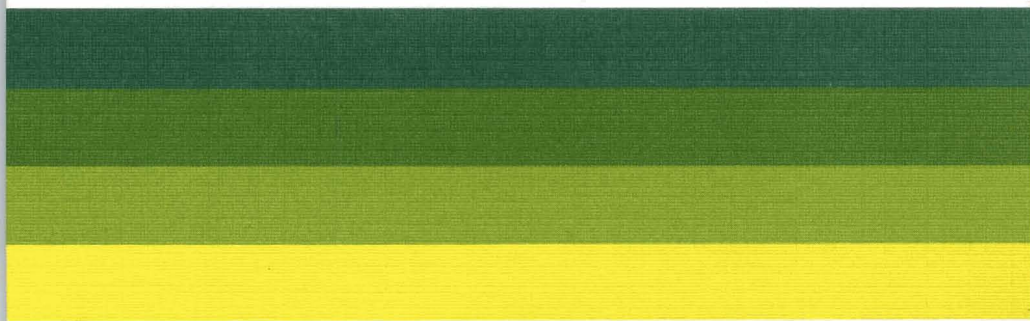


HOW TO USE *IT*

A GUIDE TO INTERLINEAR TEXT PROCESSING

Gary F. Simons
Larry Versaw

Revised Edition,
Version 1.1




SUMMER INSTITUTE
 OF LINGUISTICS




HOW TO USE *IT*

A Guide to Interlinear Text Processing

**Revised Edition,
Version 1.1**



**Gary F. Simons
Larry Versaw**



**Summer Institute of Linguistics
Dallas, Texas**

This book is sold with the software it describes. That software, too, is the copyrighted property of the Summer Institute of Linguistics. However, in the interest of sharing the fruit of our research with the larger academic community, the registered owner of the *IT* software is granted the right to share copies of the distribution diskette with friends and associates, provided this is not done for commercial gain. Such recipients of the software, if they decide to use it in their research, should in turn become registered owners by buying this book with its latest version of the software.

This book was set by the authors, using Microsoft Word and Xerox Ventura Publisher, and printed on a Hewlett Packard Laser Jet II printer. Plates were prepared by 80% photoreduction of the Laser Jet output.

© 1988 by the *Summer Institute of Linguistics*
All rights reserved

ISBN 0-88312-735-0

Version 1.0 May 1987, 300 copies

Version 1.1 January 1988, 500 copies

Distributed by:

Bookstore
Summer Institute of Linguistics
7500 W. Camp Wisdom Road
Dallas, TX 75236

Telephone: (214) 298-3331

CONTENTS

1. Introduction

1.1 Why <i>IT</i> ?	1-2
1.2 About this book	1-4
1.3 Project history and acknowledgements	1-6

2. Applications of *IT*

2.1 Text as a multidimensional object	2-2
2.1.1 The multidimensional nature of text	2-2
2.1.2 Representing multidimensional text	2-4
2.1.3 Kinds of information in multidimensional text	2-6
2.2 Unit identification	2-8
2.2.1 Defining the unit size	2-8
2.2.2 Defining the referencing scheme	2-10
2.3 The baseline text	2-14
2.4 Aligning annotations	2-16
2.4.1 Alternate transcriptions	2-19
2.4.2 Allomorphic transcription	2-20
2.4.2.1 Conventions for word division	
2.4.2.2 Handling discontinuous allomorphs	
2.4.3 Morphemic representation	2-24
2.4.3.1 The merits of morphemic representation versus allomorphic transcription	
2.4.3.2 Handling alternation, suppletion, and fusion	
2.4.3.3 Handling discontinuous morphemes	
2.4.4 Word and morpheme glosses	2-30
2.4.4.1 Informal versus formal text glossing	
2.4.4.2 Word glossing	
2.4.4.3 Glossing content morphemes	
2.4.4.4 Handling untranslatable and unknown elements	
2.4.4.5 Glossing functor morphemes	
2.4.5 Lexemic representation	2-38

2.4.6	Grammatical analysis	2-40
2.4.6.1	Labeling grammatical categories	
2.4.6.2	Identifying word and morpheme categories	
2.4.6.3	Bracketing syntactic structures	
2.4.6.4	Tagging grammatical functions	
2.4.7	Semantic notions	2-48
2.4.7.1	Semantic case roles	
2.4.7.2	Semantic subcategorization	
2.4.7.3	Participant indexing	
2.4.8	Computer-assisted dialect adaptation	2-54
2.4.8.1	Adapting directly from one dialect to another	
2.4.8.2	Adapting from an intermediate form to many dialects	
2.5	Freeform annotations	2-60
2.5.1	Translations	2-62
2.5.2	Explanatory comments	2-66
2.5.3	Topic indexing	2-68
2.5.4	Housekeeping notes	2-71
2.5.5	When the base text is a translation	2-74
3.	Basic Concepts	
3.1	Four components of the <i>IT</i> database	3-2
3.2	Mappings	3-4
3.2.1	Mappings in algebra	3-4
3.2.2	Mappings in linguistics	3-6
3.3	The interlinear text component	3-8
3.4	The lexical mappings component	3-10
3.5	The interlinear text model component	3-11
3.6	The range set component	3-14
3.7	Control files	3-16
3.8	Representing <i>IT</i> data in lexical database files	3-18
3.9	Representing <i>IT</i> data in standard format	3-20
4.	Overview of the <i>IT</i> family	
4.1	Functional overview	4-2
4.2	<i>IT</i> tools manipulate data	4-4
4.3	<i>IT</i> tools to manipulate interlinear text and text models	4-5
4.4	<i>IT</i> tools to manipulate the lexical database	4-6
4.5	<i>IT</i> tools to manipulate range sets	4-7
4.6	<i>Itprep</i> : interlinear text preparation	4-8
4.7	<i>Ita</i> : interlinear text aligner	4-10

4.8 <i>Itcheck</i> : checking the model and interlinear text	4-11
4.9 <i>Iip</i> : interlinear text processor	4-12
4.10 Tools for maintaining the lexical database	4-14
4.11 <i>Itl-dir</i> and <i>itl-bak</i> : housekeeping with lexical database files	4-16
4.12 Planned additions to the family	4-18
4.12.1 Formatting interlinear text for publication	4-18
4.12.2 Interlinear text search facility	4-20

5. Recipes for using *IT*

5.1 Use <i>IT</i> for the first time	5-4
5.2 Convert existing dictionary files to <i>IT</i> 's internal format	5-5
5.3 Gloss an interlinear text created by another system	5-6
5.4 Gloss a text	5-7
5.5 Regloss an <i>IT</i> text using up-to-date lexical database	5-8
5.6 Add another type of annotation to a text	5-9
5.7 Remove an annotation field from a text	5-10
5.8 Extract examples from an annotated text	5-11
5.9 Format and print an annotated text	5-12
5.10 Assign/add values to a mapping	5-13
5.11 Restrict the range of annotations to a specific set of abbreviations or values	5-14
5.12 Remove a troublesome range constraint	5-16
5.13 Print or display the contents of the lexical database	5-17
5.14 Look-up/create/modify/delete an entry in the lexical database	5-18
5.15 Clean the bad data out of a mapping	5-19
5.16 Retrieve only the recently updated information from the lexical database	5-20
5.17 Assign/add members to a range set	5-21
5.18 Print or display the members of a range set	5-22
5.19 List the names of mappings stored in a lexical database file	5-23
5.20 See if it is time to back up a lexical database file	5-24
5.21 Resize a lexical database file	5-25
5.22 Make a lexical database file a particular size	5-26
5.23 Merge two or more copies of the same mapping or range set	5-27
5.24 Speed things up	5-28
5.25 Deal with warnings and errors	5-29

6. Ingredients for recipes

6.1 Put a file into ASCII format	6-4
6.2. Get text into ITX format	6-6
6.3 Verify that a text is in ITX format	6-8
6.4 Put file into <i>IT</i> 's standard format lexical representation	6-10
6.4.1 Make a glossary style lexical file	6-10
6.4.2 Make a dump style lexical file	6-12
6.4.3 Make a file header for standard format lexical file	6-14
6.4.4 Declare a mapping	6-16
6.4.5 Enter the mapping data	6-17
6.4.6 Enter range sets into a dump style lexical file	6-20
6.5 Make a text model	6-22
6.5.1 Define the reference field	6-24
6.5.2 Define the baseline text field	6-25
6.5.3 Define an aligning field	6-27
6.5.4 Define a freeform field	6-30
6.6 Delete a field from a text model	6-32
6.7 Make a <i>mppg-gen</i> control file	6-34
6.8 Make an <i>itprep</i> control file	6-36
6.8.1 Define the markers in a standard format input text file	6-38
6.8.2 Instruct <i>itprep</i> how to divide the text into units	6-40
6.9 Make a <i>lex-extr</i> control file	6-42
6.10 Make a <i>lex-add</i> control file	6-44
6.11 Run <i>itprep</i>	6-46
6.12 Run <i>ita</i>	6-47
6.13 Run <i>itcheck</i>	6-48
6.14 Run <i>itp</i>	6-49
6.14.1 Overview of user interaction	6-50
6.14.2 Automatic versus manual sequencing modes	6-52
6.14.3 Entering the first annotation for a base form	6-53
6.14.4 Entering additional annotations for a base form	6-56
6.14.5 Entering freeform annotations	6-57
6.14.6 Changing/deleting a word or annotations	6-58
6.14.7 Using <i>itp</i> in verify mode	6-62
6.14.8 Skipping over already-annotated text	6-63
6.14.9 Using special characters	6-64
6.14.10 Speeding <i>itp</i> up	6-66
6.15 Run <i>mppg-gen</i>	6-68
6.16 Run <i>lex-extr</i>	6-69
6.17 Run <i>lex-add</i>	6-71
6.18 Run <i>lex-edit</i>	6-72

6.19	Run <i>itl-dir</i>	6-73
6.20	Run <i>itl-bak</i>	6-74
7.	Making an <i>IT</i> user interface	
7.1	What is an <i>IT</i> user interface?	7-2
7.2	A sample <i>IT</i> user interface program	7-5
7.3	How to make your own <i>IT</i> user interface	7-6
8.	Reference guide to <i>IT</i> programs	
8.1	<i>Itprep</i>	8-3
8.2	<i>Ita</i>	8-5
8.3	<i>Itp</i>	8-6
8.4	<i>Mppg-gen</i>	8-8
8.5	<i>Lex-extr</i>	8-9
8.6	<i>Lex-add</i>	8-11
8.7	<i>Lex-edit</i>	8-13
8.8	<i>Itl-dir</i>	8-14
8.9	<i>Itl-bak</i>	8-15
8.10	<i>Itcheck</i>	8-16
9.	Reference guide to <i>IT</i> file formats	
9.1	Standard format	9-2
9.2	Interlinear text file format	9-4
9.3	Internal lexical file format	9-6
9.4	Standard format lexical file format	9-7
9.4.1	Dump style SFL file format	9-7
9.4.2	Glossary style SFL file format	9-10
9.5	Control file format	9-12
9.6	<i>Itprep</i> control file specifications	9-14
9.7	<i>Mppg-gen</i> control file specifications	9-17
9.8	<i>Lex-add</i> control file specifications	9-18
9.9	<i>Lex-extr</i> control file specifications	9-19
9.10	Interlinear text model file specifications	9-21
10.	Messages	10-1
Appendix —	Notes on current release	
A.1	Files on <i>IT</i> release diskette	A-2
A.2	Differences between versions 1.0 and 1.1	A-5
A.3	Planned enhancements and upgrades	A-6
Glossary		G-1
References		R-1
Index		X-1

INTRODUCTION

1.1 Why *IT*?

1.2 About this book

1.3 Project history and acknowledgements

1.1 Why *IT*?

IT is a tool for developing an analyzed corpus of text. It assists the analyst in developing interlinear annotations for a user-defined collection of dimensions.

The analysis of text is absolutely central to the study of language, literature, and culture. A corpus of text, as a storehouse of examples of language in use, is the foundation on which the description of the phonology, grammar, and lexicon of a language is based. A corpus of text, as the definitive collection of works by a particular author or in a particular genre, is the primary source for a study of literary style. A corpus of text, as it exemplifies the knowledge and world view of a people, is also an indispensable resource in the study of culture.

The quality of the conclusions the investigator can reach in such studies is directly related to the size and quality of the text corpus on which they are based. To develop and manage an analyzed text corpus of adequate dimensions is a task that quickly outstrips the manual data processing capacity of the typical analyst. It is a data processing task of tremendous proportions, and requires the use of *the* tool of data processing—the digital computer.

The use of computers for text analysis in linguistics, literary studies, and anthropology has been practiced for about three decades, first on mainframe computers and more recently on personal microcomputers. The standard practice has been to use text editors (or before that, keypunch machines) to get a corpus of texts into machine readable form, and then use a retrieval tool (such as a concordance program) to find and group examples of interest. Recent advances in technology, like the battery-powered laptop computer, have made it possible for the field researcher to employ this approach in the field, beginning right at the point of text collection.

This book describes a software system for text corpus development which promises to open new dimensions for the text analyst. It is called *IT*, for Interlinear Text system. Just as the technology of text editors and concordance programs represented a quantum leap beyond the manual manipulation of data notebooks, so we believe something like the *IT* system offers a quantum leap beyond text editors and concordances.

The essential innovation of *IT* is that, as its name proclaims, it manages a corpus of interlinear texts. Rather than viewing text as a long one-dimensional string of characters, *IT* views text as a sequence of text units, each of which contains a text line plus a multidimensional set of annotations provided by the analyst. It is in these annotations that the analyst practices his craft—identifying known elements, marking unknown items, tagging phenomena of interest for later retrieval, and posing questions for further study.

The centerpiece of *IT* is a program called *itp*—the interlinear text processor. It is analogous to a text editor which has built-in knowledge about how analysts build interlinear, annotated texts. Given the user-supplied model of the desired contents of the fully annotated text, *itp* leads the user through the process of annotating the text, ensuring all the while that the resulting interlinear text conforms to the specified model. At the same time *itp* ensures consistency in annotation by retrieving all interlinear word and morpheme annotations from an on-line database of lexical information which it maintains. The interlinear text file produced by *itp* is a clean ASCII text file without embedded secret characters. Thus it is accessible to other text processing software for purposes such as concordancing, indexing, or display formatting. The *IT* package includes a collection of other software tools which support the conversion of conventional texts to interlinear format and which support the maintenance of the auxiliary lexical database files.

The book begins in chapter 2 by discussing the multidimensional nature of text and illustrating dozens of applications for which *IT* can be employed. Chapter 3 begins the tutorial introduction to the computer programs by defining many of the concepts on which they are based. Chapter 4 gives an overview of all the parts of the *IT* system showing how they fit together. Chapters 5 and 6 give step-by-step instructions for using the system, first by enumerating twenty-five recipes for performing common tasks and then by giving detailed instructions for performing each of the steps which are the ingredients for the recipes. Chapter 7 is something of a diversion, discussing how customized user interface programs can be developed to make *IT* easier to use. The remaining chapters comprise a reference guide to *IT*—chapter 8 documents each of the programs in the system, chapter 9 describes the format of every type of data and control file in the system, and chapter 10 gives an alphabetical listing of all error messages with an explanation and recommended course of action for each.

APPLICATIONS OF *IT*

2.1 Text as a multidimensional object

- 2.1.1 The multidimensional nature of text**
- 2.1.2 Representing multidimensional text**
- 2.1.3 Kinds of information in multidimensional text**

2.2 Unit identification

- 2.2.1 Defining the unit size**
- 2.2.2 Defining the referencing scheme**

2.3 The baseline text

2.4 Aligning annotations

- 2.4.1 Alternate transcriptions**
- 2.4.2 Allomorphic transcription**
- 2.4.3 Morphemic representation**
- 2.4.4 Word and morpheme glosses**
- 2.4.5 Lexemic representation**
- 2.4.6 Grammatical analysis**
- 2.4.7 Semantic notions**
- 2.4.8 Computer-assisted dialect adaptation**

2.5 Freeform annotations

- 2.5.1 Translations**
- 2.5.2 Explanatory comments**
- 2.5.3 Topic indexing**
- 2.5.4 Housekeeping notes**
- 2.5.5 When the base text is a translation**

2.1 Text as a multidimensional object

Text is a multidimensional object, and *IT* is a tool which allows the analyst to treat it as such. This section discusses the multidimensional nature of text, describes the manner in which *IT* represents multidimensional text in computer data files, and defines the kinds of information which *IT* recognizes in multidimensional text.

2.1.1 The multidimensional nature of text

The single sequence of characters by which we conventionally transcribe text is at once representing structure and meaning in many simultaneous dimensions. The analyst's perspective on the text embodies further dimensions of analyzed text.

A conventional text editing program views text as a simple sequence of characters. But from a linguist's perspective, text is far more than that. We may write down a text as a one-dimensional sequence of characters, but the object we are thereby representing is in fact a multidimensional one. The stream of speech which we hear and transcribe has form and meaning in many dimensions at once. The speech signal itself simultaneously comprises articulatory segments, pitch, timing, and intensity. A given stretch of speech can be simultaneously viewed in terms of its phonetic interpretation, its phonemic interpretation, its morphophonemic interpretation, its morphemic interpretation, or its lexemic interpretation.

Text also exhibits structure at many levels at once. Its structure may be viewed in terms of syllables, stress groups, intonation groups, or pause groups, or in terms of morphemes, words, phrases, clauses, sentences, paragraphs, and so on.

The meaning of the text operates in many dimensions as well. There is the lexical definition of the morphemes. There is the functional meaning carried by the constituents of the grammatical constructions. An utterance may have a literal meaning as well as a figurative one, or a denotative meaning as well as a connotative one. All of these dimensions, and more, lurk behind that one-dimensional sequence of characters which we think of as a text.

The participation of an analyst introduces further dimensions. Besides the analyst's interpretation of what is being observed in text, the analyst notes patterns, indexes examples for salient features they demonstrate, and has questions about matters of transcription or analysis which must be rechecked. These, too, are dimensions of the analyzed text.

The essential insight of the *IT* system is that text is multidimensional, and that to do a good job of text analysis, one must distinguish those dimensions and annotate them separately. Beginning with chapter 3, this book describes how to use *IT* to build an annotated corpus of text. In this chapter, we set the stage by illustrating many possible dimensions of annotation that an analyst could use.

2.1 *Text as a multidimensional object*

2.1.2 Representing multidimensional text

IT stores annotated text in clean ASCII text files which employ user-defined labels beginning with backslash to encode the structure of the multidimensional annotations. This format makes the information easily accessible to a human reader of the file, as well as to any other software that can manipulate a clean ASCII text file.

From the computing perspective, a multidimensional text is actually a database. The different annotations of the text are analogous to the different fields of information in a database. A unit of text plus its annotations comprise a record of the database. Thus the *IT* system is like a special-purpose database management system for annotated text.

Unlike most database programs, one of the fundamental design principles for the *IT* system was that the interlinear text files it manipulates should not have a unique format with secret characters and the like; rather, they should be clean ASCII text files.¹ The advantage of this is that the interlinear texts produced through *IT* can be manipulated by other software. One can use operating system commands to view them on the screen or dump them to a printer. They can be manipulated by any conventional text editor, or by a procedure written in the analyst's favorite programming language.

To preserve the database structure of interlinear texts while storing them as conventional text files, *IT* uses the SIL standard format. This is a convention which prescribes that different types of information in text files should be marked by unique codes beginning with the backslash character.² For instance, figure 2.1 gives a fragment of an interlinear text file (from the Lau language of Malaita, Solomon Islands) represented with standard format conventions.

The sample text in figure 2.1 begins with header information: `\id` identifies the abbreviated code name for the text, `\tit` gives an English title, and `\nar` describes the narrator. The marker `\p` indicates the beginning of a paragraph. Then follow the first two sentences of the text. All of the fields of information relating to a particular sentence amount to a record of the database. The `\ref` field is the unit identifier which marks the boundaries between text units (in this case sentences) and provides a unique reference string for the unit's place

in the corpus. The unit of text and its annotations are in four fields: `\tx` gives the baseline text, `\mr` gives a morphemic representation of the text, `\mg` gives morpheme glosses for the text, and `\ft` gives a free translation of the sentence. Note that the field names in this example range from one to three characters long. In the *IT* system the names can be as long as you want; the field name must begin with a letter and includes everything up to the next space, tab, or carriage return. Be aware, however, that some SIL software limits the length of field names to four characters.

Figure 2.1 The representation of multidimensional text
(Lau, Solomon Islands)

```

\id BARU
\tit When Baraisau built a canoe
\nar Joseph Akwasitoloa, January 1983

\p
\ref BARU s 1
\tx Tee wane na hatana 'a Baraisau 'e kasia
\mr tee wane na hata-na 'a Baraisau 'e kasi -a
\mg one man DEF name-3s.P MASC Baraisau 3s.G build-3s.O

\tx tee baru.
\mr tee baru
\mg one canoe

\ft A man named Baraisau built a canoe

\ref BARU s 2
\tx Ma nia ka 'alangia na baru nae 'ana
\mr ma nia ka 'ala-ngi-a na baru nae 'ana
\mg and 3s 3s.S call-TR -3s.O DEF canoe this COMPL

\tx na Mouanilofa.
\mr na Mouanilofa
\mg DEF Mouanilofa

\ft and he called it Mouanilofa.
```


2.1 Text as a multidimensional object

2.1.3 Kinds of information in multidimensional text

IT's model of multidimensional text distinguishes five kinds of information in an interlinear text file: the unit marker, the baseline text, aligning annotations, freeform annotations, and extraneous fields.

The fields of information in an *IT* multidimensional text fall into five general kinds: the unit identification, the baseline text, the aligning annotations, the freeform annotations, and extraneous fields. Each of these five kinds of information is defined in this section; the first four are exemplified in sections 2.2 through 2.5.

In a given interlinear text, one field code is defined as the unit identification marker. It serves two functions: (1) to mark the beginning of each unit in the text (that is, each record in the database), and (2) to specify a unique reference identifier for each unit of the text. This reference serves to identify the source of each example when examples are extracted from an interlinear text corpus. In figure 2.1, `\ref` is the unit identification marker.

There is also a single code defined as the baseline text marker. The baseline text is the text as it would read if it were a conventional one-dimensional text. The other fields of the database are annotations describing the many dimensions of this baseline text. In figure 2.1, `\tx` marks the baseline text.

An interlinear text file can contain an arbitrary number (including zero) of aligning annotations. These are annotations of the baseline text which are done on a word-by-word or morpheme-by-morpheme basis. They are called aligning because the first character of the annotation always aligns vertically with the first character of the word or morpheme in the baseline text which it annotates. A further distinctive of aligning annotations is that *IT* saves the mapping from baseline words and morphemes to their annotations in a lexical mappings database (see section 3.2). This enables the annotation process to proceed automatically once the annotation for a particular word or morpheme has been entered. Possible aligning annotations include underlying forms, word or morpheme glosses, syntactic classes, bracketed constituent structure, and semantic categories. In figure 2.1, `\mr` (for morphemic representation) and `\mg` (for morpheme glosses) mark aligning annotations.

An interlinear text file can also contain an arbitrary number (including zero) of freeform annotations. These are annotations of the baseline text that refer to the text unit as a whole. Possible freeform annotations include translations of the text, topical keywords for indexing of text units, explanations of background information, or questions for further checking. In figure 2.1, `\ft` (for free translation) marks a freeform annotation.

Any field marker which is not defined as a unit identification marker, a baseline text marker, an aligning annotation marker, or a freeform annotation marker is an extraneous marker. The programs of the *IT* system ignore extraneous fields. If they are encountered on input, extraneous fields are passed through to the output unchanged. These may occur anywhere in an interlinear text file, except in the midst of the aligning annotations. Possible extraneous fields include higher level division markers (such as for paragraphs, episodes, chapters, and the like), markers giving format information for display programs, or markers specifying header information at the beginning of the file. In figure 2.1, `\id` (for text identification), `\tit` (for title), `\nar` (for narrator), and `\p` (for paragraph) are examples of extraneous fields.

Figure 2.2 Kinds of information in multidimensional text

Unit identification	defines text unit boundaries and gives reference for location of unit in corpus (see figures 2.3 and 2.4 for examples)
Baseline text	the text being annotated (see figure 2.5 for examples)
Aligning annotations	vertically aligned morpheme-by-morpheme or word-by-word annotations of the baseline text or of another annotation (see figure 2.6 for examples)
Freeform annotations	annotations of an entire unit of baseline text or of another annotation (see figure 2.27 for examples)
Extraneous fields	any other fields of information such as format information or file headers; ignored by <i>IT</i>

2.2 Unit identification

IT treats text as a sequence of text units. Dividing the text into units serves two fundamental functions: (1) the unit is the span to which freeform annotations are attached, and (2) the unit is the span to which unique reference identifiers are attached so that extracts from the text can be located within the whole corpus.

2.2.1 Defining the unit size

The unit is the span of text to which *IT* attaches freeform annotations, such as free translations. It is thus important to select a unit size which is not too small to be annotated, and yet not so large as to contain many units that could be annotated individually. Running text may be divided grammatically into clause or sentence units, or phonologically into pause groups or intonation groups. Poetic texts generally have a built-in line structure based on meter and rhyme. In a field notebook, each observation can be treated as a unit.

There are many possible ways to organize a collection of connected text. Defining the unit size is the most fundamental decision to be made. From a grammatical perspective, there are two primary candidates: the sentence and the clause. Dividing the text into sentences has the advantage that it can be done automatically on the basis of sentence final punctuation marks. It has the potential disadvantage that many of the units may end up too large for detailed analysis of low level phenomena. Dividing the text into clauses may solve that problem, but can introduce the opposite one, namely, that some clauses may be so small (such as a single word) as to be inconvenient. Translatability is a key issue here. Since the unit is the span to which the freeform annotations are attached, one must ensure that the units are large enough, for instance, to take a free translation. When opting for clause-sized units, one cannot enjoy the luxury of having the computer automatically go through a text and divide it into units. The analyst must make all the unit breaks by hand. This being the case, it is possible to avoid the problem of units which seem too large or too short by manually splitting or combining in some principled way. Section 6.8.2 describes how to prepare a text in this way for input to *IT*.

The unit size may also be defined from a phonological perspective. In this case, we use phonological clues like pauses and intonation contours to break the text into units. These units could be called pause groups or intonation groups. Though one could attempt to define the breaks strictly on the basis of pause, or strictly on the basis of intonation, in practice a disjunctive definition of unit breaks is likely to work best. Sometimes there will be a pause with no clear intonation clue; sometimes there will be a clear break between intonation contours without any pause; and sometimes both will occur. Breaking the text on this basis is likely to yield units of a fairly uniform length which vary between clause and sentence in scope. Presumably such unit breaks would always occur at sentence boundaries. If comma is used as an orthographic symbol which marks this kind of phonological break when it occurs sentence medially, then the computer could automatically segment the text into units of this nature. Otherwise, manual processing to mark units will be required.

If the text is a form of verse, in which the phonology of rhyme or meter defines the structure of the text, then these phonological features must certainly be followed in defining the units. In this case, the text unit would probably be called a "line."

In the early stages of linguistic field work, it is common to record in data notebooks elicited paradigms, elicited sentences, and items heard in natural speech. Provided the analyst observes the usual precautions about using utterances recorded out of context, these observations can be made into a useful corpus for interlinear annotation and analysis. The unit size would be the single recorded utterance, be it a word, phrase, clause, sentence, or even sentence cluster.

Figure 2.3 Possible text units

<i>Type of text</i>	<i>Possible unit size</i>
Running text	clause sentence intonation group pause group
Poetic text	line
Biblical text	verse
Field notes	observation

2.2 Unit identification

2.2.2 Defining the referencing scheme

The referencing scheme defines a way of identifying the location of a particular text unit within the text and within the entire corpus of texts. Published texts often have established referencing schemes. In a collection of unpublished texts, it is up to the investigator to devise a referencing scheme.

When extracting examples from a large text corpus it is essential to know which text a particular example is from, and where in that text it comes from. For this purpose, a referencing scheme must be devised which assigns a unique identifier to each text in the corpus and further uniquely identifies each unit within a text. In *IT*, the unit identification field not only shows where one unit ends and the next begins, it also specifies the unique reference tag for the unit which follows.

The text corpus may be drawn from published sources, whether the classical literature of the ancient world, the literature of the modern world, or the published oral traditions of a preliterate society. In such published texts, there is usually a system of unit division and referencing already established.

Scholars have been citing references to classical texts for centuries, and there are universally accepted referencing schemes for these. The most well-known is the system of book, chapter, and verse used in the *Bible*. This system, as with all discussed here, is a hierarchical one — the *Bible* is comprised of a number of books, each of which is comprised of a number of chapters, which in turn are comprised of a number of verses. Any verse in the *Bible* is uniquely identified by citing the name of the book, the number of the chapter, and the number of the verse. To process biblical texts with *IT* one would define the verse as the unit size. Thus a unit marker would be placed at the beginning of each verse. The unit marker would be followed by the reference identifier, which in this case would include identification of the book, chapter, and verse. The full name of the book could be used (provided names did not contain spaces), but in the interest of conserving space, a system of abbreviations would best be used. The chapter and verse would be identified by a number. For instance, the

following could be the full unit identification field for the first verse in the *Bible*:

\unit GEN 1 v 1

That is, the book of *Genesis*, chapter 1, and verse 1. Note that the types of the higher level elements in the reference (in this case, book and chapter) are left implicit. (They are defined in the text model; see section 6.5.1) However, the unit level identifier (always the last in the reference string) is preceded by an identifier for the type of unit, in this case, *v* for verse. This makes it possible to distinguish text units of different types. For instance,

\unit GEN 1 st 1

might identify the first section title within chapter 1 of *Genesis*. The *itprep* program is a preprocessor which takes text with division markers for book, chapter, verse, section title, or whatever the scheme in the particular text, and converts it to an interlinear text file with this style of unit referencing. The details of that procedure are described in sections 6.8 and 6.11

There are many other referencing schemes in use for classical texts. For instance, the *Iliad* and the *Odyssey* are referenced by book (designated by a Greek letter) and line number, such as, *Il. A.105*. In this case, the line would be the unit. The history of Thucydides is organized into numbered books, chapters, and verse-like divisions consisting of one to three sentences. References are conventionally cited as, for instance, *vii. 17. 4*. The works of Plato are referred to by name of work, page number from the original Stephanus edition, section (designated by letter), and line number, for instance, *Ap. 18 d 2*. The dramas of Aeschylus, Aristophanes, and Euripedes are cited simply as name of work and line number, for instance, *Nub. 1095*.

Many classics of English literature also have widely accepted referencing schemes. For example, passages from the plays of Shakespeare are referenced by act, scene, and line. The acts and scenes are normally cited with Roman numerals, such as, *IV. iii. 214*. Though the identification of the speakers is a salient feature in the organization of a play, it does not conventionally figure into the referencing scheme. In *IT*, the name of the speaker would be encoded in an extraneous field before each line at which there is a change of speaker. The *Canterbury Tales* of Chaucer have still another referencing scheme. The tales are traditionally partitioned into nine groups designated by capital letters, and then the lines are numbered within each group. For instance, the Friar's tale is cited as *D 1301-1664*.

Modern works of prose literature, such as novels, do not have built-in referencing schemes down to the unit level. They generally have numbered chapters. Many also have the chapters divided into sections. But as for unit divisions, the analyst would apply the kinds of principles discussed in the preceding module to devise a scheme for dividing the text into units. If working from a standard edition, it might also prove useful to put the page number in the referencing scheme, so that the number of the source page would appear with all extracted examples.

When working with a contemporary corpus, the analyst is not constrained by the traditions of centuries. The analyst is therefore free to define a unit size and referencing scheme to suit his own purpose. A two-level scheme is sufficient for most purposes. The top level division would be the texts themselves. For the sake of referencing, an abbreviated name would be devised for each text. The second level would be the units, which would be referenced by number. Thus, *Hare 5* might refer to unit 5 of *The Hare and the Tortoise*, while *Wolf 13* would refer to the thirteenth unit of *The Shepherd Boy and the Wolf*.

When texts are long (such as many hundreds of units) it may prove helpful to introduce a third level into the referencing scheme. If the text already has explicit divisions into chapters or the like, then these should be followed. In the case of a lengthy narrative, there would likely be natural breaks at which it could be divided into episodes. For the sake of referencing, these would typically be numbered rather than named. A lengthy expository or hortatory text could be similarly divided into sections.

Paragraph breaks are conventionally marked in texts of all types. In general, paragraph breaks would be marked in *IT* with an extraneous field marker (such as ¶). To include paragraph divisions in the hierarchical referencing scheme could result in a cumbersome system. However, it is perfectly acceptable to do so as far as *IT* is concerned. This could be to the analyst's advantage, for instance, if a particular focus of the analysis were to be the study of paragraph structure, so that it would be possible to see at a glance if two units were in the same paragraph by comparing the references.

A collection of unconnected utterances would require a different approach. Two referencing schemes suggest themselves. (1) Label each data book, number each page, and number each example on each page. The unique reference for an observation is then book label, page number, and example number. (2) Record each

observation by date, with an added sequence number to distinguish multiple observations on the same day. In order to be able to use a general sorting program for keeping observations in chronological order, encode the date as numbers of the form *YY-MM-DD* (for instance, *86-11-18* for November 18, 1986).

Figure 2.4 Possible referencing schemes

<i>Bible</i>	Book name, chapter number, verse number
<i>Iliad</i>	Book letter, line number
Thucydides	Book number, chapter number, division number
Plato	Name of work, Stephanus page number, section letter, line number
Greek drama	Name of play, line number
Shakespeare	Name of play, act number, scene number, line number
<i>Canterbury Tales</i>	Group letter, line number
Modern novel	Chapter number, (section number), (page number), unit number
Text collection	Text name, unit number Text name, section label, unit number
Field notes	Notebook number, page number, observation number Observation date, sequence number

2.3 The baseline text

The baseline text is the text which is the basis for the annotations. It may take virtually any form, since it can always be converted to a more suitable form in an aligning annotation.

The baseline text field of the multidimensional text database is that field which contains the original one-dimensional word-processing form of the text. The difference from its original form is that it is broken up into units. The baseline text is the field which serves as the jumping off point for all of the annotations.

As foundational as this may seem, the form of the baseline text is not really crucial. This is because any field of annotation can serve as the baseline for another. Thus, if the original baseline text is not the representation of the text which seems the most appropriate for the purposes of the analyst, the basic representation can be filled in as an annotation, and then further annotations built off that field.

The baseline text amounts to a possible transcription of the text. It could represent the language in any of the following ways. It could be a phonetic transcription, in which each phone has a unique representation. It could be a phonemic transcription, in which each phoneme has a unique representation. It could be a morphophonemic transcription, in which each morphophoneme has a unique representation. Or the baseline text could be transcribed in a traditional orthography which does not satisfactorily account for the language from a linguistic perspective, or it could be the inadequate transcription of a previous investigator who had not yet mastered the language. Figure 2.5 gives a summary list of possible material for the baseline text.

The conclusion is that whatever the nature of the original transcription of the text, it may serve as an acceptable base line for *IT*. By the use of aligning annotations, the analyst may always build up a refined version of the text which puts a transcription of the desired nature and quality in an aligning field.

Figure 2.5 Possible material for the baseline text

phonetic transcription
phonemic transcription
morphophonemic transcription
morphemic transcription
transcription in traditional orthography
transcription by previous investigator

2.4 Aligning annotations

Word-by-word or morpheme-by-morpheme annotations of the text are called aligning annotations because *IT* automatically keeps these annotations in vertical alignment with the base word or morpheme which they annotate. *IT* further simplifies the task of text annotation by saving the aligning annotations in a lexical database for automatic retrieval when the base word or morpheme recurs. Aligning annotations are suited to any type of brief annotation of words or morphemes.

Before discussing the possible range of aligning annotations in interlinear text, it is necessary to understand two things about the way *IT* handles aligning annotations—the nature of the alignment and the role of the “lexical mappings database.” After describing the general mechanics of aligning annotations in this module, the remaining modules of this section illustrate their use with a number of examples.

Aligning annotations are made on a word-by-word or morpheme-by-morpheme basis. The name “aligning” comes from the fact that *IT* shows which word or morpheme a particular annotation goes with by aligning it vertically with the base word or morpheme so that both begin in the same column on the screen and output file.

The second distinctive of aligning annotations is that the annotations are saved in a file so that they can be reused automatically when the same word or morpheme appears again in the base text. It is the combination of these two features—that the annotations are automatically aligned and that the annotations are automatically retrieved—which allows *IT* to take most of the tedium out of the preparation of interlinear annotated texts.

The files in which the aligning annotations are saved comprise the “lexical mappings database,” or “lexical database” for short. The lexical database stores the mapping relationship between base words or morphemes and their annotations as glosses, category labels, underlying forms, equivalents in other dialects, or whatever kind of information the analyst has set up. (The notion of “mapping” is a technical term borrowed from algebra; it is thoroughly explained in section 3.2.) The lexical database files are stored in a format which optimizes efficiency of retrieval; in the typical case, *IT* can look up the possible annotations for a particular word or morpheme with just one disk access.

The lexical database is built dynamically and interactively as new words and morphemes are encountered in texts, and as new annotations for old words and morphemes are discovered. Normally an aligning annotation cannot enter an interlinear text except it come out of the lexical database. Thus the lexical database not only affords the convenience of semi-automatic text annotation, it also provides control for consistency in text annotation. An aligning annotation need not always be saved in a lexical database, however. The user may override the default behavior and specify a mapping relationship as not saved or as optionally saved. This would be desirable when the annotations are not lexical properties of words or morphemes (such as syntactic bracketing) or when highly agglutinated words are not expected to recur in text, respectively.

Figure 2.6 gives a listing of possible aligning annotations. *IT* has no built-in requirement to use any of these. It is completely up to the user to define the field codes and mapping relations for all desired

Figure 2.6 Possible aligning annotations

- Alternate transcriptions
 - phonetic
 - phonemic
 - morphophonemic
 - fast speech
 - careful speech
 - other orthography
 - other investigator
- Allomorphic transcription
- Morphemic representation
- Lexemic representation
- Morpheme glosses
- Word glosses
- Grammatical categories
- Syntactic bracketing
- Functional labels
- Semantic case roles
- Semantic subcategorization
- Participant indexing
- Intonation
- Melody in sung texts
- Rhythm in sung and chanted texts
- Equivalents in other dialects

aligning annotations. Note, too, that every annotation which may eventually be desired for a fully analyzed text need not be defined in advance. Just one or two annotations can be defined for the first pass. Further annotations can be added in subsequent passes over the text. The remainder of this section illustrates the many possibilities suggested in figure 2.6.

Setting up *IT* to insert an aligning annotation in text involves determining the size of the base elements (whether words or morphemes or something in between), naming the mapping relationship, defining its disposition with regard to saving of annotations, and setting up the needed lexical database file. The procedures for doing this are explained in detail in section 6.5.3.

2.4 Aligning annotations

2.4.1 Alternate transcriptions

An aligning annotation can be used to convert the baseline text transcribed in one way (such as phonetically, or by an outmoded orthography) to a form more suited to the investigator's purposes (such as a phonemic transcription or a modern orthography).

One class of aligning annotations is alternate transcriptions of the text. These have already been discussed in section 2.3. For instance, if the baseline text is a phonetic transcription, then one might want to introduce a phonemic (or morphophonemic, or practical orthography) transcription as an aligning annotation. The use of a lexical database file to save the respelling of every word will take the tedium out of the retranscription task.

When working with previously published minor-language texts, it is common for the analyst to retranscribe them into a modern (or otherwise more adequate) orthography. This is especially so when the original writing system was devised by persons not trained in linguistics. Figure 2.7 gives an example of this. It shows a sentence from the To'aba'ita language (Solomon Islands) translation of the *New Testament* book of *Revelation* made in 1923. The writing system devised by the early missionaries did not symbolize the glottal stop which is a high frequency phoneme. The presence and absence of glottal stop distinguishes a host of minimal pairs, such as *abu* 'holy, sacred' and 'abu' 'blood'. These words are homonyms in the original text.

Figure 2.7 Retranscription
(To'aba'ita, Solomon Islands)

```
\ot Ma nau ku rikia kini eri ne kuu ka bula
\rt Ma nau ku rikia kini 'eri ne ku'u ka bula
\wg and I I saw woman that who drank she be drunk

\ot nia ana abuna toa abu ki
\rt nia 'ana 'abuna to'a abu ki
\wg herself with blood of people holy plural
```

Key to field codes:

```
\ot original text
\rt retranscribed text
\wg word gloss
```

2.4 *Aligning annotations*

2.4.2 Allomorphic transcription

An aligning annotation can be used to break words down to their constituent morphemes as a basis for further morpheme-level annotation. A first step is to give an allomorphic transcription which introduces morpheme breaks into the surface forms of the words.

In order to do morpheme-by-morpheme glossing and other types of morpheme-based annotation, it is necessary to have a version of the text which is broken into morphemes. In most cases, the baseline text will not be in that form. We can use an aligning annotation field to introduce a representation of the text which breaks words into morphemes. In this section we discuss the use of a transcription of the text which segments the words into surface level allomorphs. In section 2.4.3 we discuss a more abstract morphemic representation which specifies underlying forms.

2.4.2.1 Conventions for word division

The essence of allomorphic transcription is dividing words into allomorphs. Linguists have developed a number of conventions for making morpheme breaks.

Producing an allomorphic transcription essentially involves taking the baseline words and inserting morpheme breaks to segment them into surface level allomorphs. Over the years, some widely followed conventions for marking morpheme breaks have developed among linguists. The most general convention is to insert a hyphen at all morpheme breaks. It is common, however, for linguists to represent different kinds of morpheme breaks differently.

A list of conventions for marking morpheme breaks is given in figure 2.8. These conventions largely follow Christian Lehmann (1982). He proposes the following standards: plus sign (+) to mark compounding (that is, to separate two base forms which combine to produce a new base form with a new meaning), equal sign (=) to mark derivational affixes (that is, those which derive a new word with a different area of meaning and possibly different word class than the

base form), and hyphen (-) to mark inflectional affixes (that is, those which specify grammatical relations or categories of the base form). For instance, the allomorphic transcription of the English form

baby + sit = ter-s

indicates that the roots *baby* and *sit* form a compound to which is added the derivational suffix =*ter* (namely, a contextual variant of the suffix which derives an agentive noun from a verb) and the inflectional suffix -*s* (which further specifies the derived noun as being plural).

A number of other conventions for the use of hyphen and equal sign can be found in the literature. One uses hyphen for affixes and equal sign for clitics. Another uses hyphen for suffixes and equal sign for prefixes. Still another uses hyphen on a base with a valence of one, and equal sign on a base with a valence of two (Weber 1981). Bright (1984) counsels against using morpheme boundary symbols other than hyphen as a matter of course, suggesting rather that they be used only when distinctions between kinds of word formation are a focus of attention.

When deciding on a set of conventions for a particular analyzed text corpus, it is a good idea to first consult the literature on that particular language family. Often, the tradition of scholarship within a given language family has developed conventions which are uniquely suited to the character of the languages.

Figure 2.8 Conventions for marking morpheme breaks

-	inflectional prefix or suffix
=	derivational prefix or suffix
+	compound
a < b > c	infix <i>b</i> in root <i>ac</i>
a > b < c	circumfix <i>ac</i> around root <i>b</i>

Alternatives:

-	affix,	with = for clitic
-	suffix,	with = for prefix
-	valence 1,	with = for valence 2
*	trace, as in <i>a*c-b</i> (for root <i>ac</i> with infix <i>b</i> at position *)	

2.4 Aligning annotations

2.4.2 Allomorphic transcription

2.4.2.2 Handling discontinuous allomorphs

Discontinuous allomorphs are particularly troublesome in allomorphic transcription. The impression must not be given that the discontinuous parts are morphemes in their own right. Two conventions for handling this problem are presented.

A perennial problem in the presentation of analyzed text has been the treatment of discontinuous morphemes, such as infixes and circumfixes. Possible approaches to the problem are illustrated in figure 2.9. The example, from the Ilocano language of the Philippines, is a verb form consisting of three morphemes: a root, an infix, and a suffix. Our goal in making an allomorphic transcription of the verb form is to show the morphological structure without altering the phonological structure. To simply insert hyphens at the morpheme boundaries does not disrupt the phonological structure, but it does convey the incorrect impression that the verb consists of four morphemes. To solve this problem we need special symbols to distinguish infix or circumfix boundaries from prefix and suffix boundaries. The convention Lehmann (1982) proposes uses subscripted left and right corner characters for this purpose. Since these characters are not generally available in computers and printers, we propose using the < and > characters instead. This approach is illustrated in figures 2.8 and 2.9.

Figures 2.8 and 2.9 show another approach to making an allomorphic transcription involving discontinuous elements. This is to linearize the allomorphs while using a trace character (such as *) to show where an allomorph was broken by the intrusion of the allomorph immediately following it in the linearized string. That is, the trace symbol shows the position in the root where the infix is inserted, or the point in the circumfix at which it splits around the root. Though this departs from the basic definition for allomorphic transcription given above in that it reorders the phonemes, it is still allomorphic in two ways: (1) the spelling of each morpheme is the exact surface level variant used in that situation, and (2) the trace symbol further shows the fact that the allomorph contains an intrusion and provides sufficient information to reconstruct the original sequence of phonological elements if necessary.

The solution using angle brackets is better suited for a study of morphophonemic variation, for it keeps all the phonemes in their surface level environment. The solution using a trace character is better suited to the aligning annotation mechanism of *IT*, for it makes possible a one-to-one mapping from allomorphs to underlying morphemes when a morphemic representation is introduced.

Figure 2.9 Treatment of discontinuous allomorphs
(Ilocano, Philippines)

Consider the Ilocano form:

gumatangak 'I buy (something)'

Composed of:

<i>gatang</i>	'buy'
<i>-um-</i>	'actor focus'
<i>-ak</i>	'1st person singular'

Potential allomorphic transcriptions:

Inadequate	<i>g-um-atang-ak</i>
Adequate	<i>g <um> atang-ak</i> <i>g*atang-um-ak</i>

2.4 Aligning annotations

2.4.3 Morphemic representation

An aligning annotation can be used to convert surface level words into their underlying morphemic representation. Such a morphemic representation can then serve as the basis for morpheme-level annotation of the text.

2.4.3.1 The merits of morphemic representation versus allomorphic transcription

Morpheme-level annotation should be performed from a base of underlying morphemes rather than allomorphs. This will result in the annotations for a given morpheme occurring only once in the lexical mappings database, thus ensuring consistency of annotations.

For the sake of performing morpheme glossing, it is best to gloss from a morphemic representation rather than an allomorphic transcription. The reason for this is that since all the surface level variants of a morpheme are represented by a single underlying form, each morpheme occurs only once in the lexical database with all its glosses. If morpheme glossing is done from an allomorphic transcription, each underlying morpheme ends up with a separate lexical database entry for each allomorph. Not only does this mean a duplication of glosses, it also invites inconsistency, in that the same sense of meaning occurring with different allomorphs could be glossed with different synonyms, or the same gloss could be spelled or abbreviated in different ways.

The primary use of an allomorphic transcription is for making an analysis of the range and distribution of the allomorphs, and thereby discovering the morphophonemic processes at work. For this purpose, it is necessary to use two aligning annotations—first a mapping from the baseline text to an allomorphic transcription, and then a mapping from the allomorphs to the underlying morphemes. To find all the allomorphs of a particular morpheme, one would extract the allomorph-to-morpheme mapping to a standard format file (see section 5.13), then reverse and sort the list to produce a listing of morphemes with their allomorphs. To study the contexts in which

particular allomorphs occur and thus begin to deduce the morphophonemic processes at work, one would extract the base-line-word-to-allomorphic-transcription mapping to a standard format file, and use that list of allomorphic transcriptions for each unique word in the text corpus as a basis for concordance searches on environments for the variants.

Whereas the allomorphic transcription is a retranscription of the surface level phonological material which adds morpheme breaks, the treatment of the morphemic level is something much more abstract. For this reason, it is called a representation rather than a transcription. The morphemic representation should be a form of the text which represents each word as a string of underlying morphemes. When the study of morphophonology is an object, the morphemic representation should be mapped from an allomorphic transcription. Once the morphophonology is understood, the morphemic representation can be mapped directly from the baseline transcription.

The traditional approach to interlinear text glossing (such as exemplified in Lehmann 1982, for instance) uses a two-line model: an allomorphic transcription as the baseline text, with interlinear morpheme glosses underneath. Since the baseline text is simultaneously serving to represent the phonemic (or orthographic) structure of the text as spoken (or written) and its morphemic structure, an allomorphic transcription must be used. However, problems arise when there is a skewing between the phonemic and morphemic levels of the text. Some common linguistic phenomena in which this happens are: (1) morphophonemic alternation, (2) suppletion, (3) fusion, and (4) discontinuous morphemes. These are phenomena which are not adequately handled in the traditional two-line model of text glossing.³

IT solves this problem by introducing a multidimensional model of text. Rather than collapsing phonemic and morphemic structure into a single text line that tries to represent both, *IT* allows us to keep phonemic and morphemic structure separate by storing them in different fields, while at the same time keeping them related by means of the vertical alignment. No matter what is going on in the surface form of the text, we can always represent the underlying morphemes consistently. Not only does this separation of linguistic levels aid in linguistic analysis, it also makes possible the consistency of glossing which is the analyst's ideal. This is because there is only one version of the morpheme and its glosses in the lexical database. The next two modules show how *IT* handles the four problems listed above.

2.4 Aligning annotations

2.4.3 Morphemic representation

2.4.3.2 Handling alternation, suppletion, and fusion

Morphophonemic alternation, suppletion, and fusion are problems for the conventional two-level approach to interlinear text analysis. An n-level approach which introduces an abstract morphemic representation solves these problems.

Morphophonemic alternation occurs when a single morpheme takes on different phonological forms (including loss) in different environments. Consider the English privative prefix *in-*, as in *indefinite*. It assimilates to the point of articulation when prefixed to a bilabial, as in *impossible*. It assimilates to the manner of articulation when prefixed to liquids, as in *il-literate* and *ir-responsible*. Whereas the allomorphic transcription preserves the surface forms of the morphemes, the morphemic representation expresses their unity by using the single underlying form in all cases. Two examples from English are shown in figure 2.10, where underlying prefix *in-* appears as surface *ir-* preceding a base beginning with *r*, and underlying base *revere* appears in the surface without the final *e* due to the presence of suffix *-ent*.

Figure 2.10 Morphophonemic alternation and suppletion in the morphemic representation

```
\tx He   saw   the irreverent   children.
\mr he   see-d the in -revere-ent child-s
\mg 3s.M see-PST DEF PRIV-revere-ADJR child-PL
```

Key to field codes:

```
\tx   baseline text
\mr   morphemic representation
\mg   morpheme gloss
```

Key to abbreviations:

```
3s    3rd person singular
M     masculine
PST   past
DEF   definite
PRIV  privative
ADJR  adjectivalizer
PL    plural
```

Sometimes a morpheme alternation is not a phonological or orthographic variation explained by the context, but is based on a totally different form. This is called suppletion. Figure 2.10 shows two examples from English, where *saw* is a suppletive past tense form of *see*, and the *-ren* in *children* is treated as a suppletive form of the plural morpheme. There are at least two benefits of representing suppletions by the morpheme they replace: it explicitly shows that suppletion has taken place, and it means that the list of glosses for the morpheme occurs only once in the *IT* lexical database (thus ensuring consistency).

A special case of conditioned variation is fusion in which separate morphemes merge together in the surface representation. In portmanteau, a single surface level phoneme or morpheme simultaneously represents two underlying elements making it impossible to insert a morpheme boundary at the surface level. In contraction, which normally involves what could be represented as separate elements in careful speech, the two morphemes get fused by dropping out phonemes in the area of the boundary. These types of fusion are easily accounted for in the multidimensional model of text by putting the fully expanded underlying form in the morphemic representation. If the fusion is reducing what would be two words in the underlying form to a single word in the surface form, *IT* has no problem. In the mapping from baseline text form to morphemic representation, it is possible to specify multiword analyses. Figure 2.11 gives a sample of text from the To'aba'ita language (Solomon Islands) showing the expansion of a contraction in the morphemic representation field. One advantage of expanding fused forms is realized when doing morpheme glossing—the underlying morphemes and all their glosses are already in the gloss file.

Figure 2.11 Fusion in the morphemic representation
(To'aba'ita, Solomon Islands)

```
\tx 'Uria    taa noko    kwele?
\mr 'uri -a  taa na    'oko kwele
\mg about-it what REL you surprised

\ft Why are you surprised?
```

Key to field codes:

```
\tx    baseline text
\mr    morphemic representation
\mg    morpheme gloss
\ft    free translation
```

2.4 Aligning annotations

2.4.3 Morphemic representation

2.4.3.3 Handling discontinuous morphemes

Discontinuous morphemes have been a long standing problem for the conventional two-level approach to interlinear text analysis. *IT* handily solves this problem by introducing an intermediate morphemic representation which represents each word as a sequence of underlying morphemes.

A particularly knotty problem in the presentation of analyzed text has been the treatment of discontinuous morphemes, such as infixed roots or circumfixes. Figure 2.12 illustrates three approaches to glossing words with discontinuous morphemes. The first two are traditional approaches which gloss directly from the allomorphic transcription; the third illustrates the *IT* approach which uses an intermediate morphemic representation.

Figure 2.12a illustrates the most common approach to date as advocated by Lehmann (1982). Figures 2.8 and 2.9 have already shown the method of using < and > in the allomorphic transcription to indicate the discontinuous morpheme structure of a word. Figure 2.12a shows how the interlinear glosses are inserted. The gloss for the root (or circumfix) comes first followed by the gloss for the infix (or root). If this approach were used in *IT*, the whole infixed root (*g<um>atang* in the example) would have to be stored in the gloss file, with the combined gloss for both morphemes (*buy-AF* in the example).

Figure 2.12b illustrates the approach used in the *Studies in Philippine Languages* text series. It uses a different convention for glossing infixes—a very common phenomenon in Philippine languages. It marks the boundaries of an infix with the same symbol that marks off prefixes and suffixes. Special symbols in the gloss line are then used to show the discontinuous morpheme structure. The root gloss is placed under the first part of the root and underlined. The second part of the root is then glossed as underlined space. The underlining indicates the two parts of the word which make up the discontinuous morpheme. Of these two conventions, we prefer the former as it puts all the information about the morphemic structure in the text itself, rather than splitting it between the text and the gloss. The latter approach would not be appropriate for use with *IT* as it would store the two parts of the root morphemes separately in the lexical database.

Figure 2.12c illustrates the approach favored for use with *IT*. The nature of the discontinuity problem is that there is a skewing of linearity between the phonemic and morphemic structures of the text. The multidimensional nature of the *IT* text model lets us keep phonemic and morphemic structure separate by storing them in different fields. (We have the option of including or excluding the intermediate allomorphic transcription shown in the second line of the example.) The morphemic representation gives a simple sequence of the underlying morphemes which make up the form. This underlying form can not only recombine the morphemes that are discontinuous at the surface level, but can at the same time unravel any morphophonemic alternation by giving a single underlying form. The result is a greatly simplified task of morpheme glossing.

Figure 2.12 Infixation in the morphemic representation

(a) Lehmann's solution

```
g<um>atang-ak
buy-AF      -1s
```

(b) The *Studies in Philippine Languages* solution

```
g    -um- atang-ak
buy-AF- _____ -1s
```

(c) *IT*'s multidimensional solution

```
\tx  gumatangak
\at  g<um>atang-ak
\mr  gatang-um -ak
\mg  buy      -AF -1s

or,
\tx  gumatangak
\at  g*atang-um-ak
\mr  gatang  -um-ak
\mg  buy      -AF-1s
```

Key to field codes:

```
\tx  baseline text
\at  allomorphic transcription
\mr  morphemic representation
\mg  morpheme gloss
```

Key to abbreviations:

```
AF    actor focus
1s    first person singular
```


2.4 *Aligning annotations*

2.4.4 Word and morpheme glosses

An aligning annotation can be used to assign interlinear glosses to the words or morphemes of the text. Depending on one's purpose, one may adopt an informal or a formal style of glossing. The informal approach can be used in word or morpheme glossing. The formal approach requires morpheme by morpheme glossing.

A gloss is a brief, and admittedly rough, translation equivalent for a word or morpheme. They are used in text annotation as a means of identifying all the words and their parts. Consistency is important in text glossing. The same word or morpheme in the same sense of meaning should always have the same gloss. The abbreviations used in morpheme glosses should always be spelled and punctuated the same way. Fortunately, *IT*, with its built-in retrieval of previously defined glosses from the lexical mappings database, makes this easy to achieve.

2.4.4.1 Informal versus formal text glossing

The informal style of text glossing seeks to make a readable interlinear translation of the text. The formal style seeks to accurately identify all of the morphemes in the text. These differing aims lead to some basic differences in the way texts are glossed.

Before one can proceed with text glossing, a fundamental decision must be made. Are the glosses intended to serve as a literal translation which is meaningful in its own right, or are they intended to give a precise accounting for every element in the text? We refer to these, respectively, as the informal versus formal styles of text glossing. It is the intended audience which helps to determine which style would be appropriate. Of course, with *IT*'s multidimensional model of text, there would be nothing to prevent the analyst from including both.

The informal style of text glossing, which is primarily aimed at a lay audience, uses everyday words for glosses. The formal style, by

contrast, is aimed at a technical audience and freely uses technical terms for glosses as needed. Whereas the formal style would gloss grammatical functions with linguistic terminology, the informal style would seek to achieve the effect of the grammatical function through paraphrastic constructions in the glossing language. The formal style of text glossing is exemplified in most glossed text collections produced by linguists as a basis for grammatical analysis. The best-selling example of informal text glossing is probably Alfred Marshall's *Interlinear Greek-English New Testament* (1974).

Figure 2.13 summarizes the contrast between the informal and formal styles of text glossing. The next module covers word glossing. The modules after that cover the methods for glossing each of the types of morphemes listed in figure 2.13.

Figure 2.13 Informal versus formal text glossing

	<i>Informal style</i>	<i>Formal style</i>
Purpose	Translation for layman	Identification for specialist
Content morphemes	Brief translation (with more discrimination of senses)	Brief translation (with more focus on commonality of form)
Functor morphemes	Brief translation (conforming to target language grammar)	Technical abbreviation (all caps; period forms compounds)
Untranslatable or redundant elements	Omit gloss	Gloss required
Unknown morphemes	Unique symbol (for example, ?)	Unique symbol (for example, ?)

2.4 Aligning annotations

2.4.4 Word and morpheme glosses

2.4.4.2 Word glossing

Word glossing gives a rough translation for each word in the text. It is a useful adjunct to morpheme glossing when morphology is complex. With simple morphologies, it can be a substitute for informal morpheme glossing.

In word glossing, the analyst gives a brief translation equivalent for each word of the text. It is, by definition, a form of informal glossing. There is no place in word glosses for the abbreviations of grammatical terms which is characteristic of formal glossing.

A major application of word glosses is as a supplement to morpheme glosses for languages which exhibit complex morphology. For instance, when a verb form includes seven affixes it is a substantial mental exercise for a reader to assemble the meaning of the whole word from the individual morpheme glosses. A word gloss makes this easy. The use of word glossing for a morphologically complex language is illustrated in figure 2.14 which is the opening sentence of a text in the Eskimo language of northwest Alaska. When setting up the interlinear text model (see sections 3.2 and 6.5.3) for both morpheme glossing and word glossing, it is necessary to have two different source fields for the mappings involved. Word glossing must map from a field which is defined to have no morpheme separators; this would normally be the baseline text. Morpheme glossing must map from a field which does have morpheme separators; this would normally be the morphemic representation.

Word glosses may also serve as a replacement for informal morpheme glosses where the language being described exhibits a fairly simple morphology. For instance, for a language in which the only noun morphology consists of possessive suffixes on some words, one might as well use word glossing, such as 'her hand', as opposed to informal morpheme glossing which would give 'hand-hers'. Figure 2.15 illustrates such an application of word glossing. The example is a verse from Marshall's *Interlinear Greek-English New Testament* (1974). See the next three modules for a discussion of techniques for composing the glosses.

Figure 2.14 Word glossing as a supplement to morpheme glossing
(Inupiat, Alaska)

```
\tx Akutchilighmik-uvva          uqaagtullangniaqtunga.
\at akut      -chi-ligh-mik  -uvva uqaagtu  -llang-niaq-tunga
\mr akutuq    -si -liq -mik  -uvva uqaagtuq -llak -niaq-tunga
\mg icecream-RSL-GER -s.MOD-now tell story-DUR -INT -ls.I
\wg about making Eskimo icecream I am going to tell a story
```

Key to field codes:

\tx	baseline text
\at	allomorphic transcription
\mr	morphemic representation
\mg	morpheme glosses
\wg	word glosses

Key to abbreviations:

RSL	resultative
GER	gerund
s	singular
MOD	modalis case
DUR	durative
INT	intensive
ls	first person singular
I	indicative

Figure 2.15 Word glossing as interlinear translation
(Koine Greek)

```
\ref John 2 v 2
\tx Kai th      'hmera th trith gamos      egeneto
\wg And on the  day  --  third a wedding there was

\tx en Kana ths Galilaias, ...
\wg in Cana --  of Galilee, ...
```

Key to transcription:

h	eta
'	rough breathing

2.4 Aligning annotations

2.4.4 Word and morpheme glosses

2.4.4.3 Glossing content morphemes

Content morphemes are treated similarly in informal and formal glossing, a brief translation equivalent being used in both cases. A difference may emerge in the treatment of multiple senses of meaning, where the informal approach tends to discriminate senses while the formal approach tends to highlight unity of form.

The approach for glossing content (open class) morphemes is essentially the same for the informal and formal styles. A brief (one word, if possible) nearest translation equivalent is used. Remember that the gloss is not a definition of the morpheme; it is simply a brief reminder to the reader of what it means. The reader should always be referred to a dictionary for the full definition.

The gloss is entered in lower case letters. Normally, sentence level capitalization and punctuation are omitted from the glossing. Thus, the gloss for the first morpheme of a sentence is not capitalized, nor does the gloss for the last morpheme include the sentence final punctuation. An exception to this rule is when word glossing is used for the purpose of producing an interlinear translation which is readable in its own right, as in the example of figure 2.15. In every case, capitalization should appear in the glosses to conform to morpheme-level conventions of the glossing language. For instance, when glossing in English, all glosses that are proper names would be capitalized. When glossing in German, all glosses that are nouns would be capitalized.

There is not a great difference between the informal and formal styles when it comes to glossing content morphemes. The key difference that might arise is in one's treatment of multiple senses. In the informal style, one would be more likely to have different glosses for each of the different senses. This would result in an interlinear translation that is easier to read and understand. In the formal style, one would be more likely to use fewer glosses to emphasize the commonality of form and thus give a glimpse of the ethnosemantic world view. For instance, the To'aba'ita word *kafo* has a primary sense of 'fresh water' with a secondary sense of 'river'. Informal glossing would be likely to gloss *kafo* as 'water' or 'river' depending on context, whereas formal glossing might use 'water' everywhere.

2.4.4.4 Handling untranslatable and unknown elements

In informal glossing, elements which do not contribute to the translation into the glossing language may be left unglossed. In formal glossing, this is never permissible; every morpheme must be identified. In either style of glossing, unknown elements should always be glossed as such; we recommend ? as a standard gloss for this purpose.

The treatment in informal glossing of elements which do not contribute to the translation was illustrated in figure 2.15. Since the aim of informal word glossing is to produce a readable interlinear translation of the text, it is often best to leave some elements unglossed. Two prime candidates for such treatment are elements which are redundant and do not contribute additional information to the interlinear translation, or functor morphemes which have subtle meanings so foreign to the glossing language that they cannot be briefly glossed. When an item is not glossed, it should not be left blank since a blank spot signals to *IT* an item that has not yet been dealt with. Rather, use a character like hyphen or dash or underscore to indicate an item that is purposely unglossed.⁴ This device should not be used in formal glossing because there the aim is to identify every morpheme in the text.

The investigator does not always know the gloss for a morpheme, especially in the early stages of analysis. These unknown glosses must be treated differently than the deliberately unglossed elements. Whether using an informal or a formal style, it is imperative that unknown glosses be explicitly marked as such by means of a unique symbol. We propose the question mark as a standard symbol for this purpose. Using an explicit symbol has two advantages over leaving the gloss blank: (1) it makes it clear to the reader that you have not simply overlooked a gloss or are not treating it as untranslatable, and (2) it gives a unique symbol which can be searched for (both in texts and in the lexical database) in order to find all unknown morphemes which need further analysis work. Furthermore, *IT* views blank glosses as unglossed items and will stop and ask for a gloss every time it encounters one.

2.4 *Aligning annotations*

2.4.4 *Word and morpheme glosses*

2.4.4.5 **Glossing functor morphemes**

The primary difference between the informal and formal styles of morpheme glossing appears in the treatment of functor (closed class) morphemes. The informal approach uses translations into the glossing language, while the formal approach uses abbreviations of technical terms.

In the informal style of morpheme glossing, brief (non-technical) nearest translation equivalents are used for the functor morphemes just as they are for the content morphemes. This often involves paraphrastic constructions using the nearest equivalent functors in the glossing language, such as ‘to town’ for the dative form of a noun, or ‘had seen’ for the perfect form of a verb.

The formal style uses abbreviations of technical terms for grammatical functions. These abbreviations are in all upper case letters and do not include a terminating period. For instance, one might use ‘HAB’ rather than ‘always’ as the gloss for a habitual verb aspect, or ‘DEF’ rather than ‘the’ in glossing a definite article. The use of upper case abbreviations to gloss functor morphemes is a fairly recent practice among linguists but it has gained widespread acceptance and can now be considered a standard.⁵ There is, unfortunately, no definitive source of standard terms and abbreviations for text glossing.⁶ Rather, the standard practice is for each investigator to devise his or her own set of abbreviations and to provide a complete listing of them in an introduction to the text collection or grammar sketch.

In the informal style, it is not always possible to come up with a brief gloss consisting of an everyday word. In such cases, it may be necessary to resort to abbreviated technical terms. Abbreviations in the informal style should follow the normal orthographic conventions for the glossing language. For English, this would mean that the abbreviations should be in lower case letters and terminated by a period. For instance, the informal style abbreviation for ‘habitual’ might be ‘habit.’ as opposed to ‘HAB’ for the formal style.

One exception to the general rule of all upper case abbreviations in formal glossing is normally followed. This is in the glossing of pronouns. The convention for pronoun glosses combines a digit which

designates the person and a lower case abbreviation for the number. For instance, '3sg' or '3s' for third person singular. In glossing pronouns in the informal style, pronouns would be used as glosses. When English is the glossing language, this would mean that 'he', 'she', and 'it' would all be potential glosses, in place of the single formal gloss '3sg'. Including uses in oblique cases adds 'him', 'her', 'to him', 'to her', 'to it', 'his', 'hers', and 'its' to the list of informal glosses.

The convention for abbreviating formal glosses which combine more than one category (except person and number as noted above) is to abbreviate the component terms individually and separate them with a period. For instance, 'PRES.PROG' could be used to gloss a present progressive morpheme and '3sg.POS' a third person singular possessive morpheme.

Figure 2.16 Styles of morpheme glossing
(Lau, Solomon Islands)

(a) Informal style

```
\tx Ma  na  kui baa ka haea  laugo  fuada, ...
\ma ma  na  kui baa ka hae-a  lau-  go  fua-da
\mg and the dog that it say-it again-?  for-them
```

(b) Formal style

```
\tx Ma  na  kui baa ka      haea      laugo      fuada, ...
\ma ma  na  kui baa ka      hae-a      lau-  go  fua-da
\mg and DEF dog DEM 3sg.SER say-3sg.OBJ again-? BEN-3pl.OBJ
```

Key to field codes:

```
\tx  baseline text
\ma  morpheme analysis
\mg  morpheme gloss
```

Key to abbreviations:

```
3sg  third person singular
3pl  third person plural
DEF  definite
DEM  demonstrative
SER  serial subject marker
OBJ  object
BEN  benefactive
```


2.4 Aligning annotations

2.4.5 Lexemic representation

An aligning annotation can be used to produce a lexemic representation of the text which can unite the multiple morphemes that make up a lexeme into a single unit. This makes it possible to gloss (and otherwise annotate) lexemes as distinct from the words or morphemes which make them up.

A common difficulty for interlinear glossing is a skewing between the morphemic and lexemic levels. When more than one morpheme combine to form a single lexeme, morpheme by morpheme glossing misses the point. For instance, the English idiom *off his rocker* is a single lexeme conveying the notion of 'be crazy,' and needs to be glossed as a single unit. If the morphemes involved are in the same word, word glossing provides a partial solution, though it fails to indicate explicitly the lexemic structure. But as in the example above, multimorphemic lexemes often involve multiple words and a more general solution is required.

The *IT* solution is to introduce an annotation which maps the morphemic representation to a lexemic representation, and then to gloss from the lexemic level rather than the morphemic. (For some applications, it would be possible to use a lexemic representation alone, in place of a morphemic representation.) The whole lexeme (as the composition of its underlying morphemes) should be entered as the annotation of its head morpheme, and then the other morphemes must be annotated with a place holder which shows that the morpheme is to be found elsewhere in the lexemic representation.

For conventions, we suggest using + to combine the morphemes of the lexeme, > to indicate that a morpheme forms a lexeme with a head morpheme which follows, and < to indicate that a morpheme forms a lexeme with a head morpheme which precedes. For instance, figure 2.17 gives an example showing how the clause final separable prefix of German verbs can be reunited with their verb roots. Note that this approach works for discontinuous lexemes just as well as for contiguous ones.

**Figure 2.17 Lexemic representation and discontinuous lexemes
(German)**

```
\tx Er      machte      still die      Schachtel auf.
\mr er      mach      -te    still die      Schachtel auf
\lr er      auf+mach-te    still die      Schachtel <
\lg 3s.M.NM open      -3s.PST quiet DEF.F.SG.AC box      <
```

```
\ft He opened the box quietly.
```

Key to field codes:

```
\tx  text
\mr  morphemic representation
\lr  lexemic representation
\lg  lexeme glosses
\ft  free translation
```

Key to abbreviations:

```
3s   third person singular
M    masculine
F    feminine
NM   nominative case
AC   accusative case
PST  past tense
DEF  definite
SG   singular
```

2.4 *Aligning annotations*

2.4.6 Grammatical analysis

Aligning annotations can be used to record many aspects of the grammatical analysis of a text, such as labeling of categories, bracketing of constituent structures, and tagging of functions.

As a basis for studying the grammatical patterns by which words and higher level constructions are formed, one may make interlinear annotations in a text corpus to label the categories of the minimal forms and the constructions, to show how smaller forms combine to make larger forms, and to tag the constituents with a label for their function in the construction. We now consider these three applications of aligning annotations in the next three modules.

2.4.6.1 Labeling grammatical categories

Grammatical constituents at any level can be labeled as to category (or form class). We propose a convention for constructing category labels consisting of base category (or function descriptor), level indicator, and optional subcategory.

Our systems for naming grammatical categories are essentially based on the word level notion of “parts of speech.” The tradition began with the ancient Greek grammarian Dionysius Thrax who defined eight parts of speech in Greek grammar: noun, pronoun, article, participle, verb, adverb, preposition, and conjunction. The Latin grammarians added interjection and deleted article. In traditional English grade-school grammar, a further change has been made: the addition of adjective and the dropping of participle. These traditional category names are the base from which most linguists work when seeking to define and name the relevant word categories for a particular language. Of course, new categories are added as needed, and traditional categories which are not relevant are ignored. Furthermore, names for categories above and below the word are devised.

In text annotation, abbreviations should be used for grammatical categories. Figure 2.18 summarizes the conventions we propose for forming such abbreviations. To distinguish category labels from glosses we suggest a different convention, namely, that category ab-

breviations capitalize only the first letter of each word in the category name. For instance, the following abbreviations could be used for some of the basic categories: *N*, *Pron*, *V*, *Adv*, *Prep*, and so on.

For categories above and below the word level, two-part names are generally used. The first part specifies a base category or a function descriptor, and the second part is an abbreviation for the level of the constituent. The first letter of the second part should also be capitalized and appended to the first part without intervening space. Standard abbreviations for levels above the word are *P* for "phrase," *Cl* for "clause," and *S* for "sentence." Examples of full category labels are *NP* for "noun phrase" and *RelCl* for "relative clause." For morpheme level categories, *St* for "stem" or *Rt* for "root" can be appended for base classes. For affix classes, labels like *Prf* (prefix), *Suf* (suffix), or *Inf* (infix) can be appended to an abbreviation characterizing the function of the class; for instance, *ObjSuf* for "object suffix."

Large categories are often subcategorized. For instance, nouns are often subcategorized for gender and verbs for transitivity. For this we recommend a convention that the subcategory name be abbreviated in all lower case, and that it be set off from the base category name by a colon;⁷ for instance, *VRt:tr* for a transitive verb root.

Figure 2.18 Conventions for forming category labels

A fully expanded category label has three parts:

1. Base category (such as *N* "noun" or *Adj* "adjective") or functional modifier (such as *Rel* "relative" or *Tns* "tense"). First letter capitalized.
2. Level, if other than word (such as *Cl* "clause", *P* "phrase", *St* "stem", *Rt* "root", *Prf* "prefix", or *Suf* "suffix"). First letter capitalized; no space preceding.
3. Subcategory (such as *tr* "transitive" or *f* "feminine"). No capitalization; colon preceding.

For example,

<i>N</i>	noun
<i>N:f</i>	feminine noun
<i>NRt</i>	noun root
<i>NRt:f</i>	feminine noun root
<i>TnsSuf</i>	tense suffix
<i>AdjP</i>	adjective phrase
<i>RelCl</i>	relative clause

2.4 Aligning annotations

2.4.6 Grammatical analysis

2.4.6.2 Identifying word and morpheme categories

A first step in grammatical analysis is to identify word or morpheme categories, or both. An aligning annotation can be used to build a lexical database which saves the analyst's decisions on word and morpheme categories, thus making it possible to insert category labels into a text semi-automatically.

Figure 2.19, adapted from Trail (1970:169), illustrates the annotation of text by morpheme categories (in the `\mc` field) and word categories (in the `\wc` field). To have both morpheme and word category annotations in the same text, it is necessary to have separate base fields for the annotations—a text field based on words (`\tx` in the example) for the word categories, and a text field based on morphemes (`\at` in the example) for the morpheme categories. (Sections 3.2 and 6.5.3 go into this more deeply.)

In traditional two-level text glossing, glosses and categories have commonly been mixed together. For instance, it is not unusual to see a gloss like *ASP* for “aspect”. But this is not a gloss for it does not say what the morpheme means. Rather, it identifies a general function category. Glosses should identify the specific meaning, like durative, or perfective, or potential, or whatever the particular aspect might be. Properly speaking, *Asp* would best serve as an abbreviation for a function label (see the next module). To make a category label, it needs a modifier for the level of the constituent. For instance, the category for a class of aspectual suffixes would be labeled something like *AspSuf*. There is one special case in which it may be difficult and not always necessary to keep labels for gloss, category, and function separate. That is the case in which a form class has only one member. However, even if the same labels are used, the different capitalization conventions should still be observed.

The following guidelines may be helpful in keeping labels for glosses, categories, and functions separate. A gloss label answers the question, “What does the particular morpheme mean?” A category label answers the question, “What class of forms is this form a member of?” A function label answers the question, “What function is the form filling within the construction of which it is a part?”

Figure 2.19 Morpheme and word categories
(Lamani, India)

```

\tx Ar  ek  baaper          di  beTaa    ra          cha.
\at ar  ek  baap -e      -r  di  beT-aa    r  -a      cha
\mg and one father-OB.SG-POSS two son-NM.PL be -POT  PRES
\mc Cnj Num NSt  -CsSuf-Rel2 Num NSt-CsSuf VSt-AspSuf Tns
\wc Cnj Num N:possv      Num N          V:st      Aux

```

```

\ft Now once there was a father who had two sons.

```

Key to field codes:

```

\tx  text
\at  allomorphic transcription
\mg  morpheme glosses
\mc  morpheme categories
\wc  word categories
\ft  free translation

```

Key to abbreviations:

```

OB    oblique case
NM    nominative case
SG    singular
PL    plural
POSS  possessor
POT    potential aspect
PRES  present tense

Cnj    conjunction
Num    number
NSt    noun stem
CsSuf  case-number suffix
Rel2   relator, class 2
VSt    verb stem
AspSuf aspect suffix
Tns    tense particle

N      noun
possv  possessive
V      Verb
st     stative
Aux    auxiliary

```

2.4 Aligning annotations

2.4.6 Grammatical analysis

2.4.6.3 Bracketing syntactic structures

A second step in grammatical analysis is to specify how lower level forms combine to form higher level ones. An aligning annotation can be used to indicate the hierarchical constituent structure of a text unit by means of nested pairs of matching brackets.

One possible way of specifying the syntactic structure of a text unit would be to define fields of annotation for given levels of linguistic structure. For instance, a phrase field could align the category label for a phrase beneath the first word of the phrase. Words which were not part of any phrase would have to be annotated in a special way so that they would not be misinterpreted as belonging to the preceding phrase. Similarly, there could be a clause field, and whatever other levels might be needed. This approach will work as long as the levels do not embed within themselves—for instance, as long as phrases do not occur within phrases.

Since embedding is the rule rather than the exception, we recommend that the constituent structure be indicated by means of bracketing. The first word of a construction would have the category label followed by an opening bracket underneath it. A closing bracket would then be placed under the last word in the construction. This is illustrated in figure 2.20. Note in that example that the last word of the sentence is simultaneously terminating four constructions: the entire sentence, the relative clause, the sentence embedded in the relative clause, and the object noun phrase of that embedded sentence.

Figure 2.20 Syntactic bracketing
(To'aba'ita, Solomon Islands)

```

\tx Imole 'e 'oro na      kera 'uria bada
\wg people are many who  they resemble themselves
\wc N      SM  V:st Rltr  Pron  V:tr  ReflPron
\sb S[NP[] VP[ ] RelCl[ S[NP[] VP[ ]

\tx nga wane baa.
\wg the man that
\wc Art N    Dem
\sb NP[      ]]]]

\ft There are many people who resemble that man.

```

Key to field codes:

```

\tx  baseline text
\wg  word gloss
\wc  word category
\sb  syntactic bracketing
\ft  free translation

```

Key to abbreviations:

```

N      noun
SM     subject marker
V      verb
st     stative
tr     transitive
Rltr   relator
Pron   pronoun
Refl   reflexive
Art    article
Dem    demonstrative

S      sentence
NP     noun phrase
VP     verb phrase
RelCl  relative clause

```


2.4 Aligning annotations

2.4.6 Grammatical analysis

2.4.6.4 Tagging grammatical functions

Many approaches to linguistics take grammatical analysis a step further by not only identifying form classes, but also identifying the functions constituents play in the constructions of which they are a part.

A common tradition in linguistic analysis is to account for linguistic forms in context by identifying the function of each form. Most widely known for this approach are probably the tagmemic (Pike 1967) and systemic (Halliday 1985) schools. Other recent linguistic models, such as Functional Unification Grammar (Kay 1979) and Lexical Functional Grammar (Kaplan and Bresnan 1982), account for form and function relationships as well.

Figure 2.21 gives an example of a typical tagmemic style of analysis. In that example, the formal analysis (in the $\backslash fo$ field) identifies the categories (or form classes) of the words. It also identifies the categories of the higher level constituents, using brackets as described in the preceding module to show the constituent structure. In the function analysis (in the $\backslash fu$ field), an abbreviation for the grammatical function of each word and construction is aligned beneath its form class label. Note that the brackets indicating constituent structure are carried down into the function analysis. The vertical alignment serves the same purpose as the : (manifested by) operator used in conventional tagmemic descriptions. For instance, the first elements aligned in the example illustrate the $S:NP$, or subject manifested by noun phrase, relation. The four-cell tagmeme approach (Pike and Pike 1977) can be handled in a similar fashion; simply define one aligning annotation for each of the four cells.

Figure 2.21 Form and function analysis

```

\tx The      timid boy saw three  bullies in      the   park.
\fo NP[Art Adj  N] V  NP[Num N]      PP[Prep NP[Art N]]
\fu S[Spec Qual H] P  O[Qnt H]      Loc[Rlr H[Spec H]]

```

Key to field codes:

```

\tx    text
\fo    form classes
\fu    functions

```

Key to abbreviations:

```

NP      noun phrase
Art     article
Adj     adjective
Num     number
N       noun
V       verb
PP      prepositional phrase
Prep    preposition

S       subject
P       predicate
O       object
Loc     location
Spec    specificity
Qual    quality
Qnt     quantity
H       head
Rlr     relator

```

2.4 Aligning annotations

2.4.7 Semantic notions

Aligning annotations can be used to record many aspects of the semantic analysis of a text, such as case roles, semantic subcategorization, and participant indexing.

In this section we briefly illustrate some possible annotations of a semantic nature: identification of semantic role in case grammar analysis, semantic subcategorization of verbs and nouns, and indexing of participants in discourse. These examples are suggestive of the wide range of applications for aligning annotations. The imaginative analyst should be able to devise many others as need arises in the course of text analysis.

2.4.7.1 Semantic case roles

The semantic case role of each nominal in a text can be identified in an aligning annotation. Adding the root form of each verb and paired brackets to mark the bounds of each proposition results in an analysis of the text into propositional case frames.

The grammar of case has come to be widely recognized as an important component of clause level grammar. It was Charles Fillmore's (1968) seminal study, *The Case for Case*, which popularized the idea that there are a dozen or so semantic roles which account for the systematic meaning relationships between predicates and their nominal adjuncts in all languages. Since that time, many investigators have developed their version of a universal semantic role inventory. Some resources for finding such inventories of case systems are Anderson (1971), Grimes (1975:116-138), Starosta (1978), Givón (1984:87-89,126-133), and Longacre (1976:23-97). The example below follows the scheme of the latter author.

Figure 2.22 illustrates the use of an aligning annotation to record semantic case roles. The example comprises two sentences, but four propositions. Opening and closing square brackets indicate the boundaries of each proposition. Aligned with each nominal is an abbreviation for the semantic role it plays in the proposition. Aligned with each verb is its root form. What occurs between matching

brackets amounts to a case frame. By nesting brackets it is possible to sort out the multiple semantic roles which a single nominal may be playing. For instance, in the example of figure 2.22, *key* takes part in three propositions. It is simultaneously the instrument of *open*, the patient of *lie*, and the patient of *find*. When brackets are nested like this, a case frame includes none of the material in nested propositions.

A full-scale analysis of case grammar is likely to require more than the single annotation exemplified in figure 2.22. Most authors agree that it is important to distinguish underlying semantic roles from surface level case marking systems. This suggests the use of a pair of annotations. The lexicase model (Starosta 1978, Chen 1982) goes even further, proposing a level of universal *case forms*, intermediate between underlying *case relations* and surface *case marking*. Givón (1984:135-138) accomplishes a similar thing when he observes that case marking systems simultaneously encode a nominal for *semantic case-role* (such as agent, patient, and so on) and *pragmatic case-role* (such as subject, object, indirect object, and so on). The multidimensional nature of IT would be aptly suited to any of these varieties of analysis.

Figure 2.22 Semantic case roles

```
\tx John came to a door. He opened it with a key he
\sr [A come G] [A open P I[P[P A

\tx found lying on the floor.
\sr find] lie L]]
```

Key to field codes:

```
\tx text
\sr semantic roles
```

Key to abbreviations:

```
A agent
G goal
I instrument
L location
P patient
```

2.4 Aligning annotations

2.4.7 Semantic notions

2.4.7.2 Semantic subcategorization

Correctly identifying the case role of a nominal or the kind of information which a proposition contributes to the discourse requires knowing the semantic subcategorization of the nouns and verbs. This information can be expressed in an aligning annotation.

The semantic subcategorization of verbs and nouns has a significant part in the analysis of semantic roles, as well as in other aspects of discourse analysis. For instance, assigning the semantic role to the subject in *John (Agent) opened the door* versus *The key (Instrument) opened the door*, is a matter of the subcategorization of the subject nouns since everything else in the two propositions is the same. A complete description of the case grammar of a language would have to define how these and other factors interact to determine semantic roles.⁸

A similar problem is in the analysis of kinds of information in discourse (Grimes 1975:33-100). As a general rule, simple past tense verbs mark the event line in English narrative, while other tenses and aspects function in giving settings, background, flashbacks, evaluations, and so on (Longacre 1983:14-17). However, sentences like *The road forked* and *The machinery hummed in the next room* do not encode events. The subcategorization of the subject nouns as inanimate and of the verbs as describing states rather than actions overrides the rule of thumb based on tense and aspect alone. An adequate rule for determining event line material must take into account subcategorization of nouns and verbs as well.

Figure 2.23 illustrates the semantic subcategorization of verbs and nouns by adding another annotation to the example of figure 2.22. The nouns are subcategorized according to a scheme proposed by Givón (1984:56). This scheme defines a scale of classes, each of which is more restricted than the preceding: *entity* (or *abstract*) "that which exists," *temporal* (or *semi-abstract*) "exists at a particular time," *concrete* "exists in both time and place," *animate* "a living organism," and *human*. Givón (1984:85-126) also gives a classification of verbs. For this example, however, we use the scheme of Longacre (1976:40-81). He defines the following main classes based on patterns of case

frames: *ambient*, *ambient experiential*, *experiential*, *factual knowledge*, *cognition*, *sensation*, *physical*, *measure*, *locative*, *motion*, and *possession*. In the example, we were not able to identify the class for the word *find* with certainty, and thus make use of question mark—the marker which should always be used for an explicit tagging of unknown residue. Another scheme for subcategorizing verbs, which Longacre actually uses as an intersecting dimension to the above list of classes, is the fourfold classification developed by Chafe (1970). It classifies a proposition as describing a *state*, a *process* (a change of state), an *action* (an event with a responsible agent), or an *action process* (an action which also causes a patient to change state). Note in the example that the verbal subcategories are expressed in all caps to distinguish them at a glance from the nominal subcategories.

Figure 2.23 Verb and noun subcategorization

```

\tx John came to a door. He opened it with a key he
\sc Hum MOT Con Hum PHY Con Con Hum
\sr [A come G] [A open P I[P[P A

\tx found lying on the floor.
\sc ? LOC Con
\sr find] lie L]]

```

Key to field codes:

```

\tx text
\sc subcategories
\sr semantic roles

```

Key to abbreviations:

```

Hum human noun
Con concrete noun
MOT motion verb
PHY physical verb
LOC locative verb

```

2.4 Aligning annotations

2.4.7 Semantic notions

2.4.7.3 Participant indexing

The cohesion component in discourse helps the hearer or reader keep track of who's who. A text corpus for studying cohesion can be built by using an aligning annotation to assign a participant index to each reference to a participant.

A well-formed discourse helps its hearers or readers keep track of who's who as it unfolds. This is a major part of what many authors call the cohesion component of discourse (for instance, Grimes 1975:113,316-319). This involves such phenomena in grammar as anaphora and cataphora in pronouns and deictics, switch reference systems, ellipsis, reflexives, participant orientation in embedded (directly quoted) discourse, and choice in a referential hierarchy from descriptive noun phrase to minimal noun phrase to pronoun to affix to zero marking.

To gain an understanding of how all these things work in a particular language, one needs to work from a text corpus in which each reference to a participant is explicitly tagged by an identifier for the participant. This is essentially the notion of referential indices which has been employed in the transformational-generative school of linguistics. The general understanding of that mechanism is that each referent in a discourse is assigned a unique index number which serves to equate references to the same person or object and distinguish references to different persons or objects.

This is illustrated in figure 2.24. In that example, we have taken part of the story of *Hansel and Gretel* and supplied participant indices in an interlinear annotation. We have called them participant indices (rather than referential indices) for two reasons. (1) Only main participants (and not all referents) are indexed. A good rule-of-thumb is to index only those referents which recur and are thus involved in some of the cohesion phenomena listed above. Indexing all referents would only yield interfering clutter. (2) Rather than numbers, mnemonic symbols have been used.

Figure 2.24 Participant indexing in discourse

\tx Gretel rubbed her stomach. "Hansel, let's have a feast.
 \pi G G H H+G
 \tx I'll begin with the roof, and you start on the porch."
 \pi G H
 \tx The starving children began to nibble on the house.
 \pi H+G
 \tx Suddenly the door opened and an old woman hobbled out.
 \pi W
 \tx "Hello, dear children!" she cackled. "What are you
 \pi H+G W H+G
 \tx doing here?" The old woman, who was really a witch,
 \pi W W W
 \tx pulled them into the house and locked Hansel in a cage.
 \pi H+G H

Key to field codes:

\tx	text
\pi	participant indices

Key to abbreviations:

H	Hansel
G	Gretel
H + G	Hansel and Gretel
W	the witch

2.4 Aligning annotations

2.4.8 Computer-assisted dialect adaptation

Aligning annotations can be used to adapt a baseline text in one dialect to versions of the text in other dialects.

While machine translation has remained a rather elusive goal, David Weber and William Mann (1979) demonstrated that machine-assisted translation of texts between closely related dialects is much easier to attain. They called the approach computer-assisted dialect adaptation, or CADA. Subsequent field implementations of their method have shown that it produces useful results with small computers that can be operated in the field (Weber and Kasper 1986, Reed 1985).

CADA works by substituting forms in one dialect with corresponding forms in another dialect. Because the dialects in question are closely related, the source and target dialects have virtually the same underlying grammars and semantics. The bulk of the differences amount to changes in the spellings of words due to systematic sound changes and replacements of morphemic forms.

In the Quechua dialects, for which Weber and Mann first implemented CADA, most of the complexity of the grammar is in the morphology. Thus their program was able to make more than 90% of the changes needed for translation between dialects by ignoring syntax and working only at the word level. It breaks words into underlying morphemes, makes form substitutions, and performs morphophonemic changes to synthesize the surface form for the target dialect. Their program has a built-in morphological analyzer which uses a user-supplied dictionary of morphemes to do the analysis automatically. When the analysis results in ambiguity, the program generates multiple forms among which an operator must choose. A drawback of the current implementations is that the morphophonemic component is hardcoded in the C programming language, and thus the programs cannot be applied to other languages without a substantial programming effort.

2.4.8.1 Adapting directly from one dialect to another

Text can be adapted from one dialect to another by using an aligning annotation which maps the words or morphemes of one dialect directly to corresponding words or morphemes of the other.

The problem of adapting text from one dialect to another can be viewed as a problem in text annotation. That is, a baseline text in one dialect can be annotated with equivalents in another dialect. With this approach, *IT* can be used as an off-the-shelf tool for semi-automatic computer-assisted dialect adaptation for many applications. *IT* is well-suited to applications in which word or morpheme substitutions must be made. Figure 2.25 shows an example where word substitutions were used. (See the next module for an explanation of the treatment of punctuation.)

Where morpheme substitutions must be made, *IT* will not perform automatic morpheme analysis. Rather, the user must annotate the

Figure 2.25 Adaptation of one dialect to another
(To'aba'ita and Kwara'ae, Solomon Islands)

```
\ref KUKUA s 1
\tob Ro'o   nau ku too .
\kwr ro'oki nau ku tua .

\ref KUKUA s 2
\tob 'I u'usuwadia nau ku ngalia kafo .
\kwr 'i afodangi  nau ku ngalia kafo .

\ref KUKUA s 3
\tob Aia   , ngalia kafo sui , nau ku oli mai .
\kwr aia'a , ngalia kafo sui , nau ku oli ma'i .

\ref KUKUA s 4
\tob Oli mai kwa fula , nau ku 'akwasia kafo 'ila fera .
\kwr oli ma'i ku dao , nau ku fa'asia kafo saena luma .
```

Key to field codes:

\ref	reference field
\tob	baseline text in To'aba'ita
\kwr	adaptation to Kwara'ae

baseline text into a morphemic representation as discussed earlier in section 2.4. Then *IT* can perform the morpheme substitutions automatically, except in cases where *IT* must get user input to choose among alternatives.

IT is not well-suited to applications in which differences in syntax require numerous reorderings of elements. It is possible to handle occasional reorderings, however, using a technique such as illustrated in figure 2.18 for relocating an element in an annotation. In that case the relocated element is joined with an element that is already there. If it is necessary to move a word to a different position where it will stand as a word by itself, *IT* requires that there be some kind of place holder in the baseline text to define a column into which the displaced word can be inserted. Use a unique, arbitrary symbol (like *, for instance) in the base line to define the column. It is possible to make this kind of change in the base line with the edit mode of *itp*.

2.4 Aligning annotations

2.4.8 Computer-assisted dialect adaptation

2.4.8.2 Adapting from an intermediate form to many dialects

When the aim is to adapt a text into many dialects, greater efficiency and consistency are achieved when the source dialect is mapped onto an analyzed, unambiguous, intermediate form and the target dialects mapped from that intermediate form.

If the intention is to adapt a single text into many dialects, then going directly from dialect to dialect has a significant disadvantage. The decisions that must be made as the baseline text is analyzed and disambiguated must be made over again for each target dialect. Greater efficiency and greater consistency are achieved if the decision making process is mapped into an analyzed, unambiguous, intermediate form. This intermediate form can in turn be mapped straightforwardly into the many target dialects. This is the approach used in the Quechua CADA program. In that case, the designers chose to use reconstructed Proto Quechua as the intermediate form.

Another option is to use glosses as the intermediate form. An obvious advantage of this approach is that it is not necessary to reconstruct a proto language before beginning. An even more significant advantage is that it gives a mnemonic clue to the user who is running *IT* to produce the adaptation interactively. In adapting directly from one dialect to another, the user (who might be a speaker of the target dialect who is bilingual in a major language) might not recognize a source dialect morpheme. Adapting from a gloss (which will appear aligned beneath the source dialect morpheme) would give the user a clue as to the identity of the morpheme being adapted. *IT*'s multidimensional model of text makes further help possible—a translation of the entire text unit in a freeform annotation would further clarify the sense of the troublesome morpheme. When adapting from an intermediate form consisting of glosses, it is necessary to devise unique glosses for each morpheme; that is, different source dialect morphemes should not be assigned the same gloss. This would require the user to choose from alternatives when going from gloss to target dialect morpheme.

The approach of using an intermediate form to adapt a text to many dialects is illustrated in figure 2.26. That example shows a

baseline text in the To'aba'ita dialect of north Malaita, Solomon Islands adapted into four other dialects on the island. The dialects in question are actually diverse enough to be considered different languages. On a Swadesh 100-word list, To'aba'ita scores the following percentage of cognates with the target dialects: Fataleka 75%, Kwara'ae 66%, Langalanga 66%, and Kwaio 54%. A pervasive custom of word tabooing on the island has led to a situation which appears to have accelerated lexical change (Simons 1982). Thus the grammatical and semantic diversity of the languages is much less than might be expected on the basis of the lexical relationships. Nevertheless, the example illustrates how widely the technique of computer-assisted dialect adaptation may apply.

Of course, *IT* does not produce a conventional output text for an adapted version. Producing this requires special-purpose processing of the interlinear text file. First, the field containing the target dialect version must be extracted. Then a consistent changes program would be used to remove the excess spaces (inserted for alignment) and remove the morpheme boundaries, performing morphophonemic changes at morpheme boundaries.

A further problem is preserving the punctuation—*IT* does not normally carry it down into the aligning annotations. The solution is to preprocess the input text to have punctuation marks as words by themselves, and then define the punctuation marks as word-building characters. They would then be annotated with themselves, thus preserving the punctuation in the annotations. This is illustrated in figures 2.25 and 2.26. A consistent changes program could close up the space between words and punctuation marks.

A final problem is the capitalization. One approach would be to enter lower case and upper case annotations for forms that can begin a sentence. Another would be to use a consistent changes program to recognize sentence-initial environments and change the first letter of words in those environments from lower case to upper case. Still another approach would be to preprocess the input text to change all capitalized words to lower case, but put a special character like @ in front of them as a word by itself. This symbol would be carried down into the annotations and then used with a consistent changes program to recapitalize the following word.

Figure 2.26 Adaptation via intermediate form to many dialects
(North Malaitan dialects, Solomon Islands)

```

\ref KUKUA s 1
\tob Ro'oki   nau   ku   too   .
\mr  ro'oki   nau   ku   too   .
\mg  yesterday ls.F ls.G stay  .
\ftk ro'oki   nau   ku   too   .
\kwr ro'oki   nau   ku   tua   .
\kwo aboní   nau   ku   to'oru .
\lng nalafi   lau   0    io    .

\ft Yesterday I stayed home.

\ref KUKUA s 2
\tob 'I u'usuwadia nau   ku   ngalia   kafo .
\mr  'i u'usuwadia nau   ku   ngali-a   kafo .
\mg  LOC morning   ls.F   ls.G bring-3s.0 water .
\ftk 'i 'ofodangi  nau   ku   ngali-a   kafo .
\kwr 'i afodangi   nau   ku   ngali-a   kafo .
\kwo ana 'usugani  nau   ku   ngari-a   kafu .
\lng 0  ra'afule   'i lau laka sake -a   kwai .

\ft In the morning I fetched water.

```

Key to field codes:

\ref	reference field
\tob	baseline text in To'aba'ita
\mr	morphemic representation
\mg	morpheme glosses
\ftk	Fataleka language
\kwr	Kwara'ae language
\kwo	Kwaio language
\lng	Langalanga language

Key to abbreviations:

1s	first person singular
3s	third person singular
F	free pronoun
G	general subject marker
O	object marker
LOC	locative

2.5 Freeform annotations

Annotations of the entire text unit are called freeform annotations because they are not constrained to a one-to-one correspondence with baseline text elements. Kinds of information that are appropriate for freeform annotations include translations of the text, explanatory comments, codes for topic indexing, and the analyst's housekeeping notes to himself.

The freeform annotations are annotations of the entire text unit. They are called freeform because they are not constrained by a one-to-one correspondence with the words or morphemes of the baseline text. They may be as short as a single topic indexing code, or as long as a paragraph discussing a unique point in the grammatical structure of the unit.

Like the aligning annotations, the freeform annotations can annotate any field that exists above it in the interlinear text. Thus it is possible to use freeform annotations to not only comment on the baseline text, but also to comment on the various analyses developed in the aligning annotations.

Figure 2.27 gives a list of possible freeform annotations. The list is, of course, not exhaustive; the resourceful investigator should be able to devise others. The possibilities listed in the figure are discussed and exemplified in the remaining modules of this section.

Figure 2.27 Possible freeform annotations

- Translations
 - literal (form-based)
 - free (meaning-based)
- Translations into
 - international language
 - national language
 - trade language
- Explanatory comments
 - implied information
 - cultural background
 - grammatical explanation
- Topic indexing
 - cultural topics
 - grammatical topics
- Housekeeping notes
 - transcriptions to recheck
 - meanings to clarify
 - analytical hypotheses to test
 - conventions being followed
- When the base text is a translation
 - source text
 - other translations
 - back translation
 - discussion of translation problems

2.5 Freeform annotations

2.5.1 Translations

Freeform annotations can be used to provide translations of the baseline text. Most common are literal (form-based) or free (meaning-based) translations. Other styles of translation are possible, and translations can be made into more than one language.

For most applications, the most important freeform annotation of all is a translation of the text into a language the reader understands well. The interlinear morpheme and word glosses give a clue to what the individual parts mean, but a translation is required to account for the role which syntax, pragmatic context, and cultural context have to play in determining the meaning of the text. Many textbooks have been written on the art and science of translation (for instance, Nida and Taber 1969, Beekman and Callow 1974, Newmark 1981, and Larson 1984). It is not our purpose here to teach how translation should be done, but rather to discuss some relevant issues of style in translation as it relates to audience and purpose.

Traditionally in text analysis, people have spoken of literal translations versus free translations. Literal translation has been equated with interlinear word or morpheme glossing. In free translation, on the other hand, the analyst is free to reorder and recast the source language elements so as to make a rendering which fits the pattern of the target language. This conception of literal translation as interlinear glossing is evident in the 1954 special issue of the *International Journal of American Linguistics* on translation (Hockett 1954:313, Casagrande 1954:337). It is still present today in a work like Larson's recent textbook on translation which equates interlinear translation with literal translation and declares that "a literal translation sounds like nonsense and has little communication value" (Larson 1984:15).

In our view, the traditional equation of literal translation with interlinear glossing is unfortunate, for interlinear sentence glossing is not sentence translation, any more than morpheme glossing is morpheme definition. The first definition of *translate* in the 1982 *American Heritage Dictionary (second college edition)* is "to express in another language, systematically retaining the original sense." When the result of interlinear glossing is neither a grammatically acceptable

form in the target language nor does it convey much (if any) sense, it should hardly be called translation.

Given the understanding that translation must result in target language forms that are grammatically correct and thus capable of conveying meaning, we propose that a well-annotated text generally requires two kinds of translation. It first needs a form-based translation which seeks to express, as nearly as possible, the form of the source language text in a grammatically correct form of the target language. This translation shows the sense of the source language syntax by expressing the content in corresponding syntactic forms of the target language. By rendering idioms and figures of speech literally, this translation gives a glimpse into the world view of the speakers. This kind of form-based translation is what we mean by literal translation.

The second kind of translation generally needed in text annotation is a meaning-based translation (Larson 1984). A meaning-based translation seeks to express as nearly as possible the meaning of the source language text in the idiom of the target language. Whereas the form-based translation seeks to use corresponding grammatical and lexical forms in the target language, the meaning-based translation seeks to use the most natural grammatical and lexical forms to achieve the same impact. This may involve recasting the grammatical structure, rewording figures of speech, filling in ellipses, supplying implied information, and the like. A well-done meaning-based translation not only conveys the same message as the original text, but also reads like a piece of target language literature in its own right. This kind of meaning-based translation is what we mean by free translation.⁹

Comparing these redefinitions of literal and free translation to the traditional uses, we see that the original notion of free translation was poorly defined. Was it form-based or was it meaning-based? In practice, it was generally something that vacillated between the two. When freely translating the meaning by its most appropriate target language equivalents would leave no trace of the source language form, the analyst was faced with a hard choice. *IT*'s multidimensional model of text solves that problem by making it easy for the analyst to provide both kinds of translation in the annotated text corpus. The literal translation can serve as an audit trail showing how one gets from what the text says (as shown in the interlinear glosses) to what it means (as shown in the free translation).

Figure 2.28 illustrates literal and free translations for the same text. It is the climax of a Lau (Solomon Islands) tale telling why it is

that today the planks of sea-going canoes are stitched together with *hata* vine and the seams and holes caulked with putty nut. In the story, forty men perished when they sank in a large sea-going canoe

Figure 2.28 Literal and free translations
(Lau, Solomon Islands)

```
\ref BARU u 10
\tx Na kui loko 'e gwouru mai 'ana 'i hara,
\mr na kui loko 'e gwouru mai 'a -na 'i hara
\mg DEF dog there 3s.G sit hither PRG-3s.O LOC shore
```

```
\tx ka haea,
\mr ka hae-a
\mg 3s.S say-3s.O
```

```
\lt The dog over there was sitting by the shore and it said,
```

```
\ft There was a dog over there, sitting by the shore, and it
said,
```

```
\ref BARU u 11
\tx "Hata ma haia,
\mr hata ma haia
\mg vine sp. and putty nut
```

```
\lt "Hata vine and putty nut,
```

```
\ft "There is plenty of hata vine and putty nut around,
```

```
\bi Hata is a species of climbing fern; its vine is used
as a cord for stitching together the planks of a canoe.
The nut of the haia tree (Parinarium glaberrimum)
contains a putty-like substance which is used to caulk
the seams between canoe planks as well as the stitching
holes.
```

```
\ref BARU u 12
\tx ma baru 'i asi too ka molea 'ana."
\mr ma baru 'i asi too ka mole -a 'a -na
\mg and canoe LOC sea stay 3s.S fungus-3s.O PRG-3s.O
```

```
\lt and that canoe in the sea sits getting moldy."
```

```
\ft but that canoe just sits in the sea rotting."
```

```
\bi Baru is a large sea-going canoe with an erect, and
generally ornamented, prow and stern.
```

Key to field codes:

<code>\ref</code>	reference field
<code>\tx</code>	text
<code>\mr</code>	morphemic representation
<code>\mg</code>	morpheme glosses
<code>\lt</code>	literal translation
<code>\ft</code>	free translation
<code>\bi</code>	background information

Key to abbreviations:

DEF	definite
LOC	locative
PRG	progressive
3s	third person singular
G	general subject marker
O	object marker
S	serial subject marker

which they had caulked with mud. An on-looker who heard the dog's remark (which is reported in the example) started the custom of using *hata* and putty nut in the construction of canoes.

Literal and free translation as we have defined them are not the only possible varieties of translation that might be appropriate.¹⁰ Translation is very much tied to audience, and it is likely that an analyst would produce different translations for different audiences. Those different translations could be in different languages. They could use different registers of the same language. They could give fuller detail in different areas of focus. Using *IT* it is easy to define as many translations as desired. Other software tools can be used at a later time to remove fields (from a copy of the corpus, of course) which are not relevant for a particular publication audience.

Casagrande (1954) gives a good discussion of translation style as it relates to the purpose for which the translation is being made. His linguistic, pragmatic, and aesthetic-poetic translation styles correspond roughly to our interlinear glossing, literal translation, and free translation. His fourth style, ethnolinguistic translation, is concerned primarily with the explication of the cultural context in the source language and of the differences in meaning between apparently equivalent elements in the two languages. This is done not only in the translation itself, but also in explanatory notes such as the `\bi` annotation in figure 2.28. It is to such explanatory comments that we now turn our attention.

2.5 Freeform annotations

2.5.2 Explanatory comments

Freeform annotations can be used to provide explanatory comments about a text, such as might fill in implied information, explain the cultural background, or discuss the grammatical analysis.

The narrator or author of a text shares much knowledge with the intended audience. The narrator takes advantage of this fact in creating the text—it is not necessary to explain things which everybody already knows and thus much of the information carried in the text is left implicit. When a text collection is being prepared for an audience outside the realm of experience of the original audience, the information which was left implicit for the original audience must be made explicit if the cross-cultural audience is to understand the text.

Freeform annotations can be used in text analysis to provide explanatory comments which fill in implied information. One kind of implied information consists of actions performed by participants or thoughts conceived by participants which the cultural knowledge of the audience supplies without the narrator explicitly mentioning them. Another kind is historical background—events or circumstances that have already taken place which the narrator assumes the audience knows about. Another kind of implied information is the cultural significance of material items, of social relationships, or of supernatural beings, which participants in the culture understand in all of their connotations and denotations, but which outsiders do not.

Figure 2.28 in the preceding module has an example of a freeform annotation which gives background information (in the *\bi* field). In this case, the field is being used to explain some of the items of material culture which are key props in the text. Note that understanding the climactic aphorism uttered by the dog is totally dependent on the cultural knowledge that *hata* and *haia* are not only vegetable substances to be found in the jungles, but also have important uses in the technology of canoe building. The narrator explains nothing of that since any speaker of the Lau language knows it already.

Another kind of explanation regards the grammatical analysis of the text. It is a long-standing tradition in linguistic text collections to present texts with morpheme or word glosses, sentence translations, and footnotes which discuss the analysis of the text. These footnotes

might discuss alternatives in the morphological analysis, point out the significance of a particular syntactic construction, or conjecture about the relationships of lexical or grammatical forms to forms in other languages. Figure 2.29 gives an example adapted from a text published in the traditional way (Munro 1976). The example shows how the explanatory footnotes can be incorporated into a multidimensional text as freeform annotations. The slash (/) is used in the grammatical notes field to separate individual notes.

Figure 2.29 Explanatory comments
(Mohave, USA)

```
\tx Mah   hav'idiitk      'idoptk      'iduum,
\at mah   hav -'-idii-t -k '-ido-p  -t -k '-iduu-m
\mg thus? thus-1-come-EMP-SS 1-be -OBJ-EMP-SS 1-be -DS
```

```
\tx 'iduuchm      duum.
\at '-iduu-ch -m   duum
\mg 1-be -SUBJ-tense? so
```

```
\ft I was just like that, I was.
```

```
\gn I have not seen mah used in any other context. Possibly
the m- prefix indicates some relationship to the various
question-indefinite words in m- found throughout Yuman./
This sentence contains another example of perfective
...-p ...-ch, showing the possibility of having an
extra clause intervene between the -p and -ch marked
ones./ duum is composed of 'be' plus the different
subject subordinator -m. (Note that the onset vowel i-
of iduu is omitted.) 'So' and 'but' are Mrs. Brown's
usual translations, but neither seems to make much sense
here.
```

Key to field codes:

```
\tx   text
\at   allomorphic transcription
\mg   morpheme glosses
\ft   free translation
\gn   grammatical notes
```

Key to abbreviations:

```
1      first person
EMP    emphatic
SS     same subject
DS     different subject
OBJ    object
SUBJ   subject
```

2.5 Freeform annotations

2.5.3 Topic indexing

Freeform annotations can be used to tag text units with codes for topic indexing. Such annotations could be used, for instance, to index a corpus for occurrence of cultural or grammatical phenomena of interest.

In order to develop a text corpus as a database for retrieval of examples in later analysis, it is often helpful to go beyond prose comments and to devise a system of topic indexing. In such a system, the occurrence of a phenomenon of interest in a particular text unit is marked by including a standard tag for that particular topic. All examples of a particular phenomenon in a text corpus can then be found by searching for every text unit which is tagged with the code for the desired topic. Such topic codes are analogous to the subject headings used in classifying books for the subject catalog of a library.

The key to success in topic indexing is to use a scheme of topics which adequately covers the area of interest. The ideal approach is to adopt a system that someone else has already devised and which has been proven to be adequate by its widespread use.

The best known indexing scheme in the social sciences is probably George Peter Murdock's *Outline of Cultural Materials* (1961). This is a system for classifying observations about culture. It involves a decimal numbering scheme which uses two digit numbers to identify 79 broad categories into which the whole range of culture is divided, and adds a third digit to define up to nine subcategories for each of the broad categories. The outline has been employed for decades in indexing the *Human Relations Area File*—an enormous database of ethnographic information about hundreds of the world's cultures. The method that was used in developing that database was to annotate photocopied pages from published ethnographies by writing in the margins the numerical codes for the cultural topics addressed therein. We could, of course, do the same thing in an *IT* text corpus by using a freeform annotation to tag units of text with the codes for the aspects of culture which they are commenting on or demonstrating. Figure 2.30 gives a sample of some of the codes in Murdock's system.

In the area of linguistics, the most fully developed topical index is probably the *Lingua Descriptive Studies* questionnaire, by Bernard

Figure 2.30 Sample indexing codes from the *Outline of Cultural Materials*

- 22 FOOD QUEST
 - 221 Annual cycle
 - 222 Collecting
 - 223 Fowling
 - 224 Hunting and trapping
 - 225 Marine hunting
 - 225 Fishing
 - ...
- 75 SICKNESS
 - 751 Preventive medicine
 - 752 Bodily injuries
 - 753 Theory of disease
 - 754 Sorcery
 - ...

Comrie and Norval Smith (1977). This work is a 60-page outline designed to serve as the standard for a series of monographs, called *Lingua Descriptive Studies* (LDS). That series, now discontinued, sought to publish grammatical descriptions of a wide variety of the world's language within a common framework. The purpose of having all monographs in the series conform to the same outline was to facilitate typological comparison of the languages. Strictly speaking, the LDS questionnaire defines an outline for the structure of a grammar description, rather than a scheme of indexing codes. Thus the designations get rather unwieldy, going out to as many as seven or eight decimal places in the outline numbering. However, the coverage is fairly complete and it would be possible for one to index the grammatical phenomena in text by using a freeform annotation to record the appropriate LDS outline numbers for any examples of interest. Figure 2.31 gives an example of some outline references in the Comrie and Smith scheme. For instance, a text unit which exemplifies the object of a comparison (as in, *greater than X*) would be tagged with 2.1.1.2.2.6 in a grammatical indexing field. Later, a search for this tag in the text corpus would pull together all examples of this construction.

Another approach to topic indexing is to develop mnemonic labels for categories of interest. For instance, figure 2.32 gives an example of some topic indexing codes that could be used for grammatical indexing. The example is based on some of the section titles used in Ronald Langacker's (1977-84) four volume *Studies in Uto-Aztecan*

Figure 2.31 Sample indexing codes from *Lingua Descriptive Studies*

Regarding formation of WH questions:

- 1.1.1.2.2.2.1 questioned element not moved
- 1.1.1.2.2.2.2 questioned element moved to initial position
- 1.1.1.2.2.2.3 questioned element moved to preverb position
- 1.1.1.2.2.2.4 cleft construction
- 1.1.1.2.2.2.5 questioned element forms the intonation nucleus
- 1.1.1.2.2.2.6 other possibilities

Regarding expression of syntactic functions:

- 2.1.1.2.2.1 subject of intransitive verb
- 2.1.1.2.2.2 subject of transitive verb
- 2.1.1.2.2.3 subject of copular construction
- 2.1.1.2.2.4 direct object
- 2.1.1.2.2.5 indirect object
- 2.1.1.2.2.6 object of comparison
- 2.1.1.2.2.7 object of equation
- 2.1.1.2.2.8 other objects governed by verbs

Grammar. This work is an edited collection of grammar descriptions of Uto-Aztecan languages, all of which conform to the same organizing outline. That outline is much simpler than the LDS outline, having 32 top-level categories and 96 subcategories. Topic indexing with mnemonic abbreviations is an ideal application for the range set component of *IT*. A standard set of topic indexing codes can be entered as a range set and then *IT* will require that all annotations in a particular freeform field be drawn only from that standardized set of codes. (See sections 3.6 and 5.11 for details.)

Figure 3.32 Sample mnemonic codes for grammatical indexing

QUESTIONS

- YNQ Yes/no questions
- WHQ WH questions
- EYNQ Embedded yes/no question
- EWHQ Embedded WH question

NOUN MORPHOLOGY

- NX>N N + X --> N
- VX>N V + X --> N
- AX>N Adj + X --> N
- OT>N Other

2.5 Freeform annotations

2.5.4 Housekeeping notes

Freeform annotations can be used as a means for the analyst to attach housekeeping notes to the units of text. These are notes which the analyst writes to himself, such as about transcriptions to be rechecked, meanings to be clarified, hypotheses to test, or conventions to follow.

The process of analyzing a text corpus is a process of successive refinements. This is especially true when an analyst who has not yet mastered the language is working with a corpus of transcribed oral texts. In the early stages of language study, the analyst is likely to make frequent errors of transcription or of interpretation and to encounter many items about which he is unsure. As the analyst goes through a text, analyzing its many dimensions, questions inevitably come to mind. These questions should be written down immediately, lest the observations they represent be lost forever. Rather than writing these questions on separate pages that can easily become disassociated from the text, *IT* offers the analyst the opportunity to insert these questions directly into the text corpus as freeform annotations. With the questions stored at the point of relevance, they can easily be followed up whenever the same text unit is processed again with *IT*. With all such questions stored in a specially defined field of the multidimensional texts, all text units about which there are pending questions can easily be extracted, such as for the purpose of planning the next session with a language helper.

There are three broad categories of information that could go in housekeeping notes. The common denominator is that these annotations, unlike others in the analyzed text, deal with the process of analysis and are aimed strictly at the analyst himself—they are not meant for an outside audience. Multiple fields of annotation can be defined for different kinds of housekeeping notes.

The first class of housekeeping notes is questions about the accuracy of the analysis so far. This could include questions about the transcription, questions about the morpheme cuts, questions about underlying forms of morphemes, questions about glosses, questions about the translation, and so on.

A second class of housekeeping notes is hypotheses that come to mind in the process of text analysis. Lest potentially important

insights be lost, the analyst can write notes to himself in a freeform annotation saying things like, "I wonder if ... ," or "What would it mean if"

A third class of housekeeping notes is observations about conventions that are being followed in the text glossing or in the other aspects of the analysis. These comments are helpful when it comes time to write the introduction to a published text corpus or when writing a grammar. If placed in a field of their own, such comments could be easily extracted and rearranged to build the basis for a description of the analysis conventions.

Figure 2.33 illustrates two fields for housekeeping comments. The first, `\txcmt`, is for comments about the baseline text field, `\tx`. In it the analyst has posed questions about the accuracy of the text. The second, `\sgcmt`, is for comments on the "string" gloss field, `\sg`. The morphology contains many indeterminate morpheme boundaries due to fusion, so the analyst has broken the words into strings which are defined by the determinate morpheme boundaries. The string gloss comments in the sample text unit are largely notes about conventions followed.

Figure 2.33 Housekeeping notes
(Slavey, Canada)

```

\ref BNCK u 2
\tx ezhi ala      me'h'lhe'i'Nzhu'          metah
\st ezhi ala      me-'h      -lhe' +i'N -zhu'  me-tah
\sg that at.first 3s-CONCOM-flour+3s.#=swell.P 3s-among
\wg that first    baking.powder              in.it

\tx ats'eh^iN.
\st a +ts'eh      -^iN
\sg so+UNSP:3s.h=do.I
\wg one.does.it.so

\ft First, one mixes baking powder in it.

\txcmmt One version has /dla/ for /ala/.

\sgcmmt What is best taken as the basic form of /'h/?
Note gloss CONCOM 'concomitant', chosen in contrast
to ACCOMP for /t'a'h/. Note gloss 'at.first' chosen
to remove ambiguity with ordinal 'first'. Include
some description of verbally-derived nouns like
'baking powder'? /-tah/ is another example of a
form with a context-free string_gloss and a
context-bound word_gloss.

```

Key to field codes:

\ref	reference
\tx	text
\st	string analysis
\sg	string glosses
\wg	word glosses
\ft	free translation
\txcmmt	text field comment
\sgcmmt	string gloss field comment

2.5 Freeform annotations

2.5.5 When the base text is a translation

When the baseline text is a translation, freeform annotations can be used to provide information that will aid in the production and evaluation of that translation. Possible annotations include the source text, other translations, and notes for future checking and refinement.

If the baseline text is itself a translation, freeform annotations can be used to provide much supporting information. First of all, a freeform annotation can be used to store the original source text for the translated unit of text. Additional annotations can be used to store other translations of the same source text which can be used for comparison. If the baseline translation is into a minor language, a freeform annotation can be used to translate it back into a major language so that others who do not know the base language can get a sense of how the translator has expressed the source text in the idiom of the target language.

Freeform annotations of a housekeeping nature can be used to support the process of translation. Freeform annotations could be used to describe translation problems, to document reasons for adopting the given rendering and eschewing alternatives, to record the questions and comments of reviewers, or to keep the translator's notes on items to check for future refinements.

Figure 3.34 gives an example from a verse of the *Bible* translated into the Lau language of the Solomon Islands. The baseline text is broken into morphemes, glossed, and translated just like a normal text. This provides a morpheme-by-morpheme accounting for the translated text along with a back translation into English. The fields that follow are peculiar to the fact that the base text is a translation. The `\qst` field poses questions that the translator needs to answer before being satisfied with the translation of this unit of text. Then follow fields for five other versions of the text—the Greek source text, three English translations, and the translation made into the closely related language of To'aba'ita in 1923.

Figure 3.34 Annotation of a translated base text
(Lau, Solomon Islands)

```

\ref John 1 v 14
\tx Si Baea 'e hau 'ana imola
\mr si bae -a 'e hau 'ana imola
\mg PRTV talk-NZR 3s.G become CMP person

\tx ma 'e too 'i siaga.
\mr ma 'e too 'i sia -ga
\mg and 3s.G abide LOC with-12p

\ft The Discourse became a person and lived with us.

\qst Verify that si Baea refers to a whole discourse
      rather than to an individual word. Should it be
      nga Baea (as in \tob)? Should the second 'e be
      the serial form ka?

\grk Kai 'o logos sarx egeneto kai eskhwnsen en 'hmin.
\kjb And the Word was made flesh, and dwelt among us.
\niv The Word became flesh and lived for a while among us.
\tev The Word became a human being and lived among us.
\tob Ma nga Baea eri ka imole, ma ka to i maalutamilia.

```

Key to field codes:

\ref	reference
\tx	text
\mr	morphemic representation
\mg	morpheme glosses
\ft	free translation
\qst	questions
\grk	Greek source text
\kjb	King James Version
\niv	New International Version
\tev	Today's English Version
\tob	To'aba'ita language version

Key to abbreviations:

PRTV	partitive article
NZR	nominalizer
3s.G	3rd person singular general subject marker
CMP	complement
LOC	locative article
12p	1st person inclusive plural

Notes

¹ ASCII is the American Standard Code for Information Interchange. It is the character set standard for the microcomputer industry. It defines the standard 7-bit representation for 32 control characters (such as backspace, line feed, carriage return), the space character, 94 printable characters, and the delete character. An "ASCII text file" is one in which all the bytes adhere to the ASCII standard. (There are, for instance, no numbers coded in binary form and no machine language instructions.) By "clean ASCII text file" we mean an ASCII text file in which there are no invisible control characters—the only control codes are carriage return and line feed. For a listing of the ASCII character set, see your computer's operating system manuals. Such a list is also commonly found in programming language reference books (such as Simons 1984:177). *IT* also supports the Extended ASCII standard of the IBM PC. This defines characters for the 128 extra codes obtained by allowing 8-bit character codes.

² *IT* actually uses a subset of SIL standard format called field-oriented standard format. This further prescribes that the backslash codes which identify fields occur only at the beginning of a line, that the next blank line or backslash code terminates the field contents. (See section 9.1.)

³ While the two-level model of text glossing has become a time-honored tradition in linguistics and anthropology, we feel that it has been motivated more by economy of preparation time and publication costs, than by conviction that it is the best way to present analyzed text. The age of personal computing has completely changed the former economic formulas. Software like *IT* makes it feasible for the scholar to prepare texts in complex multiline formats. New publishing media like floppy disks, on-line databases, computer output microfiche, and on-demand xerography make it possible to publish analyzed text collections with minimal overhead costs.

⁴ In informal morpheme glossing there is another class of elements which are best left unglossed. These are elements that have no lexical meaning at all. Phonemes or syllables may be introduced into the surface form of a word strictly for the sake of pronounceability or euphony. These intrusive partials may reflect a process of prothesis, epenthesis, or paragoge (for intrusions at the beginning, middle, or end of a word, respectively). In formal glossing, an abbreviation should be used to identify such elements as intrusive and perhaps even identify the process. In informal glossing, however, such detail is likely to get in the way.

⁵ The style with upper case abbreviations for functor morphemes is in fact the standard for *Language*, the journal of the Linguistic Society of America (Bright 1984). It is advocated by Christian Lehmann (1982) in what is the only recent literature we are aware of on the subject of how to

do interlinear text glossing. It is also evidenced in many recent textbooks (for instance, Comrie 1981, Foley and van Valin 1984, Givón 1984).

⁶ The most complete listing of possible abbreviations we have found is in Lehmann (1982). This listing, which includes about 170 terms and proposed abbreviations, was compiled by collating the terms and abbreviations from three published text collections. One of these, which provides a particularly good model, is Ronald Langacker's (1977-84) four volume set on Uto-Aztec languages. Another good source for terms and abbreviations is numbers of the *Lingua Descriptive Studies* series.

⁷ Note that the subcategorization relation is different than the composition relation signified by period in morpheme glosses. For instance, in the category label *VRt:tr*, *tr* is a kind of *VRt*, while in the morpheme gloss *3s.PRES*, *PRES* is not a kind of *3s*. The components of a gloss are equal partners and could be expressed in any order. It is because the subcategorization relation is something very different that we propose the different convention for representing it.

⁸ Verb subcategorization also has a part in case grammar analysis when analysts make generalizations about the case frames governed by certain semantic subcategories of verbs. See, for instance, Longacre's (1976:40-49) periodic-chart-like scheme for classifying case frames.

⁹ Many authors call this *idiomatic* translation (Larson 1984, Beekman and Callow 1974). Others call it *dynamic* translation. The term *free translation* is little used in the translation literature, except in the pejorative sense of *unduly free* translation (Larson 1984, Beekman and Callow 1974). There is, nevertheless, a time-honored tradition of *free translations* in text annotation. Our aim here is not to introduce new terms, but to assign more useful meanings to the old ones. Similarly with *literal translation*. What we have defined as *literal*, other authors have called *modified literal* (Larson 1984, Beekman and Callow 1974). It is unfortunate that this, too, carries something of a pejorative sense. Many a translator, in an effort to keep a safe distance from the traditional notion of literal translation (which we have said isn't really translation), has thrown out the baby with the bath water in eschewing the modified literal approach as well. If history is a good teacher, however, that approach is well proven. No translated work in history has had more impact on the world than the literal (in our sense) translation of the *Bible* made in 1611 under the sponsorship of King James I of England.

¹⁰ Particularly noteworthy is Newmark's work on defining the methods of semantic versus communicative translation. Semantic translation, which has a source language bias, "attempts to recreate the precise flavour and tone of the original: the words are 'sacred', not because they are more important than the content, but because form and content are one" (1981:47). Communicative translation, which has a target language bias, "attempts to produce on its readers an effect as close as possible to that obtained on the readers of the original" (1981:39).

—

—

—

1.2 About this book

This book has a distinctive structure intended to enhance its ability to communicate. Everything is taught in bite-sized modules which have a constant layout. The modular structure also enhances our ability to maintain the book, and we encourage readers to report deficiencies they find as they use it.

This book is an example of a “structured document” as defined by Edmond Weiss (1985). Weiss observes that “developing a set of user manuals is very much like developing a complicated computer program” (1985:xii). He advocates a method of decomposition by step-wise refinements in which the total program documentation is hierarchically broken down into divisions until a bottom level is reached at which there are bite-sized lessons which communicate one key point. These lowest level divisions are called “modules.”

It is the module structure which is the distinctive hallmark of this style of book. As a general rule, a module is a two-page unit on facing pages, though this book does contain a number of one-page modules, and even some three- and four-page ones. A complete module has five parts, listed here in order of presentation from upper left to lower right: (1) context headings, (2) module title, (3) summary, (4) text, and (5) exhibits. The context headings give page numbers, book title, chapter title, and titles of intermediate divisions if the current module is deeply embedded. The summary encapsulates the main point addressed in the text. It is possible to skim a book like this, simply by reading the summaries. The exhibits (not present in every module) provide a graphic example or a tabular reference summary of the material presented in the module.

Weiss shows that the process of developing user documentation is like the development of computer programs, not only because both should be hierarchically structured and modularized, but because both have the same life cycle. Just as software has a five-stage life cycle going from analysis to design to coding to testing to maintenance (see section 7.3 of this book), so does user documentation. (For user documentation, the writing phase is analogous to coding, and the editorial phase is analogous to testing.) We have now been through the first four phases, and with the publication of this book we are entering into the phase of maintenance. Just as any newly published program is guaranteed to contain bugs and undesirable features, so

too is this book. That is why we have chosen to publish it in a looseleaf format—to make it possible to maintain the book through packets of update pages. The organization into modules has a further advantage of making the book easy to maintain, since we can slip out an old module and put in a replacement without disrupting the pagination.

We therefore want your “bug reports” and “wish lists.” Where did the documentation mislead you? Where did it confuse you? Where did it fail to answer the question you had? When were you unable to find something you were looking for? What recipes are missing? For what new applications have you used the programs? What user interface programs have you developed? We are eager to receive such feedback so that we can debug and enhance the manual in parallel with our continuing work to debug and enhance the programs. Please send your comments addressed to the Linguistics Coordinator at Summer Institute of Linguistics, 7500 W. Camp Wisdom Road, Dallas, TX 75236, USA.

1.3 Project history and acknowledgements

The prototype for *IT* was developed in 1982. The project to develop the current version began early in 1985. Test versions have been in use since early 1986.

The *IT* system has its roots in the GLOSS system designed in 1982 by Gary Simons. It was implemented at that time by Kristine E. Chambers at a field site in Solomon Islands. That program, written in the PTP language (Simons 1984b), served as the prototype for *itp*.

In January 1985, work began at the Summer Institute of Linguistics (SIL) headquarters in Dallas, Texas to design and implement a new system. The major change was to generalize the original two-level system of text and interlinear glosses to an *n*-level system of arbitrary, user-defined annotations. Another key change was to make data files compatible with the SIL standard format. The redesign was done by the team of Gary Simons and Katrina Larson. That design was implemented by Larry Versaw in the C language.

Beginning in early 1986, preliminary versions of *IT* were released for user testing. A significant source of user feedback have been seminars and workshops featuring the use of *IT*. These have been conducted for the linguistics departments of the University of Oregon and of Rice University, the SIL summer session at the University of Oklahoma, and the North America Branch of SIL. This feedback has resulted in many design changes, as well as the inevitable bug fixes. The users who have contributed to this process are too numerous to mention. There are two people, however, whose special contributions we would like to acknowledge. These are Eugene Loos, who has been a most enthusiastic administrative sponsor of the project, and Linda Simons, who has served as the copy editor for this book as well as of previous versions of the documentation.

We want also to gratefully acknowledge the contributions of our financial sponsors. The selling price of the package covers nothing more than the costs of publishing and distribution. The development costs have been borne in part by the administrative budget of SIL, but primarily by donations from a host of family, friends, and churches who have sponsored the salaries of the developers.

Finally, we must acknowledge the sources of the unpublished language data examples. The examples from Solomon Islands are from the field notes of the first author; the examples from Korean are from a field methods project conducted by the second author.