highly-likely features and functions were and propagating the information. The highly-likely phase computed more islands of certainty, resulting in fewer hypotheses for the likely phase to consider. Hence, the extra time spent determining highly-likely features is made up for by saving time in the likely phase. In fact, so much time is saved in the likely phase that the overall processing time decreases as the anticipatory level increases.

Figures 32 and 33 graphically represent the time to complete the highly-likely phase and the time to compete the likely phase for each of the various anticipatory levels,
without and with run-time guidance, respectively. By using a stack chart, the total processing time is readily visible. The numbers that appear in the chart represent the CPU time in seconds to complete each of the two phases.

![Stack Chart](image)

**Figure 32:** Highly-likely and likely phase times for sentence 1 without run-time guidance.

![Stack Chart](image)

**Figure 33:** Highly-likely and likely phase times for sentence 1 with run-time guidance.
5.4.2 Run-Time Guidance Effects

Table 2 shows the completion times for each phase of processing for each of the anticipatory levels, while Figure 34 on page 116 graphically depicts the speedup caused by run-time guidance. As expected, the highly-likely phase has increased in all cases. This is expected since it is during the highly-likely phase that the additional processing of run-time guidance occurs. The purpose of spending additional time in the highly-likely phase is that hopefully the likely phase’s time will be reduced by more than the highly-likely phase’s time was increased, resulting in a lower overall processing time. This is the case for the experiments of none and average anticipation; though the highly-likely phase was slower, the likely phase was faster and the overall processing was faster with run-time guidance than without. Unfortunately, in the cases of the business and complete anticipatory levels, both the highly-likely and likely phase times were slower. Perhaps the best explanation for the slowdown of the likely phase in these two instances is that the system was left in a somewhat different state after the highly-likely phase, regardless of run-time guidance. The slightly different state can result in a different match time speed. Since the percent slowdown (negative speedup) of the likely phase is so small, it can be discounted to this difference. The slowdown in the overall processing is then attributable to the fact that the highly-likely phase was slower, and this slowness was never reclaimed by a speedup in the likely phase.
Table 2: Run-time guidance effects on sentence 1.

<table>
<thead>
<tr>
<th>Anticipation:</th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highly-likely</td>
<td>9.67</td>
<td>14.97</td>
<td>-54.81%</td>
</tr>
<tr>
<td>None</td>
<td>Likely</td>
<td>134.17</td>
<td>107.32</td>
<td>20.01%</td>
</tr>
<tr>
<td>All Processing</td>
<td></td>
<td>143.84</td>
<td>122.33</td>
<td>14.98%</td>
</tr>
<tr>
<td>Average</td>
<td>Highly-likely</td>
<td>10.22</td>
<td>15.59</td>
<td>-52.54%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>125.33</td>
<td>98.78</td>
<td>21.18%</td>
</tr>
<tr>
<td>All Processing</td>
<td></td>
<td>135.54</td>
<td>118.83</td>
<td>13.98%</td>
</tr>
<tr>
<td>Business</td>
<td>Highly-likely</td>
<td>21.33</td>
<td>25.64</td>
<td>-20.21%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>68.70</td>
<td>69.37</td>
<td>-0.98%</td>
</tr>
<tr>
<td>All Processing</td>
<td></td>
<td>80.00</td>
<td>88.00</td>
<td>-10.00%</td>
</tr>
<tr>
<td>Complete</td>
<td>Highly-likely</td>
<td>21.89</td>
<td>26.21</td>
<td>-19.74%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>58.33</td>
<td>58.86</td>
<td>-0.91%</td>
</tr>
<tr>
<td>All Processing</td>
<td></td>
<td>80.22</td>
<td>85.00</td>
<td>-6.05%</td>
</tr>
</tbody>
</table>
Analysis of each of the run-time guidance experiments supplies an explanation for the slowdown of the business and complete levels in total processing time. The input information for the level of no anticipation determined one of the leaf pragmatic features, which provided the logicogrammatical stratum with some additional logicogrammatical features. The additional input information provided at each of the other anticipatory levels was unable to trigger any additional pragmatic features. This explains why the anticipatory level of none has a speedup; it received additional logicogrammatical information from using the run-time guidance. The average case did not receive any additional run-time guidance from the pragmatic stratum than the previous level. It received the same guidance as the business and complete levels which show a slowdown. Why then does the average level have a speedup? Further analysis shows that the average anticipatory level has a speedup in the likely phase since the features produced by the pragmatic stratum were not part of the input from the syntactic parser. Therefore, run-time guidance predicted additional features in the logicogrammatical stratum which means that these features did
not have to be discovered during the likely phase. The business and complete levels do not have a positive speedup because the highly-likely phase did not provide any additional features not already in their input. As a result, the same processing had to be performed in the likely phase regardless of the existence or absence of run-time guidance.

5.5 Sentence #2 - "I can put you on that 10pm flight."

This sentence had four levels of anticipation under which to be tested; none, average, business and complete.

The case of no anticipation gives the basic syntactic parse. A graphical depiction of this syntactic parse is provided in Figure 35. The average level merely gives default features. The business level specifies the indicative interactant subject. The complete level further identifies that the Adjunct, Predicator and Residual functions in the original input are conflated with the functions Spatial, Process and Medium, respectively. The complete level also gives more default features.

The correct semantic interpretation of the input is shown in Figure 36. The reason some features appear in the input but not in the semantic interpretation is that they are either preselected features, in which case they are not included since having the preselector will guarantee having the preselectee, or they are an ancestor to a feature, in which case they would have been found through inheritance.
Figure 35: Syntactic input for sentence 2.
"I can put you on that 10pm flight."

Figure 36: Semantic interpretation of sentence 2.
5.5.1 Effects of Anticipatory Guidance

Table 3 gives the processing times for sentence 2 to run under the various anticipatory levels.

Table 3: Effects of anticipatory guidance on sentence 2.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>None</th>
<th>Average</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>22.65</td>
<td>25.52</td>
<td>28.24</td>
<td>37.32</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>241.49</td>
<td>205.42</td>
<td>135.37</td>
<td>100.78</td>
</tr>
<tr>
<td></td>
<td><strong>All Processing</strong></td>
<td><strong>264.14</strong></td>
<td><strong>230.94</strong></td>
<td><strong>163.61</strong></td>
<td><strong>147.10</strong></td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>27.11</td>
<td>29.27</td>
<td>33.37</td>
<td>43.00</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>216.08</td>
<td>207.65</td>
<td>136.89</td>
<td>101.78</td>
</tr>
<tr>
<td></td>
<td><strong>All Processing</strong></td>
<td><strong>243.19</strong></td>
<td><strong>227.92</strong></td>
<td><strong>164.16</strong></td>
<td><strong>144.78</strong></td>
</tr>
</tbody>
</table>

This sentence, just like sentence 1, displayed the expected result; as the anticipation level increases, the highly-likely phase time increases, the likely phase time decreases, and the overall processing time decreases. Figures 37 and 38 graphically represent the time to complete the highly-likely and the likely phase for each of the various anticipatory levels, without and with run-time guidance, respectively.
5.5.2 Run-Time Guidance Effects

Table 4 shows the completion times for each phase of processing for each of the anticipatory levels, while Figure 39 on page 124 graphically depicts the speedup caused by the run-time guidance. As expected, the highly-likely phase has increased in all cases. Unfortunately, the extra time spent during the highly-likely phase was not recovered by a
decreased likely phase time in the case of the anticipatory levels of average, business and complete. Once again, this is caused by the additional features provided at those levels not necessitating the existence of any additional pragmatic features beyond those achieved at the level of no anticipation. This means that the run-time guidance was no more beneficial to these levels than it was at the level of no anticipation. The extra time spent in the highly-likely phase was never reclaimed during the likely phase, resulting in a slowdown in the overall processing when run-time guidance is added. Once again the slowdown in the likely phases can be attributed to the system being left in a different state after the highly-likely phase in both the run-time and no run-time guidance cases. Since the percent slowdown (negative speedup) is so small, it can be ignored.
Table 4: Run-time guidance effects on sentence 2.

<table>
<thead>
<tr>
<th>Anticipation</th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>22.65</td>
<td>27.12</td>
<td>-19.74%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>241.49</td>
<td>216.07</td>
<td>10.53%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>264.13</td>
<td>299.24</td>
<td>-11.9%</td>
</tr>
<tr>
<td>Average</td>
<td>Highly-likely</td>
<td>25.52</td>
<td>29.27</td>
<td>-14.69%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>205.42</td>
<td>207.65</td>
<td>-1.09%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>240.28</td>
<td>246.75</td>
<td>-2.7%</td>
</tr>
<tr>
<td>Business</td>
<td>Highly-likely</td>
<td>28.24</td>
<td>33.37</td>
<td>-18.17%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>135.37</td>
<td>136.89</td>
<td>-1.12%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>163.93</td>
<td>170.45</td>
<td>-3.6%</td>
</tr>
<tr>
<td>Complete</td>
<td>Highly-likely</td>
<td>37.32</td>
<td>43.00</td>
<td>-15.22%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>100.78</td>
<td>101.78</td>
<td>-0.99%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>138.10</td>
<td>139.75</td>
<td>-1.2%</td>
</tr>
</tbody>
</table>
Figure 39: Processing speedup caused by run-time guidance for sentence 2.

5.6 Sentence #3 - “How many seats are you looking for?”

This sentence had four levels of anticipation under which to be tested; none, average, business and complete.

The case of no anticipation gives the basic syntactic parse. A graphical depiction of this syntactic parse is provided in Figure 40. The average level gives default features and identifies the sentence as having a mental process type. The business level specifies the indicative interactant subject. The complete expectations further identifies that the sentence has the feature extent, which will conflate the function Extent with the function Adjunct. The complete level also gives more default features.

The correct semantic interpretation of the input is shown in Figure 31. The reason some features appear in the input but not in the semantic interpretation is that they are either preselected features, in which case they are not included since having the preselector will
guarantee having the preselectee, or they are an ancestor to a feature, in which case they would have been found through inheritance.

Figure 40: Syntactic input for sentence 3.
"How many seats are you looking for?"

Figure 41: Semantic interpretation of sentence 3.
5.6.1 Effects of Anticipatory Guidance

Table 5 gives the processing times for sentence 3 to run under the various anticipatory levels.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>None</th>
<th>Average</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>13.62</td>
<td>19.00</td>
<td>20.90</td>
<td>25.63</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>206.33</td>
<td>143.37</td>
<td>106.96</td>
<td>88.01</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>16.26</td>
<td>20.58</td>
<td>22.12</td>
<td>27.82</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>116.45</td>
<td>109.91</td>
<td>106.61</td>
<td>85.50</td>
</tr>
</tbody>
</table>

This sentence, just like sentences 1 and 2, displayed the expected result; as the anticipation level increases, the time for the highly-likely phase increases, the time for the likely phase decreases, and the time for overall processing decreases. Figures 42 and 43 graphically represent this information, without and with run-time guidance, respectively.
5.6.2 Run-Time Guidance Effects

Table 6 shows the completion times for each phase of processing for each of the anticipatory levels, while Figure 44 on page 130 graphically depicts the speedup caused by run-time guidance. As expected, the highly-likely phase increased in time. In this
experiment, the likely phase decreased in time from no run-time guidance to run-time guidance, but the business level still shows a net slowdown. That is because the speedup attained in the likely phase did not completely compensate for the additional time used in the highly-likely phase. The reason that the complete anticipatory level managed a slight speedup is that an additional logicogrammatical feature in the input triggered the necessity of an additional pragmatic feature, which resulted in the existence of additional logicogrammatical features.

Table 6: Run-time guidance effects on sentence 3.

<table>
<thead>
<tr>
<th>Anticipation:</th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Highly-likely</td>
<td>13.62</td>
<td>16.26</td>
<td>-19.38%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>206.33</td>
<td>116.45</td>
<td>43.56%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>28.28</td>
<td>39.27</td>
<td>39.38%</td>
</tr>
<tr>
<td>Average</td>
<td>Highly-likely</td>
<td>19.00</td>
<td>20.58</td>
<td>-8.32%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>143.37</td>
<td>109.91</td>
<td>23.34%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>17.32</td>
<td>16.49</td>
<td>5.58%</td>
</tr>
<tr>
<td>Business</td>
<td>Highly-likely</td>
<td>20.90</td>
<td>22.12</td>
<td>-5.84%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>106.96</td>
<td>106.61</td>
<td>0.33%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>12.38</td>
<td>12.86</td>
<td>-3.83%</td>
</tr>
<tr>
<td>Complete</td>
<td>Highly-likely</td>
<td>25.63</td>
<td>27.82</td>
<td>-8.54%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>88.01</td>
<td>85.50</td>
<td>2.85%</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>11.25</td>
<td>10.37</td>
<td>8.23%</td>
</tr>
</tbody>
</table>
5.7 The Effect of Anticipatory Guidance on Sentences 1-3

From Table 7, it is apparent that in every case, increasing the amount of anticipated information results in quicker processing of the input. This is apparent since every single experiment has a speedup in overall processing from one level of anticipation to the next. Table 7 displays the speedups from one anticipatory level to the next and a column to show the speedup from the level of no anticipation to complete. Figures 45 and 46 graphically show that there was a speedup in processing time as the level of anticipation increased, regardless of the inclusion of run-time guidance.
Table 7: Speedup caused by increasing levels of anticipation for sentences 1-3.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>None Average</th>
<th>Average Business</th>
<th>Complete Business</th>
<th>Complete None</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Run-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>-5.69%</td>
<td>-108.71%</td>
<td>-2.63%</td>
<td>-126.37%</td>
</tr>
<tr>
<td>Likely</td>
<td>6.58%</td>
<td>45.18%</td>
<td>15.09%</td>
<td>56.52%</td>
</tr>
<tr>
<td>Run-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>-4.14%</td>
<td>-64.46%</td>
<td>-2.22%</td>
<td>-75.08%</td>
</tr>
<tr>
<td>Likely</td>
<td>7.96%</td>
<td>29.77%</td>
<td>15.15%</td>
<td>45.15%</td>
</tr>
<tr>
<td>Sentence #1: What time can you leave?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Run-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>-12.67%</td>
<td>-10.66%</td>
<td>-32.15%</td>
<td>-64.77%</td>
</tr>
<tr>
<td>Likely</td>
<td>14.94%</td>
<td>34.10%</td>
<td>25.55%</td>
<td>58.27%</td>
</tr>
<tr>
<td>Run-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>-7.97%</td>
<td>-14.01%</td>
<td>-28.86%</td>
<td>-58.61%</td>
</tr>
<tr>
<td>Likely</td>
<td>3.90%</td>
<td>34.08%</td>
<td>25.65%</td>
<td>52.90%</td>
</tr>
<tr>
<td>Sentence #2: I can put you on that 10pm flight.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Run-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>-39.50%</td>
<td>-10.00%</td>
<td>-22.63%</td>
<td>-88.18%</td>
</tr>
<tr>
<td>Likely</td>
<td>30.51%</td>
<td>25.40%</td>
<td>17.72%</td>
<td>57.35%</td>
</tr>
<tr>
<td>Run-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>-26.57%</td>
<td>-7.48%</td>
<td>-25.77%</td>
<td>-71.09%</td>
</tr>
<tr>
<td>Likely</td>
<td>5.62%</td>
<td>3.00%</td>
<td>19.80%</td>
<td>26.58%</td>
</tr>
<tr>
<td>Sentence #3: How many seats are you looking for?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Figure 45: Processing speedup caused by increasing the level of anticipation for sentences 1-3, without run-time guidance.

Figure 46: Processing speedup caused by increasing the level of anticipation for sentences 1-3, with run-time guidance.
5.8 Sentence #4 - "I’m not finding anything then."

This sentence had three levels of anticipation under which to be tested; none, business and complete.

The case of no anticipation gives the basic syntactic parse. The business level specifies the indicative interactant subject as the speaker. The complete level further identifies that the Predicator and Residual functions from the syntactic input are conflated with the functions Process and Medium, respectively. The complete level also specifies that the final constituent in the sentence has the feature time-adjunct.

Figures of the syntactic input and semantic interpreter’s output for this and the following sentences will not be included for the sake of brevity.

5.8.1 Effects of Anticipatory Guidance

Table 8 gives the processing times for sentence 4 to run under the various anticipatory levels.

Table 8: Effects of anticipatory guidance on sentence 4.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>None</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>12.71</td>
<td>15.62</td>
<td>19.32</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>14.04</td>
<td>16.87</td>
<td>20.69</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>87.07</td>
<td>60.52</td>
<td>58.16</td>
</tr>
<tr>
<td></td>
<td>Highly-likely</td>
<td>98.48</td>
<td>60.87</td>
<td>78.48</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>87.51</td>
<td>60.87</td>
<td>58.69</td>
</tr>
</tbody>
</table>
This sentence displayed the expected result of having an increased highly-likely phase time and a decreased likely phase time as the anticipatory levels increased. Unfortunately, only in the none and business levels does the decreased time in the likely phase offset the increased time spent in the highly-likely phase. Notice that at the complete level, the overall processing time is longer than in the business level. This is a result of the highly-likely phase taking more time because of the increased amount of input due to it being a level of greater anticipation, and this additional time is not offset by an equally reduced likely phase. The problem is discussed more in Section 5.10 on page 141.

Figures 47 and 48 graphically represent this information, without and with run-time guidance, respectively.

![Figure 47: Highly-likely and likely phase times for sentence 4 without run-time guidance.](image-url)
5.8.2 Run-Time Guidance Effects

As is to be expected, the highly-likely phase became longer with the addition of run-time guidance. The unexpected result was that the likely phase also slightly increased in every case (by less than 1%). This can be attributed to the system being left in a different state when utilizing run-time guidance. Another explanation as to why the likely phase slowed down from the run-time guidance is that the pragmatic features that were triggered by the logicogrammatical features in the highly-likely phase were “completely triggered.” This means that all the pragmatic feature’s preselected logicogrammatical features were already available in the syntactic parse. The only benefit then of run-time guidance was to acquire pragmatic features early. The statistics of the performance are in Table 9 and Figure 49 on page 136.
Table 9: Run-time guidance effects on sentence 4.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>12.71</td>
<td>14.04</td>
<td>-10.46%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>87.07</td>
<td>87.51</td>
<td>-0.51%</td>
</tr>
<tr>
<td>All</td>
<td>All Processing</td>
<td>30.14</td>
<td>30.58</td>
<td>-1.47%</td>
</tr>
<tr>
<td>Business</td>
<td>Highly-likely</td>
<td>15.62</td>
<td>16.87</td>
<td>-8.00%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>60.52</td>
<td>60.87</td>
<td>-0.58%</td>
</tr>
<tr>
<td>All</td>
<td>All Processing</td>
<td>36.82</td>
<td>37.14</td>
<td>-0.81%</td>
</tr>
<tr>
<td>Complete</td>
<td>Highly-likely</td>
<td>19.32</td>
<td>20.69</td>
<td>-7.09%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>58.16</td>
<td>58.69</td>
<td>-0.91%</td>
</tr>
<tr>
<td>All</td>
<td>All Processing</td>
<td>77.48</td>
<td>78.69</td>
<td>-2.45%</td>
</tr>
</tbody>
</table>

Figure 49: Processing speedup caused by run-time guidance for sentence 4.
5.9 Sentence #5 - “Can you leave earlier?”
This sentence had three levels of anticipation under which to be tested; none, average and complete. Though the figures and tables refer to the business level, this is a case where having no anticipation for a certain level implies that the previous level's information is applicable. Hence, having average expectations and no explicit business expectations means that the business anticipatory level is the same as the average anticipatory level.

The case of no anticipation gives the basic syntactic parse. The average level gives the additional information that the time element is an adjunct, the sentence is neither progressive nor perfect tense, “earlier” is a comparative adjunct, and that the functions of Process and Modal will conflate with the parsed functions of Predicator and Finite, respectively. The complete level further identifies that the indicative interactant subject is the addressee (“you”) and that “you” is singular.

5.9.1 Effects of Anticipatory Guidance
Table 10 gives the processing times for sentence 5 to run under various anticipatory levels.
Table 10: Effects of anticipatory guidance on sentence 5.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>None</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Run-time</strong></td>
<td>Highly-likely</td>
<td>9.60</td>
<td>12.58</td>
<td>17.64</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>99.82</td>
<td>75.98</td>
<td>49.89</td>
</tr>
<tr>
<td><strong>Run-time</strong></td>
<td>Highly-likely</td>
<td>9.93</td>
<td>12.90</td>
<td>18.69</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>100.30</td>
<td>76.40</td>
<td>50.17</td>
</tr>
</tbody>
</table>

This sentence performed completely as expected: the highly-likely phase took longer, the likely phase took less time, and the net result was that the overall processing decreased from one level to the next. This information is depicted graphically in Figures 50 and 51.

![Graph](image)

Figure 50: Highly-likely and likely phase times for sentence 5 without run-time guidance.
5.9.2 Run-Time Guidance Effects

Table 11 and Figure 52 depict the effect of run-time guidance on sentence 5. This information shows that all the anticipatory levels for both the highly-likely and likely phases incurred a slowdown in processing with the inclusion of run-time guidance. The slight slowdown in the likely phase (about .5%) can be attributed to the system being in a different state at the beginning of the phase than it was without run-time guidance. Also noteworthy is that in both the none and business levels, the run-time guidance resulted in no pragmatic features being selected, hence no additional semantic features to help in the likely phase. At the complete level, a pragmatic feature was chosen that helped the likely phase to some extent, but not enough to compensate for the lengthier highly-likely phase.
Table 11: Run-time guidance effects on sentence 5.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Anticipation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>9.60</td>
<td>9.93</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>99.82</td>
<td>100.30</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>89.42</td>
<td>100.72</td>
</tr>
<tr>
<td><strong>Business</strong></td>
<td>Highly-likely</td>
<td>12.58</td>
<td>12.90</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>75.98</td>
<td>76.40</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>65.73</td>
<td>66.90</td>
</tr>
<tr>
<td><strong>Complete</strong></td>
<td>Highly-likely</td>
<td>17.64</td>
<td>18.69</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>49.89</td>
<td>50.17</td>
</tr>
<tr>
<td></td>
<td>All Processing</td>
<td>34.65</td>
<td>35.36</td>
</tr>
</tbody>
</table>

Figure 52: Processing speedup caused by run-time guidance for sentence 5.
5.10 The Effect of Anticipatory Guidance on Sentences 4 and 5

Table 12 shows the speedups that were attained from one anticipatory level to the next for sentences 4 and 5. Generally, there is the expected slowdown in the highly-likely phase, a speedup in the likely phase, and overall speedup in processing. The exception to this is in sentence 4, from the business to the complete levels, for both run-time and no run-time guidance. In these cases, though the likely phase did incur a speedup, it was not enough to compensate for the additional time used in the highly-likely phase to process the additional anticipated features. The problem in the case of no run-time guidance is that the additional anticipated features added to the input slowed the highly-likely phase processing, but the features were predominantly those that were default features and would have been quickly found by the likely phase. In the case of run-time guidance, in addition to the problem that the no run-time guidance method incurred, the additional anticipations in the complete level was not enough to cause any additional pragmatic features to fire so there were no additional logicogrammatical features to reduce the likely phase time. Figures 53 and 54 graphically depict this information.
Table 12: Speedup caused by increasing levels of anticipation for sentences 4 and 5.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>None</th>
<th>Business</th>
<th>Complete</th>
<th>None</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>-22.90%</td>
<td>-23.69%</td>
<td>-52.01%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>30.49%</td>
<td>3.90%</td>
<td>33.20%</td>
<td></td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>-20.16%</td>
<td>-22.64%</td>
<td>-47.36%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>30.44%</td>
<td>3.58%</td>
<td>32.93%</td>
<td></td>
</tr>
</tbody>
</table>

Sentence #5: Can you leave earlier?

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>None</th>
<th>Business</th>
<th>Complete</th>
<th>None</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>-31.04%</td>
<td>-40.22%</td>
<td>-83.75%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>23.88%</td>
<td>34.34%</td>
<td>50.02%</td>
<td></td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>-29.91%</td>
<td>-44.88%</td>
<td>-88.22%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>23.83%</td>
<td>34.33%</td>
<td>49.98%</td>
<td></td>
</tr>
</tbody>
</table>
Figure 53: Processing speedup caused by increasing the level of anticipation for sentences 4 and 5, without run-time guidance.

Figure 54: Processing speedup caused by increasing the level of anticipation for sentences 4 and 5, with run-time guidance.
5.11 Sentence #6 - “Let me look.”

This sentence had two levels of anticipation under which to be tested; none and business.

The case of no anticipation gives the basic syntactic parse. The business level gives several default features and also specifies that the Process of the Beta is the word “look” (the syntactic parse gave “look” as the Predicator).

5.11.1 Effects of Anticipatory Guidance

Table 13 gives the processing times for sentence 6 to run under various anticipatory levels. As can be seen in this table, the sentence performed exactly as predicted. The information is presented graphically in Figure 55 and 56.

Table 13: Effects of anticipatory guidance on sentence 6.

<table>
<thead>
<tr>
<th>Anticipation:</th>
<th>None</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Guidance:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>No Run-time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>11.30</td>
<td>14.75</td>
</tr>
<tr>
<td>Likely</td>
<td>112.17</td>
<td>94.06</td>
</tr>
<tr>
<td><strong>Run-time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highly-likely</td>
<td>14.86</td>
<td>17.37</td>
</tr>
<tr>
<td>Likely</td>
<td>99.00</td>
<td>84.18</td>
</tr>
</tbody>
</table>

All Processing: 125.47 108.30
Figure 55: Highly-likely and likely phase times for sentence 6 without run-time guidance.

Figure 56: Highly-likely and likely phase times for sentence 6 with run-time guidance.
5.11.2 Run-Time Guidance Effects

This sentence is an example of the ideal situation. As already seen, the anticipatory guidance had the exact effect that was expected. Likewise, the run-time guidance has the exact effect that was expected; speedup occurs with run-time guidance.

Table 14: Run-time guidance effects on sentence 6.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anticipation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>11.30</td>
<td>14.86</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>112.17</td>
<td>99.00</td>
</tr>
<tr>
<td>Complete</td>
<td>Highly-likely</td>
<td>14.75</td>
<td>17.37</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>94.06</td>
<td>84.18</td>
</tr>
</tbody>
</table>

Figure 57: Processing speedup caused by run-time guidance for sentence 6.
5.12 Sentence #7 - "Well, all I’m showing is a 10pm flight."

This sentence had two levels of anticipation under which to be tested; none and business. Since it lacked the complete level, it is safe to assume that a business person has the same anticipations as a domain expert, which is why the processing times for the business level are listed under the complete level.

The case of no anticipation gives the basic syntactic parse. The business level gives some default features, specifies that the Subject is nominal and singular, the sentence is not progressive and that the Process of the main clause is a form of "be."

5.12.1 Effects of Anticipatory Guidance

Table 15 gives all the processing times for sentence 7 to run under various anticipatory level influences. As this table shows, the sentence reacts exactly as described in Section 5.3 on page 108.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation:</th>
<th>None</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>60.42</td>
<td>73.20</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>598.08</td>
<td>478.70</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>78.17</td>
<td>92.12</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>447.86</td>
<td>337.95</td>
</tr>
</tbody>
</table>

Table 15: Effects of anticipatory guidance on sentence 7.
Figure 58: Highly-likely and likely phase times for sentence 7 without run-time guidance.

Figure 59: Highly-likely and likely phase times for sentence 7 with run-time guidance.
5.12.2 Run-Time Guidance Effects

This sentence displayed the expected results that run-time guidance should have on the processing times; that the times should decrease with the inclusion of run-time guidance.

Table 16: Run-time guidance effects on sentence 7.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anticipation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>60.42</td>
<td>78.17</td>
</tr>
<tr>
<td>Likely</td>
<td>598.08</td>
<td>447.86</td>
<td>25.12%</td>
</tr>
<tr>
<td>Complete</td>
<td>Highly-likely</td>
<td>73.20</td>
<td>92.12</td>
</tr>
<tr>
<td>Likely</td>
<td>478.71</td>
<td>337.95</td>
<td>29.40%</td>
</tr>
</tbody>
</table>

Figure 60: Processing speedup caused by run-time guidance for sentence 7.
5.13 Sentence # 8 - “I’ve got a seat on an 11:00 flight on Treetop Airlines.”

This sentence had two levels of anticipation under which to be tested; none and complete.

The case of no anticipation gives the basic syntactic parse. The complete level gives some default features.

5.13.1 Effects of Anticipatory Guidance

This sentence’s anticipatory guidance effect was exactly as predicted and can be seen in Table 17 and Figures 61 and 62.

Table 17: Effects of anticipatory guidance on sentence 8.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>Anticipation: High</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely</td>
<td>36.68</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>360.32</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely</td>
<td>51.25</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>269.98</td>
</tr>
<tr>
<td>All-Processing</td>
<td></td>
<td>69.23</td>
</tr>
</tbody>
</table>
5.13.2 Run-Time Guidance Effects

For this sentence, the anticipatory level of none had the expected result of run-time guidance giving some speedup, but the complete level resulted in a slowdown. The reason for this is that the additional features at the complete level did not cause any additional pragmatic features to be selected by the run-time guidance. In fact, some of the features
 included in the complete level were found during the processing of the anticipatory level of none with run-time guidance.

Table 18: Run-time guidance effects on sentence 8.

<table>
<thead>
<tr>
<th></th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Anticipation:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>36.68</td>
<td>51.15</td>
<td>-39.45%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>360.32</td>
<td>270.08</td>
<td>25.04%</td>
</tr>
<tr>
<td><strong>Complete</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Highly-likely</td>
<td>56.91</td>
<td>76.95</td>
<td>-35.21%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>158.81</td>
<td>163.60</td>
<td>-3.02%</td>
</tr>
</tbody>
</table>


5.14 The Effect of Anticipatory Guidance on Sentences 6-8

In sentences 6-8, the additional information made available by anticipation resulted in faster processing of the sentences. This information is made available in Table 19 and Figures 65 and 64 on page 155.
Table 19: Speedup caused by increasing levels of anticipation for sentences 6-8.

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>None Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sentence 6: Let me look.</strong></td>
<td></td>
</tr>
<tr>
<td>No Run-time</td>
<td>Highly-likely -30.53%</td>
</tr>
<tr>
<td></td>
<td>Likely 16.15%</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely -16.89%</td>
</tr>
<tr>
<td></td>
<td>Likely 14.97%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>None Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sentence 7: Well, all I'm showing is a 10pm flight.</strong></td>
<td></td>
</tr>
<tr>
<td>No Run-time</td>
<td>Highly-likely -21.15%</td>
</tr>
<tr>
<td></td>
<td>Likely 19.96%</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely -17.85%</td>
</tr>
<tr>
<td></td>
<td>Likely 24.54%</td>
</tr>
</tbody>
</table>

**Sentence 8: I've got a seat on an 11:00 flight on Treetop Airlines.**

<table>
<thead>
<tr>
<th>Guidance:</th>
<th>None Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Run-time</td>
<td>Highly-likely -55.15%</td>
</tr>
<tr>
<td></td>
<td>Likely 55.93%</td>
</tr>
<tr>
<td>Run-time</td>
<td>Highly-likely -50.15%</td>
</tr>
<tr>
<td></td>
<td>Likely 39.40%</td>
</tr>
</tbody>
</table>
Figure 64: Processing speedup caused by increasing the level of anticipation for sentences 6-8, without run-time guidance.

Figure 65: Processing speedup caused by increasing the level of anticipation for sentences 6-8, with run-time guidance.
5.15 Sentence # 9 - “When can I bring you back?”

This sentence had only one level of anticipation; none. Therefore, this sentence can not be analyzed for anticipatory guidance effects, but only for the effect of run-time guidance.

5.15.1 Run-Time Guidance Effects

Run-time guidance had a positive, expected effect on this sentence, as can be seen in the following table and figure.

Table 20: Run-time guidance effects on sentence 9.

<table>
<thead>
<tr>
<th>Anticipation:</th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>15.14</td>
<td>21.47</td>
<td>-41.81%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>268.63</td>
<td>208.78</td>
<td>22.28%</td>
</tr>
</tbody>
</table>

Figure 66: Processing speedup caused by run-time guidance for sentence 9.
5.16 Sentence #10 - "The round-trip fare will be $295."

This sentence was only tested with the case of no anticipation. As was true with the previous sentence, this means that only the effect of run-time guidance can be evaluated.

5.16.1 Run-Time Guidance Effects

The effect of run-time guidance can be seen in the following table and figure. Once again, the expected result of run-time guidance being beneficial is evident.

Table 21: Run-time guidance effects on sentence 10.

<table>
<thead>
<tr>
<th>Anticipation:</th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>14.62</td>
<td>20.72</td>
<td>-41.72%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>221.60</td>
<td>86.12</td>
<td>61.14%</td>
</tr>
</tbody>
</table>

Figure 67: Processing speedup caused by run-time guidance for sentence 10.
5.17 Sentence #11 - "When can you leave?"

This sentence, like the previous two, also only has one level of anticipation; none.

5.17.1 Run-Time Guidance Effects

Run-time guidance has the expected results as depicted in the following table and figure.

Table 22: Run-time guidance effects on sentence 11.

<table>
<thead>
<tr>
<th>Anticipation:</th>
<th>Guidance:</th>
<th>No Run-time</th>
<th>Run-time</th>
<th>% Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Highly-likely</td>
<td>9.31</td>
<td>14.07</td>
<td>-51.13%</td>
</tr>
<tr>
<td></td>
<td>Likely</td>
<td>66.86</td>
<td>61.93</td>
<td>7.37%</td>
</tr>
<tr>
<td></td>
<td>All processing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 68: Processing speedup caused by run-time guidance for sentence 11.
5.18 Excuse me? Can you repeat that?

Up to this point, all the experiments have operated on the assumption that the input is correct and complete. This is akin to the assumption that what the brain parses the sound signals to be, and what was actually said, are always identical. In reality, though, what we think we hear as well as what we expect to hear are often not what was said at all! Since SAPI uses abduction to create semantic and pragmatic interpretations, it should be able to cope with noisy inputs better than deductive approaches to interpretation. Noisy input can be noisy in two ways; it can be incorrect or incomplete. This section explores how SAPI abductively copes with these two distinct problems of correctness and completeness.

5.18.1 Correctness

Correctness is used in the sense that the input provided to SAPI is part of the accurate description of the sentence. All the test cases have supplied correct input. Though it would be interesting to test SAPI's capability of handling incorrect input, it is beyond the scope of this research. This section will address the various ways in which the input could be incorrect. It will also point out some of the issues surrounding SAPI's ability to cope with incorrect input, some of which have already been addressed in the implementation of SAPI while others have not.

An incorrect parse can be incorrect in a variety of ways:

1. Syntactic parse correct, but not applicable to the sentence in this situation.

   In this case, the syntactic parser supplies a parse that is not contextually appropriate for the sentence. For example, there are multiple syntactic parse trees for “I saw the man on the hill with the telescope,” one of which has the man on the hill with the telescope and
another has the man being seen through a telescope. Both syntactic parse trees are correct, but only one is applicable to the situation. Testing a correct parse that applies to a sentence in a different situation would be akin to testing the system to see if it can produce a semantic interpretation for additional sentences not included in the original corpus for which the system was designed. This semantic interpretation would then be checked for correctness by examining its resulting pragmatic interpretation. Since the current pragmatic stratum was not designed to interpret more sentences, this case of incorrectness can not be tested. Theoretically, the semantic interpreter should be able to understand the correctly parsed syntactic sentences.

2. Syntactic parse incorrect (structurally inconsistent).

This case refers to the information produced by the syntactic parser being inconsistent, hence incorrect. This form of input would suggest that the syntactic parser was not working correctly. It is not just the wrong parse, but it is incorrect; an impossible alternative parse. Incorrect input will result in SAPI producing a partial semantic interpretation that explains as much as it confidently can. No complete semantic interpretation is possible from garbled input.

3. Syntactic parse and anticipatory guidance disagree, and one is incorrect.

All the tests previously performed had both the anticipatory guidance and the syntactic parse being correct. This case would test the operation of SAPI if one of the two inputs were incorrect. Four different options to test exist, each differing on the source of the incorrect information and the source of information to believe:

a) Believe the correct one, where the syntactic parse is correct.
b) Believe the correct one, where the anticipatory guidance is correct.

c) Believe the incorrect one, where the syntactic parse is correct.

d) Believe the incorrect one, where the anticipatory guidance is correct.

Cases (b) and (d) have the syntactic parse being incorrect. It has already been
discussed in the first two cases that this is not reasonable to test. The cases (a) and (c) have
the anticipatory guidance being incorrect while the syntactic parse remains correct. The
situations of (a) and (c) ought to be tested, and the issues to make such a test possible will
now be discussed.

Input from the syntactic parser as well as that from anticipatory guidance are
utilized during the highly-likely phase. Both inputs are expected to be correct, therefore
both forms of input are considered highly-likely. A problem occurs if one of the input is in
fact incorrect; both the correct and incorrect input information would be considered highly-
likely. Here is the reason that the system does not use the concept of "essential." An
essential could not be removed from the composite. If "essential" were used, both the
syntactic (bottom-up) and anticipatory (top-down) input would have deemed themselves
essential. Rather than making a seemingly odd claim that "an essential is only essential as
long as it does not conflict with another essential," the ranking of highly-likely is used.
When a conflict occurs between highly-likely alternative hypotheses, meta-knowledge is
used to determine in which alternative to place higher confidence. This issue is discussed
in more detail in Section 6.4.4 on page 182.

SAPI already avoids the existence of "essential" features and has meta-knowledge
entered by the user as to which input source should be given the greater amount of
confidence. The changes required to SAPI in order to handle incorrect data is the creation of additional rules to take advantage of this meta-knowledge of which input source to believe, syntax or pragmatics. A pair of rules (one believing the syntax and the other believing the pragmatics) would have to be written that would use the meta-knowledge to remove the effects of the conflicting feature from the incorrect input source. To remove these effects, a set of rules would need to be written that would traverse the hierarchy to the leaf-most features originating from or caused by the offending source that are related to the conflicting feature. All the features along this path would need to have their realization rules undone.

The reason that SAPI's ability to cope with incorrect input was not tested is because performing such tests would require changing SAPI's code. Altering the code would cause all the processing times of the other experiments to become inaccurate, no longer reflecting the time required by SAPI to create a semantic interpretation. Every intention exists to add the necessary rules to test this form of noisy input, but this will be done later. The problem of noisy data, where the noise is caused by the data being incomplete was tested with pleasing results. The following section explores this issue.

5.18.2 Completeness

All the previous test cases had complete input; all the words were present in the input with their associated features from the syntactic parse. This is idealistic since people do not always hear all that is said to them. Sometimes a word is missed or ignored and yet people somehow manage to understand what is said well enough to function properly. Perhaps people just need enough of the input to get the general gist of the statement, or perhaps the
abduction used in NLU is able to fill in the gaps. Regardless of how people perform the task, it would be interesting how SAPI could cope with incomplete input.

Two sentences were chosen to test SAPI's ability to handle incomplete correct input. Those two sentences were “What time can you leave?” and “I can put you on that 10pm flight.” The reason these two sentences were selected is that they were the two sentences with the shortest and longest processing times that had all four levels of anticipation.

Each of the two sentences’ inputs were altered such that at the word level, all the features from the syntactic parse associated with a specific word were removed from the input. The word itself was also removed from the input. Each of these altered inputs were processed by SAPI with each of the four levels of anticipation with and without run-time guidance. The correct understanding of this sentence would be the pragmatic features DEPARTURE-DATE-KNOWN, CHANCES-NOT-GOOD, CLIENT-ORIENTED, and INFORMAL. The results of the test on the first sentence are in Table 23.

Table 23: Results of the incompleteness test on “What time can you leave?”

<table>
<thead>
<tr>
<th>Anticip.</th>
<th>None</th>
<th>Average</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing Word</td>
<td>RT</td>
<td>RT</td>
<td>RT</td>
<td>RT</td>
</tr>
<tr>
<td>What</td>
<td>X-B</td>
<td>X-B</td>
<td>X-B</td>
<td>X-B</td>
</tr>
<tr>
<td>time</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>can</td>
<td>X-B</td>
<td>X-B</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>you</td>
<td>X-B</td>
<td>X-B</td>
<td>X-B</td>
<td>*</td>
</tr>
<tr>
<td>leave</td>
<td>X-B</td>
<td>X-B</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
The following is the key to the table:

- **•** means the correct pragmatic features were found from the semantic interpretation.
- **“A”** means that a pragmatic interpretation was found, but instead of having the feature DEPARTURE-DATE-KNOWN, its parent ORIGINAL-DEP-QUESTION was part of the explanation.
- **“X”** means no semantic interpretation of the sentence was possible.
- **“X-B”** means no semantic interpretation of the sentence was possible, but the run-time guidance did find the pragmatic feature DEPARTURE-DATE-KNOWN.

Of the 40 cases, the sentence was still completely understood in 14 of them, resulted in no semantic interpretation in 11 cases, and partially interpretable in 15 cases. Of the 15 partially interpretable cases, eleven did not reach a semantic interpretation. The partial interpretation was found by the run-time guidance.

It may be expected that as the level of anticipation increased, the quality of the pragmatic interpretation would increase. In both the cases of run-time and no run-time guidance this was true for 15 out of the 15 cases\(^1\). It may also be expected that the inclusion of run-time guidance should increase the quality of the pragmatic interpretation. Of the 20 possible comparisons, this held true for all 20 cases\(^2\).

The reason SAPI could not come to a semantic interpretation in all cases is a result of the lexification rule. Generally, in order to allow a feature with a lexification rule to be included in the composite, the word that the lexification rule lexifies to must be present in the input data. The exception to this is if the feature with the lexification rule is being

---

1. A comparison consists of looking from one level of anticipation to the next higher level. If the interpretation remained the same or improved, it was considered to have operated as expected.
2. For this comparison, * is considered better than A, and X-B is considered better than X. If the interpretation remained the same or improved, it was considered to have operated as expected.
considered as an addition to the composite as a result of highly-likely features, then the lexification rule is permitted to take effect since that feature must exist. The result is that if SAPI is trying to lexify during the likely-phase, it has to backtrack since none of the lexification words exist in the input. SAPI either backtracks completely out of all processing since it can not resolve the problem, or ends up in a loop of lexification attempts because there is no feature that it can remove from the composite.

One reason that SAPI could occasionally complete processing with a word missing is that it would try to lexify to the missing word during the highly-likely phase. Highly-likely features determined that the feature with the lexification of the missing word had to exist. Though the lexification was not legal since the word was missing from the input, SAPI has no rule to stop a lexification that is mandated by highly-likely features. Another situation where SAPI could complete processing with a word missing is if there was no information to cause SAPI to start traversal in a word-level hierarchy. If no traversal ever occurred there, SAPI could ignore the existence and the need to lexify to that word.

The second sentence “I can put you on that 10pm flight.” also had each of its words removed and the ability of SAPI to create an understanding was tested. The correct pragmatic interpretation includes the features: INFORMAL, FLYING TIMES, CHANCES-GOOD, NEED-SAT-KNOWN, AGENT-ORIENTED, DONT-MENTION-
AIRLINE, and FLIGHT-MENTIONED-PREVIOUSLY. The results of the tests are shown in Table 24.

Table 24: Results of the incompleteness test on “I can put you on that 10pm flight.”

<table>
<thead>
<tr>
<th>Anticip.</th>
<th>None</th>
<th>Average</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing Word</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>X</td>
<td>X-D</td>
<td>X-D</td>
<td>*</td>
</tr>
<tr>
<td>can</td>
<td>X</td>
<td>X-D</td>
<td>X-D</td>
<td>X-E</td>
</tr>
<tr>
<td>put</td>
<td>X</td>
<td>X-D</td>
<td>X-D</td>
<td>X-E</td>
</tr>
<tr>
<td>you</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>on</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>that</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>10pm</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>flight</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

The following is the key to the table:

- “*” means the correct pragmatic features were found from the semantic interpretation.
- “X” means no semantic interpretation of the sentence was possible.
- “X-D” means no semantic interpretation of the sentence was possible, but the run-time guidance did find the pragmatic feature FLIGHT-MENTIONED-PREVIOUSLY.
- “X-E” means no semantic interpretation of the sentence was possible, but the run-time guidance did find the pragmatic features FLIGHT-MENTIONED-PREVIOUSLY and AGENT-ORIENTED.

Of the 64 cases, the sentence was still completely understood in 42 of them, resulted in no semantic interpretation in 13 cases, and partially interpretable in nine cases. Of the
nine partially interpretable cases, all of them found a partial interpretation as a result of run-time guidance.

This sentence, just like the previous one, had the expected results. As the level of anticipation increased, the quality of the pragmatic interpretation increased in all 24 of the no run-time guidance cases and in all 24 of the run-time guidance cases. The inclusion of run-time guidance also increased the quality of the pragmatic interpretation in all 32 cases.

These two sentences suggest that SAPI generally can continue operation even when a word is missing in the input. For these two sentences, SAPI found some portion of the pragmatic interpretation in 77% of the cases and found the complete pragmatic interpretation in 54% of the cases. Increased anticipatory guidance results in the same or better pragmatic interpretation 100% of the time, regardless of the existence of run-time guidance. And finally, in 100% of the cases, run-time guidance provided the same or a better interpretation than operating without run-time guidance at the same level of anticipation.

5.19 Conclusion

The purpose of this chapter is to present the experimental results of SAPI. The conditions under which the experiments were run were explained and the expected results were discussed. Then each of the various experiments was discussed in detail with their exact processing times. Each sentence was described in terms of its anticipatory levels and what each level added to the input. Also, for each sentence were tables and figures that specified the time it took for the semantic interpreter to complete processing. These tables were organized to make clear the effects due to anticipation of features in contrast to the effects
of run-time guidance. Then each group of sentences was presented in terms of percent speedup caused by the increase in the level of anticipation. Finally, SAPI was tested to see how well it could continue operating with incomplete input. The test results suggested that SAPI can overcome a missing word in the input and still determine some portion of a pragmatic interpretation. SAPI, because it uses an abductive strategy, is able to cope with noisy data with little modification, unlike a deductive system which would require many additional inference rules. The various results of the experiments were not always as originally expected, but they are all valuable in their own right. The results and the meaning of them will be discussed in more depth in the next chapter.
CHAPTER VI
Discussion of Results

6.1 Introduction
The purpose of this chapter is to give a detailed review of the performance of the system and to discuss the level to which it met its objectives. Chapter 5 presented the timing results on a sentence-by-sentence basis with no discussion of the “big picture;” how the concepts presented in the research enhanced the overall performance. This chapter will tackle that question.

The chapter begins by restating the expected results of incorporating run-time guidance and anticipatory guidance. This will be followed by a summary of the sentences and their times to finish processing under each circumstance. This summary is then discussed in terms of the result of using anticipatory guidance with no regard to the effect of run-time guidance. Then the timings are discussed in terms of using run-time guidance, ignoring the anticipatory guidance. A further interpretation of the results is discussed in terms of anticipatory guidance with and without run-time guidance. The chapter concludes with a discussion of meta-knowledge about when to apply top-down guidance.
6.2 Expected Results

Before analyzing the results of the system, it is beneficial to review the projected results. The results can be analyzed from two primary viewpoints: the effects of anticipatory guidance and the effects of run-time guidance.

Anticipatory guidance is the bringing to bear of contextual knowledge before semantic interpretation begins. The system was supposed to have an increase in performance with the inclusion of anticipatory guidance. The more a person knows and expects in a given situation, or the more information used as input into a system, the more quickly he/she/it should be able to decipher the meaning.

Run-time guidance is the bringing to bear of contextual knowledge during semantic interpretation. The system was supposed to have an increase in performance with the inclusion of run-time guidance. The run-time guidance was supposed to decrease the likely phase processing time by increasing the number of highly-likely hypotheses found in the highly-likely phase. As the highly-likely hypotheses from the logicogrammatical stratum are placed in the composite, the pragmatic interpreter attempts to account for them with hypotheses from the pragmatic stratum. The pragmatic feature hypotheses that are found may help further determine the literal meaning by preselecting logicogrammatical features to add to the composite.

To summarize, individually run-time guidance and anticipatory guidance should decrease the processing time of determining the semantic, or literal meaning from the syntactic parse. Since individually they should be beneficial, then when utilized together, the thought is that the resulting processing time should be even more impressive. The
rationalization behind this idea is that the decrease in processing time of the two strategies should be additive.

6.3 Results

The input sentences to the system were:

Table 25: Input sentences.

<table>
<thead>
<tr>
<th>Sentence #</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What time can you leave?</td>
</tr>
<tr>
<td>2</td>
<td>I can put you on that 10pm flight.</td>
</tr>
<tr>
<td>3</td>
<td>How many seats are you looking for?</td>
</tr>
<tr>
<td>4</td>
<td>I'm not finding anything then</td>
</tr>
<tr>
<td>5</td>
<td>Can you leave earlier?</td>
</tr>
<tr>
<td>6</td>
<td>Let me look.</td>
</tr>
<tr>
<td>7</td>
<td>Well, all I'm showing is a 10pm flight.</td>
</tr>
<tr>
<td>8</td>
<td>I've got a seat on an 11:00 flight on Treetop Airlines.</td>
</tr>
<tr>
<td>9</td>
<td>When can I bring you back?</td>
</tr>
<tr>
<td>10</td>
<td>The round-trip fare will be $295.</td>
</tr>
<tr>
<td>11</td>
<td>When can you leave?</td>
</tr>
</tbody>
</table>

These eleven sentences of varying complexity were the input to the system along with varying levels of anticipation. The amount of anticipation was dependent on the expectations that a person of the various backgrounds would have in the travel agent domain. The levels defined here are none, average person, business person and complete. Not all the sentences have the same anticipatory levels defined because, for example, in some situations an average person will have the same expectations as a business person.
Table 26 summarizes the times for the system to create a complete and correct semantic interpretation for each sentence.

Table 26: Processing time in average number of seconds of eleven sentences with varying levels of anticipatory guidance, with and without run-time guidance.

<table>
<thead>
<tr>
<th>Anticip.</th>
<th>None</th>
<th>Average</th>
<th>Business</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No RT</td>
<td>RT</td>
<td>No RT</td>
<td>RT</td>
</tr>
<tr>
<td>Sentence #</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>145.87</td>
<td>122.29</td>
<td>114.37</td>
<td>95.01</td>
</tr>
<tr>
<td>2</td>
<td>243.19</td>
<td>236.92</td>
<td>170.26</td>
<td>144.78</td>
</tr>
<tr>
<td>3</td>
<td>132.71</td>
<td>130.49</td>
<td>128.73</td>
<td>113.32</td>
</tr>
<tr>
<td>4</td>
<td>109.78</td>
<td>101.55</td>
<td>77.74</td>
<td>79.38</td>
</tr>
<tr>
<td>5</td>
<td>110.23</td>
<td>89.30</td>
<td>68.86</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>113.86</td>
<td>113.86</td>
<td>101.55</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>526.03</td>
<td>321.23</td>
<td>430.07</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>321.23</td>
<td>230.25</td>
<td>215.72</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>106.84</td>
<td>76.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>230.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>76.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.3.1 Results of Anticipatory Guidance

The results of utilizing anticipatory guidance to speed up the processing time were quite encouraging. In 30 out of 32 cases (15 out of 16 cases with no run-time guidance, and 15 out of 16 cases with run-time guidance), the greater the level of priming, the faster a complete semantic interpretation was determined. The exception was sentence 4; the business level required less time to finish processing than the complete level with and without run-time guidance. These two cases had a 2.11% and 1.76% slowdown,
respectively. Why, of the 32 cases did the same sentence under the same anticipatory levels, with and without run-time guidance, encounter a slowdown? To shed some light on this question, the input data for that sentence needs to be reviewed.

Beyond the anticipatory level of none, the business level merely specified that the sentence had a non-textual-theme and the Subject was the speaker (speaker-subject). Being informed that a sentence has a feature of non-textual theme is not a big computational time-saver as non-textual-theme is a default feature and will be added to the composite at the beginning of the likely phase. Speaker-subject is a time-saving feature since there is no default for the system indicative-interactant-subject whose children include speaker-subject, addressee-subject and subject-plus-speaker. These features are all considered of equal plausibility. The identification of the Subject can be computationally expensive because of frequent backtracking. Primarily because of this feature, the processing time at the business level was less than the processing time at the no anticipatory guidance level.

The additional information at the complete level included that the clause was an unmarked-negative and time-adjunct, the Medium was non-relativized, non-qualified and generic-nominal, the head of the Temporal function was unmarked-adjunct, and the function of Process existed with the feature for the word "finding." With all this additional information, the highly-likely phase took quite a bit more processing time regardless of whether there was run-time guidance or not. Additional information from anticipatory guidance is most beneficial when it lessens the processing time in the likely phase, which in both of these cases it did; the likely phase time for the complete level is less than the likely phase time for the business level. Therefore, the additional anticipated information
did reduce the number of hypotheses to consider. Unfortunately, the additional anticipated information took up more highly-likely phase processing time than it saved in the likely phase resulting in a net slowdown. The reason the additional features at the complete level did not cause a speedup is best explained by analyzing the benefit of each of the additional features at the complete level.

- **Unmarked negative, non-relativized and non-qualified** are all default features and would have been added to the composite at the start of the likely phase.

- **Medium** is conflated with Residual by a default feature and Residual already has the feature generic-nominal assigned to it.

- **Unmarked-adjunct** is one of the parents of the gate feature time, which is a feature from the syntactic input.

- **Process** was already conflated with Predicator who already had been assigned the feature for “finding.”

From the above points, it is evident that much of the additional anticipated information provided at the complete level was either already determined by processing the syntactic parse or would be quickly found during the likely phase of processing. The only truly beneficial feature provided at the complete level was time-adjunct. The benefits of the inclusion of this one feature could not compensate for all the additional processing caused by the other features. The amount of processing time spent in the highly-likely phase was too long with too little gain for the likely phase to be able to recoup the time expenditure.

Having explained the anomaly in the expected result, and having seen that the likely phase time did decrease with anticipatory guidance, let us turn our attention to the results of run-time guidance.
6.3.2 Results of Run-Time Guidance

The results of including run-time guidance to speed-up the processing time are mixed. In the 27 pairs of sentences (the no run-time guidance and run-time guidance cases for a specific sentence and anticipatory level), 15 pairs incurred a speedup when allowed to use run-time guidance. Of these 15 cases, two had trivial speedup (less than 1%), three had some speedup (between 6% and 8%), eight had significant speedup (between 11% and 23%) and two had remarkable speedup (39.66% and 54.77%). The remaining 12 slowed down; however none of the slowdowns were dramatic. Only two slowed down by greater than 5% (the greatest slowdown was 6.05%).

Of the twelve that slowed down, sentence 3 at the business level actually had a speedup in the likely phase, but it just was not enough to compensate for the longer highly-likely phase. The remaining eleven had a slowdown in the likely phase, but all of these slowdowns were less than 1.12%, with five of them being less than 0.6%. A nominal slowdown can be attributed to the system completing the highly-likely phase in a different state with the incorporation of run-time guidance than without. The different states cause the processing to progress differently, and the different order of consideration of the features and functions will cause the difference in timings.

Of the twelve cases that did not have a speedup with run-time guidance, eight had no additional logicogrammatical features preselected with the inclusion of run-time guidance than those that were added by the less informed anticipatory level. In other words, the additional information provided at the level of greater anticipation did not help guide the likely phase any more than the additional information offered at the lower level of anticipation. The remaining four cases did select a pragmatic feature, but the pragmatic
feature had only one preselection, that which had caused it to be chosen, and the parents to
the pragmatic feature could add no logicogrammatical features to the composite. Hence,
run-time guidance used quite a bit of processing time and provided no additional
logicogrammatical features.

Another evaluation point is to compare when run-time guidance at one level aids
the processing time more than the anticipatory guidance at the next level without run-time
guidance. Of the 16 possible cases of this occurring, there were three actual incidences.

Also worth noting about run-time guidance is that in every experiment, the
slowdown could be eliminated if the processing of the pragmatic and semantic interpreters
were to be performed in parallel rather than interleaved sequentially. Perhaps the problems
correlated with run-time guidance are a matter of implementation, not of the concept in
general.

6.4 Interpretation of Results

Having condensed, evaluated, and rationalized the results of the research, it is now
necessary to determine what exactly it all means. The remainder of this chapter takes a look
at its more concrete meaning, while Chapter VII tackles the further implications.

6.4.1 Bogus Input?

Though the results were not exactly as predicted, they are impressive and important. Firstly,
it is important to mention that the results were acquired by using the input from the parser,
the input that was designated as valid anticipation knowledge, and the same
logicogrammatical and pragmatic strata that were used for the generation of the sentences.
The expected results could have been made more sensational if any of these aspects had been tampered with; moving a feature from one level of anticipation to the next, creating new features in a stratum with the necessary realization rules to acquire a speedup, adding to the input more features that were not available from the parser or the anticipatory guidance, etc. None of these tactics were utilized. The strata were created by one individual, the anticipatory guidance determined by another, the syntactic parse information provided by a third, and the implementation of the system utilizing all these aspects by a fourth. This is important to keep in mind since it enhances the credibility of the results; they were not hand-coded or jerry-rigged.

6.4.2 Anticipatory Guidance without Run-Time Guidance

In general, it appears that as the level of anticipatory guidance increases, the highly-likely phase increases, the likely phase decreases, and the overall processing time decreases. This phenomenon is depicted in Figure 69. In the figure, the lines in the graph are not based on actual processing times, but on the general processing performance seen in the majority of experiments. The highly-likely phase increases linearly assuming that each additional highly-likely feature in the input requires a constant amount of processing time. The likely phase decreases exponentially since each additional highly-likely feature in the highly-likely phase removes a portion of the logicogrammatical hierarchy. The All Processing line is the sum of the other two.

The extreme cases of anticipatory guidance are the situations where there is no anticipations or where all the features are input to the system. In the first case, the majority of the processing time is spent in the likely phase as the highly-likely phase is limited to the syntactic input. In the latter case, all the features are determined so there is no likely phase;
the highly-likely phase time equals the processing time. Usually, the likely phase is longer than the highly-likely phase (due to the possible need for backtracking). Therefore, the more highly-likely hypotheses dictated, the faster the overall processing time. At some point, the additional anticipated features do not produce such a dramatic speedup in processing. This situation is represented in Figure 69 by the boxed Area A. In this area, it would not be beneficial for the system to try to identify additional anticipated features since the time searching for them may overshadow the guidance they can provide to the processing.

![Figure 69: Processing time versus anticipatory guidance without run-time guidance.](image)

The final conclusion is that anticipatory guidance is a valuable way to reduce processing time. The more information readily available up front, the faster the processing.
6.4.3 Anticipatory Guidance with Run-Time Guidance

In the majority of the cases of processing with run-time guidance, the aforementioned projected results still hold: as the level of anticipatory guidance increases, the highly-likely phase increases, the likely phase decreases, and the overall processing time decreases. The possibility exists that the additional anticipated features, when joined with run-time guidance, can actually slowdown the processing. Figure 70 depicts the situation. The difference between Figure 70 and Figure 69 is the line representing the highly-likely phase. Though it can still be assumed that each additional logicogrammatical feature in the input adds a constant amount of computational time to the processing, the line also must include the amount of computational time used to locate a pragmatic feature to account for each logicogrammatical feature. This pragmatic feature will hopefully provide run-time guidance. Each selected pragmatic feature must pass a verification of consistency with all the current logicogrammatical features in the composite. Assume that the time to locate the pragmatic feature takes time $p$, and the verification of a pragmatic feature takes constant time $c$, and there are $n$ logicogrammatical features that can possibly be accounted for by some pragmatic feature. The time to complete this phase will be $O(n(p+c))$. Hence the highly-likely phase line is represented by a line of greater slope than in Figure 69. The All Processing line must reflect this greater slope since it represents the sum of the highly-likely and likely phase lines.

The occurrence of an increase in the net processing time can be attributed to several possible situations. These anomalies are:

1. The search in the pragmatic stratum could be avoided by performing a table look-up where the table represents the pre-computed information of the interstratal preselections.
1. None of the pragmatic features can be selected as a way to account for the existing highly-likely logicogrammatical features in the composite. Time spent to determine the possibility of run-time guidance is wasted since no guidance actually occurs.

2. The existing highly-likely logicogrammatical features cause a pragmatic feature to be selected, but the pragmatic feature was fully-determined by the logicogrammatical features. Fully-determined means that all the logicogrammatical features that the pragmatic feature preselects already exist in the composite. Time spent to determine the possibility of run-time guidance is wasted since no guidance actually occurs.

3. The existing logicogrammatical features cause a pragmatic feature to be selected, and the pragmatic feature adds only one logicogrammatical feature to the composite. If this additional feature is a default feature in the grammar, then it would have been quicker to identify it in the likely phase than in the highly-likely phase.

4. The additional logicogrammatical features provided by the anticipatory guidance substantially aids processing without run-time guidance by reducing the number hypotheses to consider. The run-time guided version is hindered by the additional information since the additional features require additional time for evaluation by the run-time guidance.

Though there seems to be many instances where run-time guidance could be a negative mode of operation, there is an instance where it helps tremendously. When the system is presented with a partial parse, the use of run-time guidance may enable the sentence to be understood, at least partially, in a reasonable amount of time. This
phenomenon is discussed in more detail along with the results of SAPI's performance in such instances in Section 5.18.2 on page 162.

Figure 70: Processing time versus anticipatory guidance with run-time guidance.

A breaking point seems to exist where the inclusion of run-time guidance is more of a hindrance than a help. This point is represented by Point A in Figure 70. Notice that point A does not necessarily co-occur with the instance where the highly-likely phase takes longer than the likely phase. It occurs where the total processing time starts to increase.

Figures 69 and 70 both display a point or region where top-down guidance no longer provides a great processing speedup. This phenomenon generates many questions. Can point A be determined before processing ensues? When should top-down guidance occur? When is top-down guidance necessary? Which level should instigate the
communication of information? To answer these questions, the system requires meta-knowledge and knowledge about when to pass information between the strata for processing.

6.4.4 Meta-Knowledge

If a sequential system is to include top-down guidance, then some information is needed to help determine when the guidance should be applied. But what can determine such knowledge about knowledge, or meta-knowledge?

People often put more emphasis on one source of input than another. For example, illusions are often based on tricks that the visual system plays with the mind, and taste is often affected by what is smelled. In hearing and processing natural language, the human mind must make decisions about what was said and heard. These decisions are presumably produced abductively, trying to best account for the input data.

The comprehension of natural language can be viewed as a bidirectional process, pragmatic $\leftrightarrow$ semantic $\leftrightarrow$ syntactic $\leftrightarrow$ word $\leftrightarrow$ morpheme $\leftrightarrow$ phoneme (or some such mapping variation), where top-down is considered traveling from left to right, and bottom-up is traveling from right to left. Top-down guidance occurs when a concept on the left-side of an arrow aids the processing of the item on the right-side. Hence, in top-down guidance, the pragmatic stratum can aid the semantic layer, the semantic layer can aid the syntactic parse, and so on. In bottom-up processing, the syntactic layer passes its solution up to the semantic layer which processes its information and passes it up to the pragmatic layer. The question that remains is twofold; when is one form of input to be believed over the other and when should information be passed?
People seem to put more emphasis on top-down guidance when they are in an extremely well-known situation. In a very familiar situation, there is a large amount of pragmatic knowledge available so there are a great number of expectations of what will be heard. At times, these expectations enable the listener to finish the speaker's sentence or they may preclude the parse totally, resulting in a return statement that does not match the speaker's statement. For example, "Fine, and you?" being said as an answer to the statement "You look nice today."

Another time when people put more emphasis on top-down guidance is when there is a great likelihood of an incorrect or partial parse. An example of this is when a person is in a noisy place, such as a tavern with music playing loudly. When someone speaks, a few of the words are heard, or perhaps just the body motions are perceived. To understand or figure out what the person said, it must be inferred in a top-down manner, from the context of what is occurring in the room to the word level, and the word level must match the words that were perceived. Though this is common in noisy situations, it will also occur in an unknown situation. The person will call upon any related experiences to help determine what he/she should expect to hear. Context and top-down guidance can be used to resolve ambiguities.

There are times when the parsed information may be preferred over top-down guidance. For example, if a person is heard clearly, and there is no question as what was said regardless of how bizarre or out of place the comment was, then the top-down guidance needs to be ignored. This situation often results in a person thinking to himself, "I can't believe they just said that!" The bottom-up processing information may also take a front
row seat when the topic of communication is changed. The context has not been set for this situation, so it can not give much guidance.

Another situation that occurs is when one direction of processing is favored, but once understanding starts the direction has to be altered. For example, when meeting a person from another country, the listener may be prepared for a certain accent and grammar competency. In other words, the listener allows the context of the situation to prepare him/her for certain syntactic structures as well as words, morphemes and phonemes. If the speaker has full competence of the native tongue of the listener, sometimes the listener is thrown off balance for a moment and has to mentally reparse what was said.

In the system created for this research, the meta-knowledge of which input source to believe was provided. At the beginning of the processing of a sentence, the system asked the user to enter in which input source to place more confidence. It would be interesting if a heuristic could be found that would determine at run-time in which source of input to place greater confidence. What is needed is meta-knowledge, or possibly some new strategy. This new strategy could simulate this heuristic by evaluating the composites at each layer, and placing greater confidence in the source of input that created the highest scoring composite.

Meta-knowledge could also be used to determine when the direction of processing should be changed. Assume that the default processing direction of the system is bottom-up. Meta-knowledge could determine when bottom-up processing should be suspended and the top-down processing enabled. I have identified four possible situations in which top-down guidance is most beneficial:
1. When the syntactic parse is "noisy," either missing segments or prone to incorrect segments. The portions that are known with some degree of confidence may help determine the rest of the parse by allowing the pragmatic layer to guide the processing in the semantic layer which in turn can aid the syntactic parsing.

2. When there are no anticipated features. SAPI encountered a speedup when including run-time top-down guidance in this situation in nine out of eleven cases. The syntactic parse can only create so many features and any additional guidance to determine the literal meaning is helpful.

3. When the anticipated features are correct and are abundant. If there is a great deal of anticipatory guidance, the system has a multitude of features which may completely determine the intended meaning of the sentence. It may be possible, if the intended meaning or gist of the sentence is sufficient, that the semantic processing does not need to complete. Between the syntactic features and the anticipated features and their inheritance, the pragmatic features may be completely determined.

4. When there is ambiguity in the input. Context can help guide the lower levels of understanding, whether they be the semantic or syntactic levels. Examples of ambiguity resolution include resolution of structural ambiguity (I saw the man on the hill with the telescope), lexical ambiguity, ellipsis, substitution, scope, anaphoric and cataphoric reference, adjectival noun phrase modifiers (small elephant), prepositional noun phrase modifiers (destruction by the huns), multiple modifiers (light red car), noun-noun modifiers (pot handles, car paint, stone wall), and tense and aspect (will have been running) (Allen, 1987).
If the syntactic parse has too many possible structures to pursue, then top-down guidance can help reduce the number of alternatives, thereby reducing the chance of an incorrect parse as well as reducing the likelihood of backtracking.

In summary, the use of top-down guidance is generally beneficial, however deciding when to bring this contextual knowledge to bear is the crucial question. I believe the additional information provided by top-down guidance should be used to minimize processing time and maximize the likelihood of a correct understanding. If the combination of top-down (pragmatic) and bottom-up (syntactic) knowledge are in conflict, meta-knowledge should be used to guide the processing. In a sequential system, the processes have to occur one at a time, obviously. In humans, however, the understanding of a sentence may be happening in parallel, on multiple levels, concurrently. It would be interesting to devise a computer system that when an ambiguous situation arose, it could spawn off multiple processes, each with its own copy of the current state of the understanding and with a different alternative path to follow. The first process that arrives at a complete and consistent understanding is considered correct and all other processes will halt. This is called the race-based model of processing [McRoy and Hirst, 1990]. The rationale for accepting the first explanation is that processing time is costly, and the first one done has taken the least amount of time. But these ideas are beyond the scope of this research.

6.5 Conclusion

This chapter has presented the results of the experiments and analyzed them to determine the anticipatory guidance benefits, run-time guidance benefits, and the benefits of combining the two. In general, anticipatory guidance will always reduce processing time
while run-time guidance, as utilized in this work, may or may not speedup the processing. This chapter also raised issues surrounding the inclusion of anticipatory guidance and run-time guidance. Whether the anticipatory guidance uses run-time guidance or not, there seems to be a breaking point (or region) where the acquisition of additional anticipated features may not be computationally beneficial. Predetermination of when this occurs is crucial to be able to determine when run-time guidance should be used and to what extent. The chapter concluded with a discussion of the knowledge needed to make such a pre-determination, referred to as meta-knowledge. It appears that the use of top-down guidance is related to the situation, external to the system itself. The following chapter takes the analysis one step farther and considers the how these concepts can be applied on a larger scale.
CHAPTER VII

Future Work and Further Implications

7.1 Introduction

What has been introduced in the previous six chapters is a computational model for natural language understanding that utilizes additional knowledge in the form of run-time and anticipatory top-down guidance. The system implemented is interesting in and of itself as it represents a NLU system that utilizes a bidirectional systemic grammar representation, an abductive processing strategy, and interleaved semantic and pragmatic interpretation. The concepts behind the system have resulted in some important findings. Still more can be learned from this work that can be applied to the next problem-solver, whether it be human or computer.

This chapter presents future work that can occur to the strata and to the system. The ways in which the research can be applied to other problems as well as the complexity of the abduction techniques are explored. A discussion of a claim made by Noam Chomsky is presented with a rebuttal regarding the use of abduction in natural language understanding. The chapter concludes with a discussion of context and its relationship to NLU and abduction.
7.2 Future Work

This dissertation represents a “frozen” state of the research and the system. In reality, neither has to be frozen, both can be expanded beyond this point. Though SAPI is a working system, it can be improved upon or altered to investigate other concepts. Some areas that would be interesting to change and analyze further are listed in this section.

7.2.1 Expand the Strata

The current system has two strata, the pragmatic stratum and the logicogrammatical stratum. Each of these layers can be expanded. The following list is just a set of suggestions and is not meant to be exhaustive.

7.2.1.1 Pragmatic Stratum

The system currently has a rather small pragmatic stratum. It would be interesting to expand the pragmatic stratum and analyze its interaction with the logicogrammatical stratum. I believe the expansion would create a need for a re-evaluation of the criteria for making pragmatic stratum features highly-likely. The current process of picking pragmatic features requires the pragmatic feature to be the only way to account for the logicogrammatical feature. When the pragmatic stratum is expanded, it is probable that the ways to account for a logicogrammatical feature will increase. This will result in the inability to assign a confidence value of highly-likely to a pragmatic feature. The remedy depends on along which dimension the pragmatic stratum is expanded. If the pragmatic stratum is expanded to account for more domains, then the solution is to identify the domain at the start or as early as possible in the processing. Each domain will have its own sub-classification hierarchy, and by determining the domain, the subhierarchy can be identified
and the other hierarchies can be ignored. It would be less problematic if the pragmatic stratum were altered in the dimension of adding to the current context of a travel-agent. The current pragmatic stratum was designed with the generation of the eleven test sentences in mind, so adding to the pragmatic stratum in this domain will probably create realization rules that preselect other, previously unpreselected, logicogrammatical features. The structure of the hierarchy would become less sparse with the gaps filled in with more features.

The current system uses the pragmatic stratum as it was designed by Patten, Geis and Becker (1992) and can be seen in its entirety in Appendix A on page 220. This stratum could be reorganized to give more guidance. One problem with the pragmatic stratum is there are instances where realization rules are repeated for related pragmatic features. This creates a “heavier” tree than necessary. The problem with repeating realization rules is that one of the requirements for a pragmatic feature to be counted as highly-likely is that it must be the sole way to account for a logicogrammatical feature. If the rule appears with multiple pragmatic features, then none of those pragmatic feature can be assigned the confidence value of highly-likely. If the realization rules were “spread out” more amongst the pragmatic features, eliminating the redundant calls to realization rules where possible, then there could be more selections at the pragmatic level and therefore more guidance. This phenomenon is represented in Figure 71. In the figure, the features are represented by capitalized, bold letters and the preselection rules are identified by the lower case letters. Imagine the scenario where the logicogrammatical stratum had the feature that suggests realization rule “a.” In Case A, no feature could be selected since “a” is accounted for by both X and Y. In Case B, the rule “a” has been moved up to Z. If there are no features in the
logicogrammatical composite that conflict with the other preselections of Z, then Z can become highly-likely. As a result, in Case B the feature Z is chosen because of rule “a” and rules “b” and “e” are able to add features to the logicogrammatical composite.

Case A

\[
\begin{array}{c}
Z_c \\
X_a \\
Y_a \\
\end{array}
\]

Case B

\[
\begin{array}{c}
Z_c \\
X_c \\
Z_a \\
Y_d \\
\end{array}
\]

Figure 71: Reducing the heaviness in a stratum.

### 7.2.1.2 Logicogrammatical Stratum

The system could have the logicogrammatical stratum expanded. In particular it would be most beneficial to eliminate the situation of having multiple “void features” [Kasper, 1988] in a system. Void features are features in the system that do not have any realization rules or do not appear as an input condition to a system. The problem of having void features in the strata is there is no way to identify whether or not the feature should be part of the composite. This is because a system feature can only be removed from the composite if it has a realization rule that creates an inconsistency with the structural implications of the current composite.

The logicogrammatical stratum could also be expanded to have a larger and better organized lexicon. The current lexicon tends toward being a rather flat hierarchy. The
breadth of the hierarchy comes from the root of each word being a feature at the same hierarchical level, and stemming from the root are all its tense, voice, plurality, etc. variations. With each word added to the lexicon, the breadth at the root word level and the number of lexical entries will expand linearly. Even this linear expansion is too much and the need to incorporate a deeper lexicon will become more evident. This need is motivated by efficiency; when searching for the features representing a word, the search progresses through the set of features of each word. A feature is only ruled-out when the system encounters a realization rule that lexifies the feature to a word that does not exist in the data. When this occurs, the system must backtrack, which is a computationally expensive maneuver. It would be computationally beneficial to acquire more depth in the hierarchy, perhaps by identifying classes of words. This idea is similar to a characteristic of semantic grammars, where each verb is a member of a semantic category. By assigning verbs to semantic categories, ruling out a category rules out all the verbs of that category. “This means that for any given sentence only a small subset of the grammar will ever be applicable,” [Allen, 1987, page 252]. By implementing this in systemic grammar, the grammar's lexicon would have more depth, and ruling out a class would eliminate much of the search. Therefore, I feel the organizational strategy of the implemented lexicon used by SAPI, as well as any similarly designed systemic grammar, would benefit from identifying classes of words.

7.2.1.3 Additional Strata

It would be extremely interesting to expand the system to accommodate more levels or strata of the NLU task. The first step would be to have the syntactic parser interact with the semantic stratum, which already interacts with the pragmatic stratum. As features are
hypothesized at the syntactic level, pass them to the semantic interpreter to see what processing it can do. In turn, this information can be passed up to the pragmatic interpreter. In the meantime, any anticipatory top-down guidance from the pragmatic stratum could cause features to be passed to the semantic and on to the syntactic levels. Research has already been performed that determined it to be computationally beneficial to allow semantic and pragmatic expectations to guide the syntactic parsing [Skon, 1993].

Figure 72: Three layers and their interaction.

Figure 72 graphically depicts the three layers of information and their interactions. The various levels represent data spaces and the data observations made at one layer cause hypotheses to be abduced for them (as explanations) at the next. If the abduction in the hypothesis space encounters some generalization that can help guide the lower level, then that information is predicted to occur at the lower level. In this way, the layers can all synergistically guide the search at each other's level.
The next logical step is to accommodate more levels of processing. After including syntax in the system, the next layer to add would either require speech understanding or text recognition. Currently, these two issues have been completely avoided as this research's thrust and concentration has been at a higher level of natural language understanding. But to be a complete natural language understanding system, it would eventually have to accommodate either one of these issues.

If SAPI were to be used as a speech recognition system, layers representing the acoustic, auditory, phonetic, phonological-prosodic, and morpheme strata would need to be added to the syntactic, semantic and pragmatic layers [Josephson and Josephson, 1994]. The exact number and types of strata would depending on the implementation. In general, each stratum would abduce a conclusion to explain the input from the lower stratum with the aid of top-down guidance of the higher strata.

If SAPI were to be able to perform as a text recognition system, translating written or typed text into its corresponding pragmatic interpretation, SAPI would need several other layers of knowledge. It would have to translate the various pixel patterns into words. The pixel patterns could be the input to a letter recognition unit. This unit would abduce what the letter might be and pass this abduction to a word recognition unit. The words abduced could be the input to the syntactic parser. The top-down guidance to the word recognition layer would be in the form of words to look for based on syntactic correctness, semantic contribution, or contextual knowledge. The word recognition unit could guide the letter recognition unit by suggesting possible letters based on the partial completion of words.
7.2.1.4 Parallel Processing

The idea of parallel processing was introduced in Section 3.3.5 on page 64 which discussed a way to parallelize EFLI. Parallelizing SAPI to see the issues and benefits that would arise from parallelization would be interesting. A rough outline of one way to parallelize SAPI follows.

Let each layer (pragmatic, semantic and syntax) be assigned to its own processor. Let each processor run its corresponding piece of code shown in Figure 73. The basic idea of the code is that processing starts at each end layer (pragmatic and syntactic, or P and X). The pragmatic processor accepts as input all the anticipated features. It uses this information to construct its composite. When it has performed as much processing as possible, it sends a message to the semantic processor (S) of all the predicted features. The processor P now loops, either checking for additional input from S or working on creating its composite and sending those features to S.

While P processes its anticipated input, X receives the typed sentence and parses the portions it can with a high degree of confidence. These features are sent to S. The X processor now loops, performing a blocking wait for a message from S, which will be processed and a response sent.

S's construction of its own composite begins when it receives a message from P or X. It is a programmed server, receiving messages from P and X, and constructing replies according to the features hypothesized from the input. The relevant information added to S's composite that was not part of the incoming message is sent to the message sender. The third processor (not S and not the processor that was the source of the message) receives a
message from S of all the relevant information that was added to S's composite on this loop's iteration.

**Pragmatic Processor's Code**

P: get any anticipated features  
work on local knowledge  
send features to S  
while not done  
  if message from S then  
    utilize S's info if applicable  
  else  
    work on stratum to get features  
    send preselections of rules to S  
end-if  
end-while  
end-P.

**Semantic Processor's Code**

S: loop  
  if message from P then  
    utilize P's info  
    reply to P  
    send message to X  
  else if message from X  
    utilize X's info  
    reply to X  
    send message to P  
  end-if  
end-loop  
end-S.

**Syntactic Processor's Code**

X: get outside input  
do high confidence parsing  
send to S  
loop:  
  wait for S  
  process S's information  
send to S  
end-loop  
end-X.

Figure 73: Code to make the layers of processing occur in parallel.

Not represented in the parallel code is the handling of contradictory information. To handle such information, the processor with the contradiction could spawn two children processes, where one ignores the incoming information while the other accepts it and eliminates the features with which it contradicts. The children processes continue creating composites until either a complete, consistent composite is created, or the processor that
suggested the contradictory feature revokes its decision. Revoking a feature does not require backtracking, but only a killing of the children processes that reflect the inclusion of that feature.

This strategy has all the levels of processing progress simultaneously, sending messages to the processors that represent the neighboring level’s stratum. This processing continues until the pragmatic stratum has a complete and consistent composite that has adequate explanatory power to account for the input sentence.

7.3 Generalizing the Concepts

For the concepts and methods explored by this research to be of greater significance, they must be generalizable. This generalization can take many forms, such as generalizing to additional sentences, domains, or multiple NLU applications. Furthermore, the concepts ought to be globally applicable in some fashion, or at least have greater applicability than just to the domain of NLU. Generalization is often tied to scalability, which is the ability of a system to handle “more” of whatever it processes. In this case, it is more of natural language. Tied to the issue of scalability is the issue of computational complexity. For a system or concept to be able to scale up to handle its “more,” it must prove to be tractable, or at least manageably intractable.

The purpose of this section is to determine the computational complexity of SAPI. Having shown this, the scalability of the system will be addressed.
7.3.1 Computational Complexity

The computational complexity of abduction depends on the type of abduction performed. In abduction problems where the "best explanation" is defined to be the most plausible combination of hypotheses that explains all the data, abduction has been shown to be intractable (NP-hard) [Bylander et al., 1991]. In general, major factors that lead to intractability are "... choosing between incompatible hypotheses, reasoning about cancellation effects among hypotheses, and satisfying the maximum plausibility requirement..." [Josephson and Josephson, 1994].

In the description of the abduction strategies used in the semantic interpreter and the pragmatic interpreter, it would be useful to adopt some notational conventions and definitions. The ones described below are a subset of those identified in Bylander et al. (1991) and Josephson and Josephson (1994).

\(d\) stands for a datum

\(D_{all}\) is a finite set of all the data to be explained

\(h\) stands for an individual hypothesis

\(H\) stands for a set of individual hypotheses

\(H_{all}\) is a finite set of all the individual hypotheses

\(e\) is the map from subsets of \(H_{all}\) to subsets of \(D_{all}\) (\(H\) explains \(e(H)\))

\(I\) is a set of two-element subsets of \(H_{all}\), indicating pairs of \(h\) that are incompatible with each other

\(H\) is complete if \(e(H) = D_{all}\)

\(H\) is parsimonious if \(\not\exists H' \subset H (e(H) \subseteq e(H'))\)

\(H\) is an explanation if it is complete and parsimonious
The discussions in the next two subsections will use these terms and notation in their discussion of the complexity of the semantic and pragmatic interpreters.

### 7.3.1.1 Semantic Interpreter

Several classes of abduction problems exist, one of which is called the *independent incompatibility abduction problem* class. An independent incompatibility abduction problem satisfies the formula:

\[
\forall H \subseteq H_{all} \left( (\neg \exists i \in I \ (i \subseteq H)) \Rightarrow e(H) = \bigcup_{h \in H} e(h) \right) \tag{Eqn 1}
\]

By definition, all abduction problems that are a member of the class of independent problems are also members of the *monotonic* class. A monotonic abduction problem satisfies the formula:

\[
\forall H, H' \subseteq H_{all} \ (H \subseteq H' \Rightarrow e(H) \subseteq e(H')) \tag{Eqn 2}
\]

The abduction problem of semantic interpretation as solved by SAPI is a member of the class of independent incompatibility abduction problems. The rationale for placing it in this class is that the logicogrammatical stratum contains incompatibilities (the children of a system are incompatible), the sum of the explanation of the members of the composite is equal to the composite’s explanation, and any subset of the composite will explain a subset of what the composite explains.

As proven by Bylander et al. (1991), for the class of independent incompatibility abduction problems, finding a best explanation is NP-Hard. In this case, the term “best” implies the explanation with the greatest plausibility. SAPI manages to avoid this constraint since its definition of best is the first explanation found.
Finding the first explanation does not rescue the semantic interpreter from all problems. Another theorem in Bylander et al. (1991) states that for the class of independent incompatibility abduction problems, it is NP-complete to determine whether an explanation exists. Hence, as a worst case, finding an explanation with the semantic interpreter would require creating and evaluating all combinations of the knowledge represented in the system. Though this is prohibitive, the computational complexity of the type of abduction problem should not be the sole factor in determining its problem-solving efficiency. Sometimes it is beneficial to resort to "tricks" to make the processing manageable, even though it is intractable.

The semantic interpreter can utilize several manageability tricks to cope with its intractability. These tricks include:

- Allowing the pragmatic stratum to halt the processing in the logico-grammatical stratum. This halting can take place when the pragmatic stratum has determined "enough" pragmatic features to account for the highly-likely and likely features of the logico-grammatical stratum. This determination can be made by using the hypothesized pragmatic features and trying to generate the sentence represented in the input. What makes this method possible is that the pragmatic stratum does not have to account for all the logico-grammatical features (as will be discussed in Section 7.3.1.2 on page 201).

- Allowing the pragmatic stratum to decide at which hierarchy of the logico-grammatical stratum the processing should continue. This can be used to direct the system to the important, or previously unknown, information and avoid wasting time on the redundant portions of natural language.

- Shielding the semantic interpreter from trying to account for ill-formed sentences (syntactically incorrect). The syntactic interpreter must be able to parse the sentence to create input for the semantic interpreter. This guarantees that the syntactic input to SAPI represents well-formed sentences.
• Halting the processing if two incompatible highly-likely hypotheses are placed in the composite and they are both from the same source (both from the pragmatic or syntactic input). In this case, the problem does not lie in the semantic interpreter or the logicogrammatical stratum, but a flaw exists in the source that provided the information that caused two mutually exclusive features to be highly-likely.

• Backtracking in either a dependency-directed or chronological manner if two incompatible likely hypotheses are placed in the composite. Another alternative is to remove the hypothesis that is determined least likely of the two by the pragmatic stratum.

• Having a "nonsense" hypothesis that can account for any input. After a predetermined time interval, add it to the composite. This hypothesis causes the system to ask the "speaker" to repeat the statement, or add more information. If the same syntactic parse is received, the semantic interpreter will continue processing the input a little longer. This strategy can be repeated a number of times within some limit. When the limit is met, additional information from the speaker is mandatory.

The first two tricks relax the definition of complete, no longer will the semantic interpreter create an $H$ such that $e(H) = D_{all}$. This is acceptable since the ultimate goal is to create a pragmatic interpretation. The third and fourth tricks are used to make sure that the semantic interpreter is still efficient in the face of bogus input. The fifth trick enables the semantic interpreter to efficiently cope with conflicting information confined to the logicogrammatical level. And the sixth trick is used if processing has taken "too long." The semantic interpreter can check to make sure that it received the correct information. These tricks can be used to make the semantic interpretation manageable.

7.3.1.2 Pragmatic Interpreter

The abduction problem of pragmatic interpretation also falls within the class of independent incompatibility abduction problems. One big difference that exists between the pragmatic interpreter and the semantic interpreter is that while the semantic interpreter explains $D_{all}$, the pragmatic interpreter does not $(e(h) \subseteq D_{all})$. This difference is significant;
it is the reason that the pragmatic interpreter can solve its problem in polynomial time.

Figure 74 is the algorithm used by the pragmatic interpreter as discussed in Section 4.5 on page 88.

\[
W \text{ represents the working composite, or accounts-for group}
\]
\[
C \text{ is the composite that will explain } D_{all} (e(C) = D_{all})
\]

\[
W = C = \{ \}
\]

For each \( h \in H_{all} \)

if \( e(h) \subseteq D_{all} \) then

\[
W \leftarrow W \cup \{ h \}
\]

Return \( W \)

For each \( h \in W \)

if \( e(h) \sim \subseteq \bigcup e(h') \) where \( h' \in W \setminus \{ h \} \) then

\[
W = W \setminus \{ h \}
\]

\[
C = C \cup \{ h \}
\]

\[
D_{all} = D_{all} \setminus e(h)
\]

For each \( h' \) that is an ancestor of \( h \)

\[
W = W \setminus \{ h' \}
\]

\[
C = C \cup \{ h' \}
\]

\[
D_{all} = D_{all} \setminus e(h')
\]

For each \( h' \in W \)

if \( \{ h, h' \} \subseteq I \) then

\[
W = W \setminus \{ h' \}
\]

if \( e(h') \sim \subseteq D_{all} \) then

\[
W = W \setminus \{ h' \}
\]

Return \( C \)

Figure 74: Polynomial algorithm for pragmatic interpretation.
The algorithm of the pragmatic interpreter is polynomial. The proof of this can be shown by letting $|D_{all}| = k \geq 1$, $|H_{all}| = m \geq 1$, $|I| = p \geq 1$, $n = k + m + p$, and the time to evaluate $e$ is $O(C_e)$ where $|C_e| \geq 2$. Assume that sets are represented as bit vectors. Assigning $W$ and $C$ to the empty set takes $O(1)$. Since $|H_{all}| = m$, the first loop iterates $m$ times. The loop calls $e$ ($O(2C_e)$), verifies that its answer is in $D_{all}$ ($O(k)$), and performs up to $m$ set additions ($O(1)$). This results in the complexity of the first loop being $O(m (C_e + k + 1))$.

The second loop is performed at the most $m$ times. The first part (before the first embedded loop) evaluates $e$ once for each member of $W$ ($O(C_e + mC_e)$), performs two set differences, the first of complexity ($O(m)$) and the second of ($O(k)$), and one set addition ($O(m)$). This portion simplifies to $O(mC_e + k)$. The first embedded loop executes $m$ times, performs two set differences, the first of complexity ($O(m)$) and the second of ($O(k)$), and one set addition ($O(m)$). Thus its complexity is $O(m^2 + k)$. The second embedded loop executes $m$ times, performs a set comparison ($O(p)$), one evaluation of $e$ with a comparison to $D_{all}$ ($O(C_e + k)$), and two set differences ($O(m)$). Thus its complexity is $O(m (p + k + C_e + m))$. Therefore, the total complexity for the entire second loop is $O(m^2 C_e + m^2 + m^2 k + m^2 p)$. Since $n$ is greater than $m$, $k$, or $p$, $O(n^2 C_e + n^3)$ clearly bounds the complexity. This is polynomial time.

Though the algorithm represents a way to perform pragmatic interpretation, it is not the only algorithm that exists. But since this one is polynomial, then the problem is polynomial and tractable.

Having now discussed the computational complexity of the components of SAPI, it is fair to discuss its scalability.
7.3.2 Scalability

The future work has been discussed in terms of how the grammar could be expanded and altered and how the system could incorporate more levels and be parallelized. Additional future work exists in another dimension, that of using the ideas presented and expanding them. This future work shows that the research can be generalized at many different levels of abstraction.

7.3.3 Increased Coverage, Same Domain

The system currently understands eleven sentences, however, there is no upper bound limit to the number of sentences that it could understand. Making it understand more sentences should simply require adding more lexical features to the grammar, and perhaps fine-tuning the existing features. As discussed earlier, the efficiency of the system from adding words to accommodate additional sentences may depend greatly on the organization of the addition. In other words, the system should scale up quite nicely to accommodate additional sentences. These additions should not require any changes to SAPI as the grammar was created as its own module.

7.3.4 Additional Domains

The current system operates in the SERVICE domain, and in particular the service of a travel agent. SAPI is not limited to this domain as it was developed independently of any grammar or domain. Changing the domain would require changing the pragmatic stratum to a new domain, and making certain that the lexicon in the logicogrammatical stratum has the words that will be used in the new domain.
7.3.5 Broader Applicability

As defined by Ralph Grishman (1986), a primary motivation for computational linguistics has been the "...development of specific practical systems which involve natural language."

Though the objectives of the research in the field have varied greatly, there have been three classes of applications identified which have been central in the development of systems. These classes are: machine translation, information retrieval, and man-machine interfaces. For this research to have true substance it is important to point out how it can contribute to each of these three areas.

7.3.5.1 Machine Translation

The work presented in this system has direct application to machine translation. Two approaches to machine translation are "direct translation" and "transfer approach." Direct translation systems perform manipulations on the source sentence to directly produce the target sentence. This type of system has been found to be inadequate. Translation by transfer performs a sophisticated analysis of the structure of the source sentence to produce an explicit intermediate representation of its structure. This representation has "transfer rules" applied to it. Transfer rules are rules that are specifically designed to translate specific structures from a source language into the corresponding specific structures of the target language. Neither of the two approaches involve semantic analysis so they are both prone to semantic ambiguities. The transfer technique generally requires post-editing by a human translator to correct any semantic mistakes [Goodman and Nirenburg, 1991]. Transfer systems that do not require post-editing have been constructed for very limited domains.
A third approach to machine translation is the “interlingua” approach. An interlingua is an intermediate language that encodes the significance of the sentence. Several notations have been suggested as interlinguas, but the determination of which should be used is immaterial, as long as the meaning is actually extracted. The sophisticated structural analysis of the source sentence is still required, as it was in the transfer approach. The semantic analysis determines the literal meaning, and from this the significance can be extracted and encoded in the interlingua. Once this encoding occurs, the intended meaning can be generated in the target language. By utilizing the pragmatic stratum as the interlingua, the system is concerned with sociolinguistic issues, as well as “who did what to whom.” This interlingua is sensitive to the social aspects of language such as jargon, formality and politeness. This information is represented by the mapping from the pragmatic stratum to the logicogrammatical stratum.

SAPI can be used as a portion of a machine translation system as shown in Figure 75. The shaded area represents the portion of the system that is computed by SAPI. This is a model of an actual prototype system discussed in Patten and Hartigan (1994).
Figure 75: A proposed machine translation system.
7.3.5.2 Information Retrieval

Much of the information people acquire and use today is in natural language form. This has resulted in a great interest in performing automatic information retrieval from natural language texts. Ideally, the user will be able to enter a query and the computer will retrieve only those texts relevant to the user's search. Many obstacles confront this process, independent of the complexity of the fields being researched. Three problems that arise are caused by synonyms, homonyms, and specificity. Because of synonyms, a text may be indexed by one word, while the user searches with a synonymous one. Synonyms cause the user to not retrieve all the relevant information; by searching under word A, the user misses all the articles stored under word B. In the case of homonyms, the user retrieves irrelevant information. For example, a search under the keyword of "depression" will retrieve articles concerned with The Great Depression, being depressed, and depressions in the ground. To further complicate the issue, the user may choose a keyword that is too broad or too specific, resulting in a proliferation or deprivation of material.

How can this research help compensate for these problems and help the application of information retrieval? If the context of the keyword is known, then the system can help determine the various synonyms under which to search and identify the areas that are not of interest. Perhaps the database of the keywords of articles can include a tag that specifies the field of concern of the article. For example, for depression, the articles would be tagged with "history," "psychology," and "geology" (to identify the three meanings given in the paragraph above). In this way, the correct field of the word can be searched. The idea is that SAPI could be used to help identify the appropriate keywords under which to search for texts by identifying the context in which the user is interested.
7.3.5.3 Man-Machine Interfaces

People communicate with each other in a natural language all the time. Many people find it frustrating and intimidating to try to communicate with the computer in the computer's language. Therefore, it would seem more non-threatening to non-computer users if they could communicate with the computer with the same language they use with people. The research presented here can be seen as a part of a system to accomplish such a man-machine interface. The corpus of sentences represented a conversation between a travel agent and a prospective traveler. Replacing the travel agent with a system that understands the typed requests of the traveller and that can interact with a database of flight scheduling information, the system could perform as a man-machine interface. In a way, it would also be performing information retrieval as it would have to understand the traveller's statements and queries and retrieve the relevant information from the database.

To make the man-machine interface even more acceptable, it would need to perform speech recognition. It has already been discussed that SAPI can be part of a speech recognition system.

7.3.6 Computational Theory

The final area of scalability is really more of a matter of generalization. The system has been discussed in terms of understanding more sentences, understanding other domains, and use in the various applications of computational linguistics. But what has the system suggested in general about processing in any domain, for any problem? What can be gained by applying its task structure to other problems?
This system utilizes and enables analysis of some useful ideas and methods that may benefit other systems that perform cognitive tasks. This is not to say that the approach here is cognitively plausible, but that the method used to make the processing of this system efficient can be applied to other cognitive tasks, such as perception.

What SAPI demonstrated that can be used by other systems is the computational effects of incorporating context as part of the overall processing. SAPI used context in two separate ways, as anticipatory and as run-time top-down guidance. Anticipatory guidance generally caused a speedup in processing. For a system that performs a perception task, taking advantage of anticipatory top-down guidance is not an ad hoc computational trick, but a wise incorporation of available knowledge. This knowledge can not be foolproof though, so the system must be have a method of recovery. In SAPI, this was manifested in its meta-knowledge.

SAPI also showed that run-time guidance may be beneficial. As more features or data became known, the context becomes more decided and other predictions can be made. This can guide the interpretation of the data so the processing does not have to be completely from scratch.

The general emphasis of SAPI was that context brought to bear before processing begins in the form of anticipations can nearly always cause the interpretation process to be more efficient. Usually context used as run-time guidance is helpful. The system should not spend extra time trying to determine or better define the context, as it can sometimes slow down the overall processing time. Context which is readily available or determinable should be used.
7.4 NLU and Abduction

Noam Chomsky made a claim during a presentation at The Ohio State University that natural language understanding is not abductive since abduction is too deliberative. In this section, I will present evidence that Chomsky was mistaken, and not only is NLU abductive, but his own Innateness Hypothesis fits the non-deliberative abductive method.

Chomsky believes that the “general structure of human language is fixed biologically, and the human language learner acquires those aspects of the language that differentiate it from other languages,” [Akmajian et al., 1984, page 32]. This statement suggests that a child is born with a Language Acquisition Device which has built in knowledge of the possible types of rules and structures that are permitted in natural language. This knowledge will limit the “search space” of language learning by eliminating the infinite number of impossible rules of language which in turn will facilitate rapid learning. These preprogrammed rules are the rules of Universal Grammar. This theory of the linguistic universals being innate is called the Innateness Hypothesis [Chomsky, 1971]. The Innateness Hypothesis elucidates the rapidity of language acquisition and can help account for the fact that language is learned by children without explicit instruction, that children acquire infinite generative ability in finite time, and that children receive different inputs and have different intelligences and yet develop a similar language competency level.

Furthermore, the innateness of language can help account for the rapidity at which adults process it. With the universal principles being “hard-wired”, and the acquired rules eventually being highly compiled, real-time processing of language can occur. Also, if language processing has its own processing module in the brain, then that module can be
designed specifically for the processing of language, rather than having language processed by a module of general intelligence.

7.4.1 Rebuttal

Having previously defined abduction and briefly stated Chomsky's view of language, Chomsky's belief that abduction is too deliberative and too explicit a method to be of use during language understanding is comprehensible. People interpret what is said to them with such speed that there cannot possibly be time for each level of language understanding to consciously consider all the hypotheses to explain the data, form competing composites, and then score these composites to find which has the greatest plausibility. To say that abduction occurs during this process by the listener and to propose that it happens in real-time seems preposterous. Furthermore, often people know what is being said to them even before the speaker has finished the utterance. This would imply that people are abducting with an incomplete data set. In light of this, how can abduction possibly be used in the task of language understanding?

Chomsky has a strong claim, and correct one, if the deliberative and explicit abductive methods discussed previously are the only methods. Deliberative and explicit tasks, such as theory formation and diagnosis, fit naturally into the description of abduction and the methodology that abduction employs. These are tasks in which a theoretician or doctor can verbalize their thought process, how they evaluate the various alternatives, and why they settled on their conclusion. Tasks that we can not explain how to solve, those that we can not give step-by-step explanations of the decision making, are more difficult to conclusively show to use any specific methodology. NLU is such a task.
I agree with the claim that language understanding is indeed abductive in nature. By definition, abduction is inference to the best explanation. Isn’t that what we are doing when we process language? NLU is specified by a set of data, such as acoustic data, and the goal is to process it into a description that captures the semantics and pragmatics of the utterance. In the face of multiple hypotheses explaining the same data, the best one is chosen as part of the composite explanation. Simply stated, in language understanding, we infer to the best semantic and pragmatic explanation what the acoustic data represents. At the computational theory level, natural language can be described as:

The purpose of human language is presumably to transform a data structure that is not inherently one-dimensional into one-dimensional form for transmission as a sequential utterance, thereafter to be retranslated into some rough copy of the original in the head of the listener. [Marr, 1977, page 138]

There are many representations and algorithms that can realize this mapping. Abduction is just one. And abduction can be realized by multiple implementations, each differing in degree of deliberation and explicitness (as seen in Chapter III). I don’t believe that there has to be one abductive method that accounts for all tasks at the various levels of deliberation and explicitness. Some abductive methods may be better for the perceptive and implicit abductive tasks, while others are good for the deliberative and explicit abductive tasks. Which to use depends on our time constraints, the urgency of a conclusion, the benefit of being right and the cost of being wrong, as well as the ease and naturalness of use.

Since NLU cannot be a deliberative, explicit abductive task, yet it is abductive, it must be an example of a perceptive, implicit abductive task. In NLU, we hypothesize and account for the data but we do it swiftly by reasoning over highly-compiled knowledge and processing in parallel. Knowledge compilation means that the various hypotheses and their interactions can be prestored and need not be searched for at run-time.
How can this compilation occur? It may be evolutionary with the compilation being in the hardware. This corresponds to Chomsky's Language Acquisition Device. Evolution created a special purpose device in the brain to enable language processing to happen swiftly. Compilation can occur from learning. This is the selection or refinement of the rules of the Universal Grammar (which is compiled knowledge through evolution) into the rules of the Particular Grammar. Lastly, compilation can occur incrementally as needed at runtime. This form of compilation accounts for the speed of processing information that was not compiled by either of the other two means. The imperative need for compilation is to avoid computationally expensive run-time search. NLU operates by abducing over highly-compiled knowledge.

SAPI, though not the panacea of NLU systems, is a step in the right direction. SAPI is a deliberative system that reduces the explicitness of the task. The explicitness is reduced by compiling the effect that context has on language use. SAPI performs a best-first search with the aid of top-down guidance to create the composite. This top-down guidance is in the form of preselections from the pragmatic stratum to the semantic stratum. SAPI does not explicitly compare alternative composites because it takes advantage of this tight coupling of the strata. One pragmatic feature preselects multiple logicogrammatical features. The semantic interpreter has its explicit nature lessened by using a classification hierarchy, ordering the consideration of hypotheses, and compiling the preselections developed according to context. Therefore, it is possible to propose an abductive strategy where the consideration of hypotheses is not completely explicit.

I believe that Chomsky's mistake was that he overlooked the idea of implicit abduction, non-deliberative abductive methods, the availability and use of highly-compiled
knowledge, and the wealth of knowledge called context. NLU is abductive, we have just not yet found a way to have the knowledge highly-compiled to the same degree as humans or compiled into special mechanisms.

7.5 The Role of Context in NLU and Abduction

It has been presented that context generally helps to decrease processing time. This section will discuss other benefits of the use of context in NLU as well as how the various senses create an "abductive hierarchy of cognition."

Contextual guidance is an invaluable tool in NLU. Besides the purpose of efficiency, context imposes biases and helps with identification and disambiguation. In SAPI, the pragmatic stratum, with its sociopragmatic features, encoded the context. The result is that:

...sociopragmatics can make the sentence understanding faster. Sociopragmatics depends on context: social, linguistic, physical. As a result context contains enough information to make some good predictions about what speakers are going to say and how they are going to say it. These predictions can then be used to help find the particular grammatical constructions in the sentence, as finding something is easier if you know what you are looking for. [Patten and Hartigan, 1994]

Context was used to provide expectations. Utilizing expectations to guide the processing is really an exploitation of best-first search. This technique enables the system to consider the most plausible hypotheses first, generally making the time needed to find a possible solution shorter. In language understanding, the sociopragmatic expectations tell the language analyzer which possibilities to look at first, and thus guide the language understanding process.
When people believe that they are in a certain context, they have expectations of what is to be said, not just in terms of word choices but also in terms of topics and sentence structure. When little Billy breaks a vase, and Mom, hearing the shattering of lead crystal, comes running into the room, you can bet Billy is not expecting a hug! The context of the situation has Billy expecting disapproval of some sort. The sentence structure is likely to be short and to the point. Billy knows this and expects it, so when Mother starts yelling, he is not abducing what she is saying from scratch, he already knows what to expect.

Now, imagine that Grandma runs into the room instead. Billy may expect a different reaction from Grandma since he and Grandma have a different relationship than he and his mother. Grandma might be more relaxed about these things, since her darling little angel of a grandson obviously did not intend to have this happen and therefore should not be punished.

Having stated that NLU is an abductive task, how can contextual information greatly reduce the immensity of this abductive task? The abductive task for language understanding on the surface seems to be an impossible task since the range of abductive hypotheses is immense. I would like to claim that abduction is aided by context. For example, imagine a noisy setting that makes definite comprehension difficult, such as a bar, a concert, or a pep rally. Suppose I were telling you a story and you knew I was a singer. In this story, you heard what you thought was the phonetic sequence /sIksstonz/. In the mapping from these phonemes to words\(^1\), the abduction is aided by context and you would hypothesize that I had said “six tones.” If I were a rock collector, there would have been a greater probability that I was talking about “six stones.” Furthermore, if I were reading a

\[1\] There are several layers of mappings that occur between these two levels, but for the purposes of this paper, it is sufficient to make the example at these levels.
children's fairy tale, then I might have said "sick stones." These examples show that context can reduce the number of possible hypotheses at the time of their formation.

Another example of how abducing what is heard is dependent on context can be seen in the experiments conducted by D. J. Bruce. Bruce (1958) performed an experiment, where he told the participants that the sentence they were about to hear was going to be about a certain topic. Then the participants were presented with "noisy" sentences that may or may not have been about the aforementioned topic. Many of the participants, since they were given a context to be listening for, thought they heard words and phrases related to the topic. According to Bruce, this shows "...the major role anticipation plays in the perception of speech," [Bruce, 1958, page 94]. This anticipation reduces the number of word and phrase hypotheses to consider. As Bruce discovered:

For speech perception, the recognition value of the immediate sign seems to be insignificant compared to the information coming from the contextual sources. The particular interest of these tests is that the critical source lay outside the messages themselves. The syntactical and associative structure of the latter remained constant, but the situation, as established by a given keyword, changed. With consequent change in set went drastic alterations in the intelligibility of word signs. "Being in certain familiar surroundings" was one of the essential attributes for recognition to occur. [Bruce, 1958, page 96]

An interesting aspect that is overlooked in the interpretation of natural language is the use of body language. A person's stance or facial expression can dictate what type of statement is about to be made. Returning for a moment to the story about Billy, if Mom enters the room and has her finger pointing in the general direction of his bedroom, he has greater expectation that he is about to receive an imperative statement such as "Go to your room!" While if Mom rushes into the room and has a disastrous look of shock on her face, Billy can expect to receive an interrogative statement such as "What happened?"
Furthermore, facial expressions in some languages play a larger role than they do in English. For example, in American Sign Language, there are many facial expressions used to express the meaning of the sentence. The sentence “The dog chased the cat” would be said with a “straight” face. If the sentence were to be topicalized, “The cat, the dog chased,” then the facial expression of “raised eyebrows” would be needed to convey the message. Without the facial expression, the sentence would be interpreted as “The cat chased the dog.” This is a case where a single mode of communication (hand signs) is not enough to make the language interpretable.

To include other factors, such as information provided by other modalities, suggests that there exists an “abductive hierarchy of cognition.” This presupposes that each modality performs its task abductively, which other researchers have already claimed for vision [Marr, 1982] and speech recognition [Fox, 1992; Josephson and Josephson, 1994]. This abduction hierarchy of cognition operates by letting each modality interpret the modality-specific data it receives. As it processes, it can ask for “help” from the other modalities. This help can be in many forms, for example it might be that the modality is trying to evaluate a hypothesis and the scoring depends on the hypotheses of another modality. Or one modality may have scored a hypothesis with high confidence and it causes a rise or a fall in the score of a hypothesis of another modality. Or perhaps two hypotheses are being considered and another modality has a hypothesis that can make the decision between the two. The modalities all work together, creating partial explanations and letting these partial explanations “mingle,” causing further hypothesis evocation and re-evaluation of existing hypotheses.
The above examples argue that abduction can operate simultaneously on many abstraction levels on different data input streams. The number of hypotheses considered at one time can be limited by the expectations derived from the context of the situation as well as inter-modality relationships, thereby increasing the efficiency of the abductive task. When this is coupled with the possibility of parallelizing the task and using compiled knowledge, even greater efficiency can be achieved since the already reduced number of hypotheses can be considered and evaluated simultaneously.

7.6 Conclusion

The final conclusion is using context is a valuable way to reduce processing time. Both the system and the research discussed within the bounds of this dissertation have presented interesting and novel ideas. SAPI is a computational model of a natural language understanding system that performs semantic and pragmatic parsing, utilizes a bidirectional systemic grammar, is sensitive to sociopragmatic issues, interprets sentences by appealing to a layered abductive strategy, and can cope with incomplete input. The use of context to guide the interpretation was explored, both in the form of anticipatory top-down guidance and as run-time top-down guidance, to aid in hypothesis evocation and instantiation. This research investigated the issues of when context is the most beneficial, what may be done when information sources conflict or are incomplete, and the potential realms of future use for extrapolated concepts. Context is everywhere and contextual knowledge often plays a decisive role in NLU, but not always. It is this ambiguous applicability that confounds researchers, creates wonderment and intrigue to many, drives research programs, and lays the groundwork for dissertations.
Example of the Pragmatic Stratum

A.1 Introduction

To better comprehend the organization, representation and type of information represented in a stratum, it may be useful to view an actual stratum. The vastness of the logicogrammatical stratum makes it difficult to display and discuss so the pragmatic stratum will be used as the example.

The original pragmatic stratum was created through a collaboration of efforts of Patten, Geis and Becker and is described in their paper [Patten et al., 1992]. The cryptic language used to describe a stratum (that is shown in this Appendix) is translated into Ops5 code which is used as input to SAPI. The translation mechanism was created by Patten for his work described in [Patten, 1988].

The actual pragmatic stratum used in this work for both top-down guidance and pragmatic interpretation is a version of the one discussed in [Patten et al., 1992]. Before displaying the pragmatic stratum “code,” it is necessary to provide an explanation of the various symbols. Following the explanation is an example of a portion of the pragmatic stratum and its corresponding system network. This appendix concludes with the actual pragmatic stratum code.
A.2 Explanation of Symbols

In the code for the pragmatic stratum, several symbols appear. The symbols can be subdivided into those that are used to describe the structure of the hierarchy (systems and branching symbols) and those that describe the expected logicogrammatical features associated with each pragmatic feature (realization rules). Each of the various symbols will be explained in this section.

A.2.1 Systems and Branching Symbols

Each system represents a group of related features. A system is represented by an “S_” prefixed to a term that describes the system. For example, S_FORMALITY is the system leading into the formality system that includes the features “informal” and “formal.”

Several different branching symbols are used in the notation, each with its own meaning. In each case, the parent(s) would be those feature(s) listed to the left of the symbol and the child(ren) would be those feature(s) listed on the right side. The symbols are:

- [ which represents an OR branch. The feature on the left side of the symbol is the parent of all the features on the right side. Only one of the children can be part of the composite if the parent is part of the composite.

- ]- represents as OR join. All of the features on the left of the symbol are the parents to the feature on the right. Any of the parents being in the composite will make it necessary for the child to be part of the composite.

- {= represents an AND branch. The feature on the left side of the symbol is the parent of all the features on the right side. All of the children will be part of the composite if the parent is part of the composite.

- =}- represents an AND join. All of the features on the left of the symbol are the parents to the feature on the right. All of the parents must be part of the composite in order for the child to be part of the composite.
A.2.2 Realization Rule Symbols

With a system or gate feature may be realization rules. Realization rules specify the logico-grammatical implications of the selected pragmatic feature. The symbols used in the realization rules of the pragmatic stratum include:

- "U" represents the entire input string.
- The symbol "^" represents the adjacency realization rule. The items listed on each side of the symbol will appear consecutively in the order that they appear in the rule.
- The "$" symbol represents the beginning of the sentence if it is left of the "^" or the end of the sentence if it is right of the "^".
- The symbol "@" represents the rule of preselection. The role to the left of the symbol preselects the feature to the right of the symbol.
- The symbol "<" helps specify a path of roles. For example, U<Spatial<-Range@non-possessive-nom specifies that the input string's Spatial role's Range will have the feature of non-possessive-nom in the logico-grammatical stratum.

A.3 Example System Network

Having explained the various symbols, it may be easiest to understand the code of the pragmatic stratum if a small portion of it is portrayed graphically as a system network. Therefore, below is a section of the code from the pragmatic stratum with its corresponding system network. In the system network, the realization rules are in the shaded boxes. Generally in a system network, the names of the systems are omitted, but for readability, the system names have been included.
Figure 76: System network sample for a portion of the pragmatic stratum.
A.4 Example Code

Below is the actual pragmatic stratum code used to provide top-down guidance, to justify the results of the semantic interpreter, and to create the pragmatic interpretation.

```
(setq grammar '(
  (nil service ($ ^ U) (U ^ $))

    ((service -{=} S_SERVICE-TYPE)

      ((S_SERVICE-TYPE -{}) booking-agent)

      (booking-agent -{}) travel-agent)

    (service -{=} S_FORMALITY)

    ((S_FORMALITY -{}) formal
      (U<Process @ !formal)))

    ((S_FORMALITY -{}) informal
      (U<Process @ !informal)))

    ((service -{=} S_ORIENTATION)

      ((S_ORIENTATION -{}) agent-oriented
        (U @ speaker-subject))

      ((S_ORIENTATION -{}) client-oriented
        (U @ addressee-subject))

      ((client-oriented -{}) client-singular
        (U<Subject @ singular))

      ((client-oriented -{}) client-plural
        (U<Subject @ plural))

      ((travel-agent -{}) need-determination
        (U @ positive))

      ((need-determination -{}) general-need-det
        (U = I may I help you!))
```
((need-determination -[] specific-need-det)

((specific-need-det -{=} S_NEED-TYPE)

((S_NEED-TYPE -[] non-negotiable)

((S_NEED-TYPE -[] negotiable)

((travel-agent -[] need-satisfaction)

((negotiable -[] flying-times
  (U @ operative)
  (U @ material))

((flying-times -[] departure-info
  (U<Process @ !296vb))

((departure-info original-question =}-) original-dep-question
  (U @ no-residual))

((original-dep-question -[] departure-date-known
  (U<Wh @ bare-np)
  (U<Wh<Range<Head @ !108n)
  (U<Wh<Range @ mass)
  (U<Wh<Range @ determined)
  (U<Wh<Range<Deictic @ !what)
  (U<Wh<Range<Deictic @ !wh-det)
  (U<Wh<Range @ noun-head))

((original-dep-question -[] departure-date-unknown
  (U<Wh @ bare-np)
  (U<Wh<Range<Head @ !question-adverb)
  (U<Wh<Range<Head @ !108adj)
  (U<Wh<Range @ non-superlative-adj-head))

((flying-times -[] return-info
  (U @ place)
  (U @ residual)
  (U<Process @ !bring)
  (U<Spatial<Range<Head @ !286adj)
  (U<Spatial<Range @ non-possessive-nom)
  (U<Spatial<Range @ non-relativized)
  (U<Spatial<Range @ non-qualified)
  (U<Spatial<Range @ unmarked-adj-head))
(U<Spatial @ bare-np)
(U<Wh @ bare-np)
(U<Wh<Range @ question-adv-head)
(U<Wh<Range<Head @ !108adj))

((non-negotiable -l) seat-quantity
 (U @ wh-adjunct-head)
 (U @ no-residual)
 (U @ extent)
 (U @ perception)
 (U @ present)
 (U @ present-in)
 (U @ range-operative)
 (U<Process @ !459vb)
 (U<Phenomenon<Minor-process @ !prep-goal))

((specific-need-det -{=} S_NEED-DET-ORIG/FOLLOWUP)

((S_NEED-DET-ORIG/FOLLOWUP -l) original-question
 (U @ unmarked-wh-theme))

((S_NEED-DET-ORIG/FOLLOWUP -l) followup
 (U @ no-residual))

((followup flying-times =}-) time-followup
 (U @ time-adjunct)
 (U<Temporal<Head @ !comparative-adjunct))

((time-followup -l) question-time-backward
 (U<Temporal<Head @ !135adj))
 (U<Temporal @ non-pre-modified-ap)
 (U<Temporal @ non-post-modified-ap))

((time-followup -l) question-time-forward
 (U<Temporal<Head @ !136adj))

((need-satisfaction -l) search)

((search -{=} S_SEARCH-SUCCESS)

((S_SEARCH-SUCCESS -l) success-so-far
 (U @ positive))

((S_SEARCH-SUCCESS -l) unsuccessful-so-far
(U @ negative))

((S_SEARCH-SUCCESS |-) incomplete-success
 (U @ positive)
 (U @ textual-theme)
 (U<Conjunct @ !responsive/explanative))

((search |-) S_SEARCH-PROGRESSION)

((S_SEARCH-PROGRESSION |-) search-about-to-begin)

((S_SEARCH-PROGRESSION |-) search-in-progress
 (U @ present)
 (U @ residual)
 (U @ unmarked-declarative-theme))

((incomplete-success search-in-progress =|-) S_SEARCH-ORIENTATION)

((S_SEARCH-ORIENTATION |-) system-responsibility
 (U<Carrier<Beta<Process @ !443vb))

((S_SEARCH-ORIENTATION |-) agent-responsibility
 (U<Carrier<Beta<Process @ !484vb))

((S_SEARCH-PROGRESSION |-) search-just-completed
 (U @ residual)
 (U @ unmarked-declarative-theme)
 (U @ place))

((search-just-completed =|) S_FLIGHT-MENTION)

((S_FLIGHT-MENTION |-) flight-mentioned-previously
 (U<Spatial<Range<Deictic @ !far))

((S_FLIGHT-MENTION |-) flight-unmentioned-previously
 (U<Spatial<Range<Deictic @ !non-add))

((search-just-completed =|-) S_AIRLINE-MENTION)

((S_AIRLINE-MENTION |-) mention-airline
 (U<Spatial<Range @ qualified)
 (U<Spatial<Range<Qualifier @ unmarked-prep-phrase)
 (U<Spatial<Range<Qualifier<Range @ non-determined)
 (U<Spatial<Range<Qualifier<Range @ non-possessive-nom)
((U<Spatial<Range<Qualifier<Range @ non-relativized)
(U<Spatial<Range<Qualifier<Range @ non-quantified)
(U<Spatial<Range<Qualifier<Range @ non-classified)
(U<Spatial<Range<Qualifier<Range @ noun-head)
(U<Spatial<Range<Qualifier<Minor-process @ !prep-positive)
(U<Spatial<Range<Qualifier<Minor-process @ !prep-position)
(U<Spatial<Range<Qualifier<Minor-process @ !line/surface))

((S_AIRLINE-MENTION -[]) dont-mention-airline
 (U<Spatial<Range @ non-qualified))

((need-satisfaction -[]) report
 (U @ unmarked-declarative-theme)
 (U @ future)
 (U @ singular-subject)
 (U @ nominal-subject)
 (U @ range-operative)
 (U @ class-attribute)
 (U @ residual)
 (U @ non-location)
 (U @ non-extent))

((travel-agent -{=} S_NEED-SAT-PROB)

((S_NEED-SAT-PROB -[]) chances-not-at-issue)

((S_NEED-SAT-PROB -[]) chances-good)

((S_NEED-SAT-PROB -[]) chances-not-good
 (U @ modal)
 (U<Modal @ !X-can))

((travel-agent -{=} S_NEED-SAT-KNOWLEDGE)

((S_NEED-SAT-KNOWLEDGE -[]) need-sat-unknown)

((S_NEED-SAT-KNOWLEDGE -[]) need-sat-known)

((S_NEED-SAT-KNOWLEDGE -[]) need-sat-borderline)

((service -{=} S_UTTERANCE-ROLE)
((S_UTTERANCE-ROLE -[]) content-utterance)

((S_UTTERANCE-ROLE -[]) pause-filler)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;,
;; Start of Gate Section
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

((chances-good formal specific-need-det =) polite-optimistic-quest
 (U @ modal)
 (U<Modal @ !X-would)
 (U<Process @ !859vb))

((chances-good informal specific-need-det =) informal-optimistic-quest
 (U<Process @ !859vb))

((chances-good
 need-sat-known
 flight-mentioned-previously =) can-put-customer
 (U @ modal)
 (U @ operative)
 (U<Modal @ !X-can)
 (U<Process @ !187vb))

((need-sat-known
 chances-not-at-issue
 success-so-far
 search-just-completed =) Ive-got-item
 (U @ present)
 (U @ past-in)
 (U @ range-operative)
 (U<Process @ !771vb))

((need-sat-known chances-not-at-issue incomplete-success =) All-Ive-got
 (U @ class-attribute)
 (U @ singular-subject)
 (U @ nominal-subject)
 (U<Carrier<Deictic @ !sum-negative)
 (U<Carrier @ determiner-head)
 (U<Carrier @ reduced-rel))

((need-sat-unknown followup =) check-alternative
 (U @ unmarked-yes/no-theme)
 (U @ modal)
(U<Modal @ !X-can))

((need-sat-borderline followup =)-) Would-X-be-OK)

((pause-filler search-about-to-begin =)-) initial-search-pause
   (U @ oblate)(U @ unmarked-imperative-theme)
   (U @ residual)(U @ operative)
   (U @ positive)
   (U<Residual @ personal)
   (U<Residual @ singular)
   (U<Residual<Head @ !first)
   (U<Residual<Head @ !objective)
   (U<Beta<Process @ 1459vb)
   (U<Beta @ perception)
   (U<Beta @ positive)
   (U<Beta @ no-residual))

((flying-times original-question =)-) setting-flight-dates
   (U @ wh-adjunct))

((general-need-det return-info flight-mentioned-previously |-) addressee-medium
   (U<Medium @ pronoun-head)
   (U<Medium<Head @ !second)
   (U<Medium<Head @ !singular-pronoun)
   (U<Medium<Head @ !objective))

((incomplete-success search-in-progress =}-) some-search-success
   (U @ singular-subject)
   (U @ nominal-subject)
   (U @ range-operative))

((incomplete-success system-responsibility =}-) blame-on-system
   (U<Carrier<Beta @ declarative)
   (U<Carrier<Beta @ present)
   (U<Carrier<Beta @ present-in)
   (U<Carrier<Beta @ operative)
   (U<Carrier<Beta @ speaker-subject)
   (U<Carrier<Beta @ positive)
   (U<Carrier<Beta @ no-residual)
   (U<Carrier<Beta @ material))

((incomplete-success =}-) some-return-success
   (U<Attribute<Head @ !296n)
   (U<Attribute @ noun-head)
((unsuccessful-so-far search-in-progress =\}-) search-failure
   (U @ present-in)
   (U @ operative)
   (U @ time-adjunct)
   (U @ material)
   (U<Process @ !484vb)
   (U<Medium @ quasi-neg-generic)
   (U<Temporal @ non-pre-modified-ap)
   (U<Temporal @ non-post-modified-ap)
   (U<Temporal<Head @ !108adj))

((search-just-completed \}-) temporal-flight-description
   (U<Spatial @ unmarked-prep-phrase)
   (U<Spatial<Minor-process @ !prep-positive)
   (U<Spatial<Minor-process @ !prep-position)
   (U<Spatial<Minor-process @ !line/surface)
   (U<Spatial<Range<Head @ !296n)
   (U<Spatial<Range @ singular)
   (U<Spatial<Range @ determined)
   (U<Spatial<Range @ classified)
   (U<Spatial<Range @ noun-head))

((flight-unmentioned-previously \}-) seat-found
   (U<Range<Head @ !187n)
   (U<Range @ singular)
   (U<Range @ determined)
   (U<Range<Deictic @ !non-add)
   (U<Range @ noun-head))

((report \}-) fare-description
   (U<Carrier<Head @ !809n)
   (U<Carrier @ noun-head)
   (U<Carrier @ singular)
   (U<Carrier<Deictic @ !non-selective)
   (U<Carrier @ determined)
   (U<Carrier @ classified)
   (U<Carrier<Classifier @ !314n)
   (U<Carrier<Classifier @ !singular))
((report |-) monetary-nominal
  (U<Attribute @ non-determined)
    (U<Attribute @ noun-head)
      (U<Attribute @ singular))

))

(setq defaults nil)
BIBLIOGRAPHY


233


*Abductive Inference: Computation, Philosophy, Technology.* Cambridge University Press.


