COMPUTER ASSISTED DIALECT ADAPTATION

THE TUCANOAN EXPERIMENT

by

ROBERT BRUCE REED

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 1986

To the women in my life:

My mother, Frances Norton Reed;

My wife, Judith Carlson Reed;

and

Our daughters, Christie and Robin Reed.

ACKNOWLEDGEMENTS

The research presented in this dissertation was carried out under the auspices of the Colombia Branch of the Summer Institute of Linguistics during the period from 1979 to 1984. I am grateful to my colleagues who shared published and unpublished notes relating to their research in Tucanoan languages, and took the time to discuss various aspects of the languages that they studied. I especially thank Janet Barnes, the linguist for the Tuyuca language which served as the target for the adaptation effort, for her input on Tuyuca.

The computer program for dialect adaptation described here is an outgrowth of similar work done by Robert Kasper and David Weber of the Summer Institute of Linguistics for Quechuan languages of central Peru (1982). That work in turn was based on pioneering research done by David Weber, of the Summer Institute of Linguistics, and William Mann, of the University of Southern California Information Sciences Institute (1980, 1981). I am especially grateful to David Weber for his encouragement and insightful observations during various phases of this project, both in the design phase and during the writing phase of this dissertation.

I am also grateful to Donald Burquest for chairing my

doctoral committee. His input and guidance were indispensable to its successful completion. I thank the rest of my committee for their help and insightful comments: Drs. Robert Mugele, Lynn Peterson, Virgil Poulter, and Ronald Werth.

Any extended work requires sacrifice, and I thank my wife Judy for sacrificing in many ways to see this work through to completion.

Last, but certainly not least, I am grateful to my Lord Jesus Christ, who has given me a purpose in life, and who has given me the privilege of serving Him in this way.

April 9, 1986

ABSTRACT

COMPUTER ASSISTED DIALECT ADAPTATION:

THE TUCANOAN EXPERIMENT

Publication No. _____

Robert Bruce Reed, Ph.D.

The University of Texas at Arlington, 1986

Supervising Professor: Donald A. Burquest

This dissertation provides the theoretical basis for a computer program that adapts textual material from one language of the Tucanoan family to another. Tucanoan languages are spoken by small groups living in southeastern Colombia, northwestern Brazil, northern Peru, and northern Ecuador.

This work represents the first attempt to apply principles of machine translation and computational linguistics to indigenous languages of Colombia. It discusses aspects of translation theory relevant to machine translation. Some features of the Tucanoan languages relevant to the adaptation process are discussed in depth, including differences in suffix systems marking case, noun classifiers, and the

evidential systems of the various languages. Of particular interest for automated parsing is the problem of null allomorphs of certain morphemes.

CONTENTS

REFERENCE MATERIALS

APPENDIX

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 General Problem Statement

1.1.1 Communication: the Purpose of Language

Human languages are vehicles for the communication of ideas, feelings, aspirations, procedures, beliefs, counsel, and many other concepts. When people share a common language, this vehicle can operate smoothly to convey the desired message. On the other hand, where there is no common language, and the message to be conveyed is considered important enough to justify it, an effort is made to translate from the language of the originator of the message to a language that the intended recipient can understand. This process, translation, is fraught with many complexities and pitfalls, but is preferable to the alternative, i.e., no communication at all.

Since electronic computers were first invented, there have been many attempts to use them to accelerate and simplify the translation process. These attempts have met with varying degrees of success, but the desirability of automatic translation and the commercial or practical value of a successful system have provided a continued impetus for research leading toward the goal of automatic translation systems.

Many of the problems involved in translation increase as the differences in languages increase. This is intuitively obvious, but there are also good empirical reasons for this. If translation can be initially viewed as a mapping of a message in one language to the same message in another language, it is apparent that there are many different areas in which the mapping can become distorted.

A message, whether spoken or written, commonplace or literary, prose or poetry, has the particular form, style and structure that the speaker or writer gives it when it is created. All of the various aspects of the message are important to its impact on its hearers, and perfect translation would recreate the message in the target language (TL), in keeping with the formal requirements of that language, in such a way that the stylistic, aesthetic, and semantic parameters of the message remain the same as they were in the source language (SL). Obviously, such perfect translation is rarely, if ever, achieved.

1.1.2 Problems in Communication

Imperfect translations are due to such factors as differences in mapping lexical and semantic notions; lexical specificity versus lexical generality; cosmology; and cultural expectations or values shared by the speakers of the SL, yet not shared by speakers of the TL -- expectations

which allow some required information to be left implicit in the SL.

1.1.2.1 Mapping differences

There are six types of mapping logically possible: one-to-one, one-to-many, many-to-one, many-to-many, one-(or many)-to-none, and none-to-one (or many). Of these, the last five provide mapping differences, possible problem areas.

A one-to-one mapping occurs when there is one possible lexical choice in the SL and one possible lexical choice in the TL. This is the most desirable situation for translation, since no mapping problems are possible.

A one-to-many mapping is found where a given lexical choice in the SL may have a number of choices in the TL, such as the well-known example that English has one word for snow (which may have adjectives further specifying its consistency, texture, etc.) while Eskimo languages have many different words for snow, which are lexically distinct from each other. The implication that this has for translation is that either all possible output forms must be generated, allowing the post-editor to select the one appropriate under the circumstances, or the post-editor must be aware, when he or she sees the word selected to represent the TL mapping set, that special attention is required.

Both cases occurred in the Tucanoan CADA experiments. An example of the selection approach is seen in the Tucano root masã, which means either the noun 'people' or the verb 'resurrect.' In some cases, selectional tests are able to determine whether the noun or the verb is intended, but in other cases, both forms are accepted as possible analyses. In Tuyuca, the noun 'people' is basocá, while the verb 'resurrect' is masã. The linguist chose to have the system produce both alternatives, and she selects the desired form during post-editing. On the other hand, the Tucano root ni means either 'to be' or 'to say,' both common forms in the language. In Tuyuca, nii means 'to be' and jĩĩ means 'to say.' Rather than require the linguist to delete an alternative every time the verb 'to be' appears, she chose in this case to be aware of the problem, and change the form during post-editing when it should be the verb 'to say.'

A many-to-one mapping is the converse of a one-to-many mapping. In the converse of the snow example mentioned above, an Eskimo language would be the SL and English the TL. The implication that this has for translation is a possible loss of information in the translation. This can be handled by building the descriptive modifier into the equivalence table, or simply allowing the information to be lost, having the single TL form as the equivalent of each of the many forms in the SL.

The fourth possible kind of mapping difference is many-to-many. This is inherently different from the previous kinds because it deals with many meanings instead of many forms. This occurs where the lexical item in the SL has several meanings, which may all be desirable in the particular context, but the corresponding lexical item in the TL has a different set of meanings, only some of which overlap. This is by far the messiest problem for translation: if all possible TL equivalents are generated for each form, it is possible to overwhelm the post-editor with the effects of a combinatorial explosion of verbiage. Generally, a compromise could be reached empirically as the result of experimentation with the kind of translation being done, but this might be difficult to determine. The alternative is to leave it to the post-editor to grapple with the problem. I am not aware of any problems of this kind between the source and target languages used in our adaptation experiment.

The fifth kind of mapping is a special case: one (or many)-to-none, the case where there is no TL equivalent. One approach to this situation is to require the translator to provide an explanatory definition as the translation. In the case of this computer program, the approach would be to fail the analysis for the SL word. This has the effect of passing the SL word through the transfer and synthesis phases untouched, requiring the post-editor to deal with the

problem.

The last kind of mapping is another special case: none-to-one (or many), the case where there is no equivalent in the SL for an obligatory TL morpheme. An example of this can be seen by comparing the possessive noun phrase constructions in Spanish and Portuguese. In Spanish, articles are not permitted in possessive noun phrases, while Portuguese requires them with possessive pronouns; i.e., Spanish su casa 'his/her house', versus Portuguese a sua casa 'his/her house.' This type of construction is sufficiently common in Spanish and Portuguese to cause problems in a word-by-word approach to adapting textual material between them. The none-to-one mapping would be in the direction from Spanish as the SL to Portuguese as the TL. This mapping puts an extra burden on the post-editor to resolve the problem.

1.1.2.2 Lexical generality versus lexical specificity

Partly overlapping with mapping differences is the question of lexical specificity versus lexical generality; in the example of snow above, the Eskimo language is lexically specific about each type of snow, and the various types of snow are not grouped together generically in any way (unless there is a separate generic term), while English has only one general word 'snow,' which is generic for all

types of snow. If any distinction between types of snow is desired in English, the distinction must be made using a modifier to indicate the specific type.

1.1.2.3 Cosmology

The problem of cosmology has attracted the attention of anthropologists and linguists for many years. Whorf and Sapir outlined a hypothesis of linguistic relativity which states "that the structure of a human being's language influences the manner in which he understands reality and behaves with respect to it" (Carroll 1956:23). Whorf also wrote "A change in language can transform our appreciation of the Cosmos" (Whorf 1956:263). He believed that Western science, as well as all human reasoning activity, is molded and shaped by the linguistic patterns of the primary language of the researcher. His studies of American Indian languages led him to conclude that Hopi (as well as other languages) and English have such different cosmological bases that any kind of translation between them is impossible. Whorf's conclusions have of course been disputed, but the issues raised are important ones to consider for any attempt to translate, especially between languages spoken by drastically different cultures.

The linguistic categories which influence (or Whorf would say determine) our thinking are of various types:

temporal, spatial, aspectual, etc. Whorf's claim is that these linguistic categories determine the semantic categories into which we segment all experience, thus determining and limiting our perception of the real world. These categories are reflected in various ways in different languages.

For example, Indo-European languages have three primary subdivisions within the temporal category: past, present, and future. Past tense refers to events or states which took place or existed before the time of speaking; present tense refers to events or states which are taking place or exist at the time of speaking; and future tense refers to events or states which are expected to (or may possibly (or hopefully)) take place or exist after the time of speaking. Within each of these primary subdivisions there may be further temporal relations, such as perfective, which denote relative time between two events or states. In Tucanoan languages, on the other hand, there are from four to six primary temporal subdivisions (some languages have two or three past tenses, a present tense, and one or two future tenses). The past tenses (and future tenses for languages with more than one) indicate degrees of remoteness from the time of speaking -- both spatial and temporal remoteness. Thus a remote past tense may refer to an event which took place a long time ago, or a long distance away (regardless of remoteness in time). This combination of spatial and

temporal categories has no equivalent in Indo-European languages -- any attempt to translate it requires a recasting of the message into terms which are relevant to the categories of the TL.

Within the aspect system of the Tucanoan languages lies the area of speaker certainty with respect to the information being presented. This area, discussed more fully in section 4.3.3.2.1, presents the problem of categorial mismatch. Briefly, the Tucanoan evidential system requires that the speaker indicate the source of the information being presented -- first-hand visual or non-visual, reported, evident, or assumed. These obligatory categories for Tucanoan are irrelevant for Indo-European languages.

These and other problems of equivalence are the kinds of problems to which Whorf refers when he raises the issue of intranslatability -- problems arising from different ways of segmenting the semantic space. In some cases, the translation process may involve an explanation of cultural differences, making explicit some information that is implicit in the linguistic forms of the SL, but this does not mean that communication between speakers of different languages is impossible, as Whorf claimed.

1.1.2.4 **Mutual knowledge**

Related to the problem of cosmology is the more general area of mutual knowledge and shared information (implied information), shared (or unshared) cultural expectations or values, which is the subject of intense on-going research in the areas of artificial intelligence and cognitive psychology. Attempts to develop computer systems that understand normal human speech or dialogues and respond appropriately (whether by obeying commands, as in robot systems, or by supplying the requested information, as in natural language interfaces to computer data bases) run into the problem that humans use to great advantage the shared cultural cues and extralinguistic references and allusions given in a dialogue. The problem that computer systems have in understanding this type of information closely parallels the problem that non-native speakers of a language have in understanding such cues or allusions. Schank and Abelson (1975) and others have attempted to provide mechanisms for representing this implicit or contextually defined knowledge that speakers share. They have called these mechanisms scripts and frames. These mechanisms define what is expected in a given situation, such as a restaurant. Specific details expected by the speaker or hearer may be specified or omitted. If they are omitted, the hearer supplies them in his own mental model of the situation by drawing on default values supplied by his own expectations of what is

appropriate behavior in the given situation.

1.2 Definitions

This study deals with computer-assisted dialect adaptation (CADA). The component terms are defined below.

1.2.1 Translation

The term translation means different things to different people. In its simplest form, translation involves taking a message currently available in one language and expressing it in another language. Chapter 2 will attempt to provide a more complete definition.

1.2.2 Computer-assisted

The term computer assisted translation (CAT) or machine assisted translation (MAT) is used in the literature to describe a wide range of activities, from an automatic dictionary lookup function which scans a term bank for corresponding terms in the TL, to interactive translation aids of varying capabilities. In the context of this study, it refers to the use of a computer program which operates in a batch mode in three steps to automatically analyze a SL text (morphological parsing), do lexical and phonological transfer from the analyzed text form to the TL form using language-pair specific phonological rules and substitution tables, and synthesize the TL form by applying TL specific

rules to the transferred form.

CAT systems are generally contrasted with machine (or automatic) translation (MT) systems, which have as their goal automatic (unassisted) translation. MT systems operate unassisted during processing, but although they have the goal of automatic translation, they generally require human post-editing of the resultant output. On the other hand, CAT systems tend to focus on interactive computer assistance of a human translator. In this sense, CADA works like a MT system, but is designed to assist a human translator. MT and CAT systems will be covered in more depth in Chapter 3.

1.2.3 Dialect

The term dialect refers to variation within a speech community. Francis defines a dialect as a variety "of a language used by groups smaller than the total community of speakers of a language" (Francis 1983:1). Dialect differences can be based on variations of any part of language, or more specifically, phonological or phonetic variations (such as the "Southern drawl"), lexical variations (such as the invention or use of words indicating membership in a group or a sub-culture), syntactic variations, semantic variations (similar to lexical variations, but including redefinition of words already used in the language), etc. These dialect differences can indicate geographical origin, social class,

group membership, and other things. Dialects continue to diverge from each other until the differences become great enough to cause difficulty in understanding speakers of other dialects. At this point, linguists refer to them as separate languages. There is no clearly defined way of determining when a pair of dialects are to be considered separate languages (Francis 1983; Trudgill 1983). This study will not attempt to distinguish between languages and dialects, and, for our purposes, the terms may be used interchangeably -- members of the Tucanoan family will be referred to as separate languages.

The translation or adaptation process is generally applied to mutually unintelligible, or partially unintelligible, dialect pairs (otherwise, the process is not needed at all). This process could also be applied in those cases where sociolinguistic factors, such as dominance relationships or differences in social status (prestige), dictate the unacceptability of the existing text in the other dialect.

1.2.4 Adaptation

The idea of machine translation (MT) often evokes either undeserved optimism or excessive pessimism on the part of people who think about it.

This optimism is often based on the idea, prevalent among lay people today, that computers can do anything. Another source of this optimism is a fundamental ignorance, on the part of even some computer science professionals, of the complexity involved in human use of natural language, not just in the size of the lexical space (the number of words), or in the complexity of natural language grammars (there probably does not exist any human language whose grammar has been completely described or formalized), but also the implicit information that human listeners bring to bear on what they hear, based on shared cultural values and experiences.

Conversely, pessimism is based on an overemphasis on these problems and limitations, without realizing that computers can be very helpful in many areas of translation, even if they can't do it all.

In order to avoid both extremes of optimism and pessimism associated with the term MT, Weber and Mann, the researchers associated with the initial experiments in dialect adaptation for Quechuan dialects, on which this work is based, selected the term adaptation as more neutral, making a much more modest claim. In keeping with its association with Weber and Mann's work, I retain the use of the term adaptation as a title, but propose to show that the results

produced are comparable in quality, if not in complexity, to results produced by some systems claiming to do machine translation. In the end, however, the choice of term depends on definitions rather than indicating substantive differences.

Melby (1985) argues that the distinction between MT and computational translator aids is a false distinction: fully automatic high-quality translation by machine does not yet exist, so every translation system using a computer is a computer-aided translation system -- the question is who is the tool: does the human translator help the computer do the translation by answering questions or post-editing, or does the computer help the human (Melby 1985:46)?

Languages that are closely related linguistically share many formal and structural features due to their common ancestry. These shared features can simplify the translation process by limiting some of the differences in form and style between languages. Some reasons why shared features come about are presented in section 1.3.

There is a continuum of difficulty in implementing a system for automatic translation. As indicated above, the languages that are least difficult to translate between are those that are closely related in every way: phonologically,

lexically, syntactically, and semantically. Take as a minimal case translating from a language to itself, where there is an identity mapping and virtually no analysis needed, i.e., x -> x for all characters (or strings) x. At the other extreme are languages that are unrelated. In this case, the syntactic and semantic categories of the TL may be sufficiently different as to require changes in the mapping of the semantic content of the SL onto the available lexical, morphological, and grammatical structures of the TL. Semantic categories and components that are required in one language may not even exist in the other. An example of this is the way the Tucanoan evidential marking system, discussed in section 4.3.3.2.1, requires semantic categories in Tucanoan languages (the evidentials) that are irrelevant and even difficult to determine in, say, an Indo-European SL.

Phonological differences are not as much of a problem as syntactic and semantic differences. If differences between surface phonological forms are regular, the proper phonological form can be generated by phonological rules. Otherwise, the character string representing the proper form can be derived by simple substitution. In some cases, it may be simplest computationally to ignore the linguistic generalizations describing the differences and treat all phonological selections as cases of substitution. In the

Tucanoan CADA experiment, some instances of each case (phonological rule and allomorph substitution) are used.

1.3 Historical Change

Over the course of centuries, languages that were originally dialects of one language diverge in different, yet generally systematic ways, due to factors outlined below, until they reach the point where speakers of one dialect can no longer understand speakers of another. At this point, translation is required to communicate messages from one dialect to the other, just as much as between totally unrelated languages. However, the translation process can be simplified in these cases by the fact that there are regular correspondences between these related languages, whereas such correspondences do not exist between unrelated languages.

Living languages are constantly changing. These changes come from many sources and result in phonological and phonetic change, morphological change, lexical change (including lexical borrowing through contact with other languages as well as diachronic semantic shift within a language), syntactic change, and semantic change. Some sources of language change include phonological and morphological regularization of irregular forms by analogy to other regular forms; morphological or lexical borrowing;

linguistic dominance of one dialect over another due to conquest or association with a higher social class (prestige dialects); semantic shift; and other influences from neighboring languages (Antilla 1972; Bynon 1977).

Tucanoan languages have had more opportunity than other languages to affect each other over extended periods of time due to the practice of tribal exogamy by the people who speak them (see section 4.2). Since men marry women who speak other languages, many Eastern Tucanoan villages may have from six to ten different languages represented (Metzger personal communication). Children may be fluent in several languages as they grow up, and be conversant with several more. This cross-linguistic contact presumably has had a leveling effect on the Eastern Tucanoan languages especially.

Tucano is the trade language of a large area of southeastern Colombia, spoken by many for whom it is not a family language. This may have resulted in a tendency toward simplification in certain areas of the grammar: Tucano has the fewest number of tenses, cases, and classifiers of any Eastern Tucanoan language (see section 4.3.3).

Language change is not something clear-cut that comes about overnight. Individual speakers and groups begin to

gradually change the pronunciation of certain words or phonemes, or regularize irregular forms, and over a period of time, as their children learn the language from them, they learn the changed forms. Many changes are constantly occurring within a speech community. Some of these changes are shared by different subsets of the speech community than those which share other changes. Linguistic atlases have been developed which map the geographical areas sharing certain dialect features, called isoglosses. These isoglosses may be phonological, lexical, or grammatical.

The following sections will discuss phonological, morphological, lexical, syntactic, and semantic changes in more depth. This discussion of language change will be relevant to implementation details of the analysis, transfer and synthesis phases of the adaptation process. Phonological changes, which are consistent between the pair of languages being adapted, are frequently derivable by phonological rules which capture the linguistic generalizations involved. Other changes which can be described by rules include morphological insertion, deletion, coalescence and reordering. Other types of change, such as lexical change, may be better treated by the computer program as cases of direct substitution. The following discussion of language change helps provide a better understanding of the kinds of linguistic differences with which an adaptation system must deal.

1.3.1 Phonological Change

Robert King (1969) outlines phonological change within the theoretical framework of classical generative phonology. Generative phonology has little use for static phonemes with allophonic statements of distributional environments. Instead, the emphasis is on capturing "linguistically significant generalizations" in phonological rules. These rules refer to binary features, and are strictly ordered with respect to each other: the first ordered rule applies to a systematic phonemic representation of the underlying form, and can add or change features or segments in that representation, creating an intermediate form to which the other rules which can apply are applied successively until all possible rules have applied. Within this framework, King proposes four primary types of phonological changes: rule addition, rule loss, rule reordering, and rule simplification. He distinguishes between primary changes (changes in the rule component) and restructuring (changing the underlying representation). Two related languages can differ with respect to rule inventory and rule ordering.

Later revisions to generative phonology emphasize a need for "natural" or phonetically motivated rules. This has led to various competing theories calling themselves "natural phonology" or "natural generative phonology." What characterizes these theories is, among other things, the

prohibition against extrinsically ordered rules (Hooper 1976). Hooper argues that language changes are usually modifications to existing rules due to phonetic environments. These phonetic changes create alternations which may later become fixed in the morphological structure of the language, losing the phonetic basis which created them in the first place.

Within both approaches, language change takes place in two ways: in the speech of adult speakers, and in the process of language transmission (language learning by children). Several kinds of language change exist among adults: additions to the lexicon (learning a new word) (see also section 1.3.3), changing to a more prestigious pronunciation, and hypercorrection to the pronunciation of a prestige dialect (King 1969). Language change during the process of language learning by children results from the tendency of children to simplify the language as they learn it, such simplification at times resulting in reordering or losing rules existing in the adults' grammar during the process.

Phonological changes are the easiest to describe and the correspondences described by a comparative phonological study form the usual basis for determining closeness of relationship, represented either by genetic trees or by tables such as Table 1 in Chapter 4 for the Tucanoan

languages.

Such phonological comparisons between languages are done by observing pairs or sets of words from different languages which are phonetically and semantically similar. These pairs or sets of words are called cognates. On the basis of observations of the differences in large cognate sets, rules or formulas can be developed expressing the phonetic or phonological differences between the dialects or languages (Key 1981).

Languages which tend to share rules in a systematic way are grouped together, until a genetic tree emerges. Such a tree is merely a hypothesis graphically depicting the purported closeness of relationship, and may well be incorrect, based on other criteria, such as morphological or syntactic similarity. On the basis of these rules, it is sometimes possible to reconstruct the phonological system of the (possibly) hypothetical language from which the currently existing languages descended. This language is called a proto-language.

The genetic tree is traditionally based on lexical similarities of "core vocabulary," vocabulary that is generally thought to be resistant to borrowing, such as kinship terminology, household artifacts, environmental features

such as rivers, water, mountains, etc. One problem with relying strictly on lexical comparisons is that such "core" vocabulary occasionally is borrowed, skewing the results. Syntactic and morphological features, such as tense markers, word order, etc., are also important for the process of determining genetic affinity, yet relatively little research has been done in comparative syntax and comparative morphology.

Sound changes can involve changes in the phonological structure of the language, or merely changes in the phonetic realization of a particular phoneme. Antilla (1972:69-70) lists several types of changes that may occur in the phonological structure of a language (or word):

1. Phonemes may be completely lost.
2. Phonemes may partially merge with other phonemes.
3. Phonemes may be partially lost, i.e., lost in certain environments.
4. Phonemes may completely merge with other phonemes.
5. Phonemes may split into other phonemes.
6. Phonemes may come into existence in certain phonetic environments where they were previously conditional variants.

Antilla (1972:71) also lists several kinds of phonetic processes which may result in phonological changes:

1. Assimilation (feature changes to increase the similarity between the sounds).
2. Dissimilation (feature changes to increase the difference between the sounds).
3. Metathesis (order reversal).
4. Haplology (reduction of geminate sequences).
5. Contamination (influence of non-contiguous segments).

Waltz and Wheeler (1972) provide a partial reconstruction of the phonological system of Proto-Tucanoan. This reconstruction forms the basis for the classification of the relationships between the Tucanoan languages presented in Table 1 in Chapter 4.

The Tucanoan CADA program handles phonological change through rules incorporating these changes. These rules are applied during the transfer or synthesis phases.

1.3.2 Morphological Change

Morphological change involves changes to the morphemes of a word. These changes can include metathesis (and other changes in order within the word), loss of morphemes, addition of morphemes, and coalescence (combination) of morphemes, which create new morphemes. Morphemes, particularly pronouns, which are originally free (can stand alone as

separate words) may reduce and become bound, creating new suffixes or prefixes. This is the case with possessive pronouns in Tatuyo and Carapana (Metzger 1981), which are the only prefixes in otherwise suffixing languages. It has also happened with subject pronouns in Retuarã: it is the only Tucanoan language which marks subject pronouns as prefixes on the independent verb (Strom forthcoming).

An example of morpheme loss or addition can be seen in the difference between Tucano and Carapana past tense markers (discussed in section 4.3.3.2.1). Carapana has a morpheme -ya which marks historic past tense (one of three past tenses in Carapana). Tucano distinguishes between only two past tenses. This morpheme (and tense) has either been added in Carapana or lost in Tucano.

The Tucanoan CADA program handles morphological change through a rule mechanism which allows the linguist to specify morphological insertion, deletion, substitution, reordering, or coalescence. These rules are applied during the synthesis phase.

1.3.3 Lexical Change

Lexical change generally comes about through contact with other languages. This change includes borrowing words from neighboring languages (such words may eventually

replace the original words), or inventing new terms to express new ideas or concepts which did not previously exist in the language.

Some lexical change comes about through a semantic shift which renders the previous word unacceptable, as in the case of the Spanish word siniestra 'left (direction),' which came to have the meaning 'sinister, evil,' and was replaced in its meaning as 'left' by the word izquierda, an Iberian loan-word.

The Tucanoan CADA program handles lexical change through root substitution tables which apply during the transfer phase.

1.3.4 Syntactic Change

Transformational generative linguistics holds that proper domain of linguistics is the study of competence, what an idealized speaker-hearer must know about his or her language in order to generate all and only the grammatical sentences of the language (Chomsky 1965). This theory of linguistics has its basis in the study of formal or mathematical models of languages and grammars. Chomsky is credited with some definitions in the early studies of formal languages (a hierarchy of grammar types is known as the Chomsky hierarchy (Hopcroft and Ullman 1979)).

What a speaker knows about the language includes a rule component which generates underlying structures; a lexicon which contains phonological, syntactic, and semantic information which determines lexical selection and the semantic interpretation of the word in the sentence; a semantic component which assigns a meaning to the lexical and larger structures; a phonological component which assigns a phonetic representation to the sentence; and a transformational component which performs operations on the underlying structure to arrive at the surface structure which is actually written or spoken.

Antilla's (1972) discussion of grammar change focuses entirely on analogical changes: irregular word forms tend to be regularized by analogy to the regular word forms. Both forms may exist together for a while until one clearly wins out in common usage.

The Tucanoan CADA program deals only with changes on or below the word level. Syntactic changes are left to the human post-editor. However, syntactic differences between Tucano and Tuyuca are minimal, so the lack of a mechanism for dealing with syntactic change has not presented a significant restriction for this adaptation effort.

1.3.5 Semantic Change

Semantic change involves a change in the mapping between meaning and the lexicon. This can either take the form of semantic shift, or may involve a narrowing or broadening of the range of meaning that the word has. An English example of semantic shift is the word gay, which formerly meant 'happy or brightly colored,' but which has come to mean 'homosexual' through its use by that community to refer to themselves. In the case of narrowing, a word loses the larger range of meaning and refers to a subset of its former meaning. Broadening involves a change to include a range of meaning which was previously not included.

The Tucanoan CADA program makes no attempt to understand the text being adapted; therefore semantic factors are not considered. Cases of semantic difference are generally treated as lexical change, and are handled by substitution or left to the human post-editor.

1.4 Purpose Statement

This study considers the linguistic bases for an experimental computer program to adapt text material from one member of the Tucanoan language family to another, and compares this work to other MT systems. This computer program has been successful because it is based on sound principles of cross-language equivalence between closely

related languages. It operates under relatively modest goals. Rather than attempt to do everything required for translation from one dialect to another, the focus has been on the systematic lexical and morphological differences between the languages. This focus has allowed the computer program to introduce approximately 80-90% of the required changes between the two dialects, and leaves the rest of the decision making to a human post-editor, who also makes any stylistic changes which may be desirable. These underlying goals, though modest, have resulted in a system capable of greatly facilitating the work of producing usable, initial draft material in a new dialect.

The Tucanoan language family, in which this experiment was carried out, consists of approximately twenty-two dialects, spoken by small groups of people living in southeastern and southern Colombia, northwestern Brazil, northern Peru, and northern Ecuador (Grimes 1984; Key 1979; Loukotka 1968; Tovar and Tovar 1984). Significant linguistic and sociolinguistic features of the Tucanoan languages will be discussed in some depth in Chapter 4.

The Tucanoan languages are divided into three branches, Eastern, Middle, and Western. The Eastern branch of this language family was selected by the Colombia Branch of the Summer Institute of Linguistics as a potential target for

experiments in CADA because of the relatively large number of languages involved, providing a potentially greater pay-off than other possible language families if the experiment were to prove successful. The author served as a computer consultant to the linguists already involved in traditional translation projects for these languages. Weber and Mann (1980) describe a manual experiment which helps to determine whether a computational approach to adaptation between a dialect pair is feasible.[1] This manual experiment indicated that there was a reasonable hope for success in the Tucanoan languages, so a computer program was developed to support these traditional translation projects, with no consideration for any potential commercial use.

1.5 Overview

The experimental work presented in this study consists of a computer program that adapts text material from one dialect to another by analyzing the SL text and synthesizing the TL text, producing a rough-draft quality text which can then be manually post-edited. No restrictions are placed on the type of text material being adapted; it can be literary or procedural, complicated or simple.

This study will consider various areas that relate to the experiment.

Chapter 2 provides an overview of translation theory, and identifies where MT and MAT fits into the larger picture. An introduction to translation theory in general is important for a understanding of the limitations of the approach presented in later chapters. There is discussion of the usefulness of the approach despite the limitations.

Chapter 3 surveys MT, including a brief history. The discussion considers the various stages through which it has gone, along with an overview of the goals and methods at each stage. Since MT is a branch of artificial intelligence (AI), a general introduction to this field is considered relevant in order to provide a broader perspective. The discussion concludes with an overview of the state of the art in MT, and surveys methods of doing morphological parsing by machine. This discussion of AI and MT provides a framework with which the efforts of the CADA project can be compared and contrasted. By contrast to the goals of AI, which include natural language understanding, the approach taken by CADA is based on contextually defined string substitution of morphemes. This approach is justifiable due to the close linguistic relationship of the languages involved, and empirically recognizable results.

Chapter 4 describes in some depth salient features of the Tucanoan languages which crucially affect the success of

the experiment and points to problem areas for the program.

Chapter 5 describes the workings of the computer program, with examples of how the program operates on the data, and how the linguist can control the various aspects of the adaptation through manipulating control files.

Chapter 6 summarizes the results of this experiment, and outlines directions for further research.

Notes:

[1]The experiment consists of interleaving a text in a prospective SL with its translation in the prospective TL. The differences between the two texts are noted and classified according to type of difference: word order, morphological, phonological, lexical, or stylistic. If there is a large percentage of regularity between the two texts, and the differences are systematic and specifiable, the pair of languages is considered to be a possible candidate for CADA.

Chapter 2

TRANSLATION[1]

2.1 General Problem Statement

Translation theorists hold a variety of different views of what translation is, ranging from the point of view that translation is any change in the form of a text, to the view that it is specifically the process of taking a text originally produced in one language and recreating or reproducing its meaning in another language.

This latter view naturally raises the question of when two dialects of a language become different enough from each other to be considered different languages. This study will not attempt to answer this question, but will proceed on the assumption that someone decides that comprehension by members of one speech community of textual material in the speech of another community is undesirably low, and that some kind of transfer or translation process is desirable to improve comprehension (cf. Steiner 1975:31).

This study will not address the issue of translatability, of whether translation is in fact possible, because the issues usually considered in discussions of translatability (great cultural differences and concepts not known by the target language and culture, e.g. translation of snow

into an Amazonian jungle language) are not very relevant when dealing with the transfer of text between two similar languages spoken by peoples of similar cultural backgrounds living in similar ecological environments (see section 1.1.2.3). The question of translatability is primarily a theoretical issue that exerts its influence in the area of whether certain goals in translation are attainable rather than in the actual impossibility of translation as a whole. The goals of dialect adaptation are too low-level to affect the philosophical issues raised by this question.

It is fundamental in any discussion of MT to define what is meant by the term **translation**, and to determine whether (or to what extent) the process of adapting textual material in one language to another language is a special case of translation. As the following discussion will attempt to show, **translation** is a very difficult term to define, since it means so many different things to different people. The position that this study will take is that adapting texts between two languages is a form of translation based on lexical and morphological equivalence, and supporting evidence for this position will be developed below.

2.2 Definitions of Translation

The many different views regarding the nature of translation reflect in part the way the authors who hold them define "translation." These definitions comprise three major groups:

1. translation involves a change in the form of a text,
2. translation involves a transfer of meaning,
3. translation involves the replacement of a text by an equivalent text.

Most authors require that translation involve two or more languages, while others allow for translation between forms within a language, i.e., written to oral form, or oral to kinesic (such as signing for the deaf).

The discussion of translation fits into a more general theory of human communication, whether intralingually or interlingually. Nida (1964) outlines five phases involved in any communication:

1. the content or subject matter;
2. the participants -- speaker or writer and the intended audience;
3. "the speech act or the process of writing;"
4. the language, with all its resources, used to encode the communication; and

5. the message itself (Nida 1964:120).

The translation process is crucially affected by all of these phases, while striving to produce the best fitting message in the TL. The difficulty faced by a translator is that each of the various phases poses its share of problems for the translation task, problems which interact with problems posed by other phases.

2.2.1 Change of Form

Hjelmslev is one of few authors who views translation as any change of form or "physiognomy"; from handwriting to voice, person to person, language to language (Hjelmslev 1970:135).

Steiner states that "a human being performs an act of translation, in the full sense of the word, when receiving a speech-message from any other human being" (Steiner 1975: 47). He goes on to say that

> interlingual translation is the main concern of this book, but it is also a way in, an access to an inquiry into language itself. 'Translation', properly understood, is a special case of the arc of communication which every successful speech-act closes within a given language. On the inter-lingual level, translation will pose concentrated, visibly intractible problems; but these same problems abound, at a more covert or conventionally neglected level, intra-lingually.... In short: inside or between languages, human communication equals translation. (1975:47).

This quotation views translation as an integral part of communication, whether between two languages or within a language. It seems that Steiner includes within translation the pragmatic and interpretive aspects of communication.

Frawley defines translation as "recodification... -- a theory of translation is a set of propositions about how, why, when, where (...) coded elements are rendered into other codes" (Frawley 1984:160). This definition includes changes of codes or coding systems within a language or between languages, and his examples yield a definition similar to Hjelmslev's definition above.

2.2.2 Transfer of Meaning

The vast majority of the authors consulted in this study view translation as the process of transferring the meaning and, to a far lesser extent, the form of a text written in one language into another. Beekman and Callow (1974:19-20) are fairly typical in defining translation as a process involving "1) at least two languages (form), and 2) a message (meaning)" (cf. Brislin 1976:1; House 1977:25; Larson 1984:3). They use the term form to refer to the "formal linguistic elements of a language" (Beekman and Callow 1974:19-20), and meaning to refer to the message communicated by the form of the language (1974:20). Beekman also would include the function of the message as an

integral part of the message. He defines function as "the significance of, the reason for, the purpose of, the use or uses of an object or action" (Beekman 1965:37), in this case, a text or utterance.

Beekman and Callow are referring to linguistic meaning in this definition, but their book is very clear on the importance of translating the cultural, as well as the linguistic, meaning of a text. Linguistic meaning refers to the semantic content communicated by the lexical and grammatical structures of a language. Cultural meaning refers to the conceptual content that a communicator expects the intended recipients of the message to bring to bear in understanding the message, including historical, sociological, and pragmatic factors which provide the context for the message.

Van Slype and his coauthors (1983) give two working definitions of translation, the traditional and the modern:

> -- the "traditional" definition: the process of replacement of a text written in a source language by a text written in a target language, the objective being the maximum equivalence of meaning.
>
> -- the "modern" definition: the process of transfer of message expressed in a source language into a message expressed in a target language, with maximization of equivalence of one or more levels of content of the message: i.e referential (information for its own sake, e.g. organization note), expressive (centred on the sender of the message,

> e.g. speech), conative (centered on the recipient, e.g. publicity), metalinguistic (centred on the code, e.g. dictionary), phatic (centred on the communication, e.g. courtesies), poetic (centred on the form, e.g. poetry). (van Slype et. al. 1983:33).

The traditional definition corresponds to the definitions of translation discussed elsewhere in this section, while the modern definition relates to the purpose or function of the translation, as discussed in section 2.3.2.

2.2.3 Reproduction of an Equivalent Text

Nida and Taber demand that "The translator must attempt to reproduce the meaning of a passage as understood by the author" (1974:8). Whether such understanding is possible or not is a philosophical question beyond the scope of this study; however, since they are involved in translation work, they must believe that the the results of such an attempt are or can be close enough to justify the effort.

John Catford, while holding to the general definition that more than one language is involved, would not agree that it is a transfer of meaning. Within the systemic linguistic theory,

> "meaning ... is a property of a language.... [It is] the total network of relations entered into by any linguistic form -- text, item-in-text, structure, element of structure, class, term in system -- or whatever it may be" (Catford 1965:35).

Consequently, it does not make sense to talk about

transferring meaning between two languages, within a systemic framework. He defines it in functional terms: "Translation is an operation performed on languages: a process of substituting a text in one language for a text in another." (Catford 1965:1) or "... the replacement of textual material in one language (SL) by equivalent textual material in another language (TL)" (Catford 1965:20; also Pinchuk 1977). The key term in this definition is the term equivalent, as Catford points out: "The central problem of translation practice is that of finding TL translation equivalents. A central task of translation theory is that of defining the nature and conditions of translation equivalence" (Catford 1965:21). Catford mentions two aspects of equivalence: commutation equivalence is concerned with finding the TL form which can be substituted for the corresponding SL form in a given context; formal equivalence is concerned with finding constructions which function in the TL in the same way as SL constructions function in the SL.

The concept of textual equivalence can be viewed in many ways. For some types of textual material, such as assembly or maintenance manuals, what is important is the clear communication of the steps involved in accomplishing the desired procedure, i.e., only communicating the content or meaning of the text. For other types of textual material, it may be desirable to produce a TL text which has the

same effect and impact on the intended TL audience as the SL text has on its intended SL audience. This relates to what Nida has described as dynamic equivalence (Nida 1964). In this case, the translator may view the desired rhetorical effect as more important than general commutative equivalence: the clearest example of this type of translation is the translation of poetry, where feeling, rhyme, and meter may be more important that words or phrases which "mean the same" (Nida 1964). Although this type of translation emphasizes rhetorical effect, meaning equivalence is still involved.

2.3 Types of Translation

This general heading of types covers various parameters involved in translation. These include categories of form and function, as defined by Beekman in section 2.2.2. The category of form subsumes the distinction between literal and idiomatic translation. The category of function includes the reasons for translating, reasons which in large part determine the form that the translation will take.

Nida discusses three basic factors which account for most differences in translations:

1. "the nature of the message,
2. the purpose or purposes of the author and, by proxy, of the translator, and

3. the type of audience" (Nida 1964:156).

Messages can place importance on content or form. Poetry, for example, places importance on form, and the translation of poetry should take this into consideration, while sacrificing as little of the content as possible.

The purpose of the author may be to entertain, communicate information, give procedural instructions, or several of these at the same time. The purpose of the translator may be the same, or it may be to communicate how other people think, what they consider entertaining, or important, etc. The way a translation is done should reflect the purpose for which it is done -- the form of the translation will reflect its purpose.

2.3.1 Form

The parameter of form is the one that generally comes to mind when one thinks of different kinds of translation. This parameter is a continuum with two poles. Beekman and Callow (1974) call these two poles literal, in which the form of the TL text conforms closely to the form of the SL, and idiomatic, where the general linguistic form of the TL text conforms more to the form of the TL. Literal and idiomatic should be seen as poles on a continuum from strictly literal, such as an interlinear gloss, to free, in

which the original form is virtually ignored.

Nida (1964) refers to these two poles as expressing formal and dynamic equivalence. Formal equivalence is concerned with the form and content of the message -- trying to produce a TL message that has, as far as the TL allows, the same form and content as the SL message. Dynamic equivalence aims at recreating the message of the SL in the forms and context (social and cultural, as well as linguistic) of the TL (Nida 1964:159). Again, these types of equivalence are poles on a continuum -- current opinion preferring the dynamic equivalence end (Nida 1964:160).

The form of a translation will depend largely on its purpose or function. Translation which is intended to bring about changes in thoughts or behavior may emphasize the form which will have most impact on the intended audience -- an idiomatic translation focusing on culturally emotive expressions and styles which will bring about the desired emotional effect. Some cultures may use stories or parables as a primary tool for teaching desirable character qualities or behavior, while others might prefer a more direct, hortatory or imperative style. Factors such as these will affect the form that the translation takes.

Literal translation (formal equivalence) focuses more on the SL, and it is useful in terms of understanding the social, historical, and cultural context of the original message.

Idiomatic translation (dynamic equivalence) pays careful attention to both languages, selecting the TL forms which best express or recreate in a form meaningful to TL speakers the social, historical, and cultural context of the message.

Burton Raffel gives an interesting analogy in his discussion of a literal versus a free translation of poetry:

> The literalist assumes that his job is to act as a kind of inverse mirror: like Alice, his translation is intended to take one through the looking glass back into the original poem--or as close to the original as may be possible, given linguistic, cultural, and personal differences. The 'free' translator assumes that his job is to take the poem out through the mirror, bring it from its original environment into the world of those who read whatever language he is translating into. ... the translator's task is to recreate, for someone <u>without the linguistic ability to do the job himself</u>, a pre-existing poetic experience (Raffel 1971:11). [italics added].

With this in mind, it is incumbent on the translator to determine his or her reasons for translating the particular work, and consider the implications of the type of translation for the intended reader or listener.

2.3.2 Function

Casagrande (1954) discusses four categories of purpose or function in translation. There is a close relationship between the kind of material to be translated and the purpose for translating; in some cases, the kind of material determines the purpose. The categories of purpose discussed are pragmatic, aesthetic-poetic, linguistic, and ethnographic. Some texts may be translated with several purposes in mind, and the purpose may determine the form of the translation.

Pragmatic translation has the goal of communicating clearly the content of the source text, without any significant attention to style or other similar factors. This is the type of translation that would be done, for example, on technical assembly or repair manuals where the only interest is in communicating the required procedure (Brislin 1976:3).

> **Aesthetic-poetic** translation
>
> takes into account the effect, emotion, and feelings of an original language version; the aesthetic form (e.g., sonnet, heroic couplet, dramatic dialogue) used by the original author; as well as any information in the text (Brislin 1976:3).

This purpose of translation in the most difficult to achieve satisfactorily, because it is the most rigidly constrained. Discussions of translatability often focus on the difficulty or impossibility of achieving this type of translation for

certain texts.

Ethnographic translation involves a consideration of the cultural factors involved in expressing SL cultural factors in appropriate TL terms -- creating a TL cultural context which corresponds to the SL cultural context. It has a "concern for the social situation. It also introduces the importance of context in translation..." (Brislin 1976:3-4). Casagrande's ethnographic translation corresponds to Nida's concept of dynamic equivalence in translation (Nida 1964). It tries to make a foreign text understandable in cultural terms relevant to the TL and target culture, if necessary using cultural equivalents to recreate in the target culture the meaning and impact that the SL text had for the source culture. This may also involve extended commentary explaining the meaning or background for a SL term. Ethnographic translation also serves to distinguish between concepts which are similar yet not identical between the SL and TL (Casagrande 1954:336).

Linguistic translation deals with such things as linguistic correspondences between the SL and TL. It is "concerned with 'equivalent meanings of the constituent morphemes of the second language' and with grammatical form" (Brislin 1976:4). Translations produced with this purpose in mind include literal translations and interlinear

glosses. It is most useful in determining lexical and grammatical structures in the SL, and in determining the ways SL structures correspond to TL structures.

2.4 Translation Theory and Machine Translation

MT can, in principle, be programmed to be any of the kinds of translation outline in this chapter, except, perhaps, aesthetic-poetic. It should be obvious that the more literal approach to translation is easier to both design and implement. This is the approach taken by the Tucanoan CADA project: linguistic translation, based on lexical and morphological equivalence between closely related languages (cf. Tosh 1965). This approach is possible using the property of commutative equivalence (Catford 1965), and can be based on substitution tables involving no direct reference to semantic or pragmatic considerations. All the other approaches require an understanding of the text being adapted or translated, so are beyond the scope of the approach taken by the Tucanoan CADA project.

In CADA, the literal approach used does not necessarily suffer from the drawbacks outlined in section 2.3.1 because the languages <u>are</u> very similar in lexical and morphological structure and, hopefully, semantics. The languages are spoken by people sharing many cultural, religious, and social features, thus reducing the problems involved in

recreating the context of the SL.

The most important factor involved in producing successful results from CADA is the quality of the original SL text, since, for all applications to date, this SL text has itself been a translation from another language -- the texts processed to date by the Tucanoan CADA project are books of the Bible. This factor, though crucial, is outside the scope of the CADA effort itself.

Notes:

[1]I thank Dr. Ellis Deibler for reading and commenting on an earlier draft of this chapter.

Chapter 3

REVIEW OF THE LITERATURE

3.1 Artificial Intelligence Research

Since the early days of computers, there has been an interest in using them to solve symbolic as well as numeric problems. This interest led to research aimed at developing artificial (or machine) intelligence (AI). Barr and Feigenbaum give the following definition of AI:

> Artificial Intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior -- understanding language, learning, reasoning, solving problems, and so on. Many believe that insights into the nature of the mind can be gained by studying such programs. (1981:3).

AI research has produced programs, techniques, and concepts that have proven useful in many areas of computer science, as well as providing tools that are in everyday use in a number of different academic and business applications, tools that can solve problems in chemistry, physics, geology, medicine, and industry at or beyond the level of expertise of human experts in those fields (Barr and Feigenbaum 1981; Rich 1983; Charniak and McDermott 1985; Mishkoff 1985; Peat 1985). These AI tools are known as <u>expert systems</u>. The area of most rapid growth within AI in the mid-1980's has been the development of such tools for many different application areas. This growth is evidenced by

the explosion of articles on the subject in the business and popular literature in 1985-1986.

Another area of interest to AI researchers pertains to problems dealing with robotics: motion, vision, perception, understanding commands, etc. (Winston 1984; Charniak and McDermott 1985; Peat 1985; Rich 1983). There is some research in developing speech recognition and generation (synthesis) systems for use in robots so that the human controlling the robot could just speak to it and the robot could generate and answer appropriately in natural language, as well as perform the proper action. Natural language understanding and answering in robotics is a subset of the area of AI research most closely related to the topic of this chapter, natural language research in general.

The Japanese Fifth Generation computer project is currently attempting to develop a large-scale AI system to interact with humans on human terms. The proposed goals include the ability to use speech, graphic, image, and document input and output; generalized conversational interaction between the computer and the human; the ability to learn, associate, and infer facts from information supplied by the human or from specialized data bases; provide automatic programming services based on specification of problems so that humans will not need to relate to the computer

on its terms; machine translation (including conversational interpretation between languages); question answering systems; problem solving systems, etc. (Moto-oka et. al. 1982:13).

Problems for which AI solutions have been attempted differ from other problems in computer science in interesting ways. Most problems with traditional computational solutions have been numerical: the computer has been used to solve problems beyond the reach of human capabilities due to the sheer volume of computation required, as well as the need for very fast computation in changing environments. Typical problems of this kind include applications in physics, engineering, meteorology, as well as statistical studies of various kinds.

AI solutions may involve mathematical computation of information needed for solutions, but the solutions are reached through the application of specific knowledge about the problem domain. This knowledge may be encoded in the form of rules, such as syntactic rules for natural languages, or inference rules for problem solving, but the solutions are knowledge-based, rather than numeric. Accordingly, AI research has greatly contributed to research in search techniques (dealing efficiently with the extremely large combinatorial search spaces possible in some problem

areas), knowledge representation techniques, formal logic systems, and modeling techniques for other disciplines, most notably psychology, with its emphasis on learning and perception.

3.2 Natural Language Research

3.2.1 Goals

Natural language parsing and understanding systems were among the early areas of research within AI. Interest in machine translation (MT) was expressed in the letters and publications of Warren Weaver and Andrew Booth as early as 1946 (see section 3.3), while Booth and Locke (1955) mention the need for speech recognition and generation systems in order to simplify human interaction with computers.

Current natural language research within the AI framework is continuing in many directions:

1. designing and implementing question answering systems for various languages,
2. machine translation,
3. speech recognition,
4. speech synthesis,
5. text understanding and analysis (whether from written or spoken text),
6. text generation (generating texts from abstracts or generating cohesive text, including answers to data

base queries) (Mann and Matthiessen 1983; Mann 1981, 1983; Appelt 1985),

7. text abstracting and summarization (Eibl 1985; Tait 1985), and
8. natural language command interpreters to simplify human interaction with data bases (Eibl 1985).

These research areas encompass many aspects of language: from recognizing and producing the phonetic speech signal to processing the semantic and pragmatic information required for natural human communication (including discourse factors such as cohesion, pronominal reference, and implied information), as well as sentence level syntax, semantics, and phonological signals such as intonation contours. Research in these areas has come a long way, but much remains to be done.

Eibl (1985) briefly discusses a data base query system called SPICOS, currently under development as a joint project of Siemens and Philips, which aims at understanding and answering spoken questions, either through spoken output or by displaying the answers on a screen.

Peter Muhlhausler's comments about the applicability of theoretical linguistic findings to real-world problems in AI may be worth noting here:

> ... development in linguistics [since the Chomskyan revolution has exacted a] cost for its progress, mainly that language ... has become an abstract object (in spite of minority movements in natural phonology, morphology or semantax) removed from both its speakers and from the use speakers make of it in concrete situations. The widening gap between linguistic theory and the real world accounts for the irrelevance of most of the findings of theoretical linguistics to any applied area and has furthermore made most claims impervious to empirical testing (Kreckel 1981:v).

Chomsky's research in formal languages and the incorporation of this research into the linguistic theory of transformational-generative grammar provided an initial framework for building rigorous descriptions of natural languages. The claim was made that this rigorous formal system modeled the knowledge (competence) that an idealized speaker-hearer had of his or her language. A transformational-generative grammar of a language attempted to describe the formalism necessary to generate all and only the grammatical sentences of that language. Competence was contrasted with performance, how language is used in everyday speech situations. Performance limitations, such as ungrammatical constructions, slips of the tongue, false starts, memory limitations, etc., were considered irrelevant (Chomsky 1965). However, it is precisely these areas, plus intersentential references, that provide the most problems for natural language processing and understanding by computer programs. These programs, to be useful, must deal with language as it is used (performance); the reason for using a natural language processing

system is to have the computer understand what the human is saying in order to get the right results, whether these be answers to questions, summary information, or appropriate movement by robots.

3.2.2 Current Research

Human use of natural language appears simple; after all, children of three or four years of age can talk fairly well, expressing their desires and feelings over a fairly wide range of subjects.

Natural language research in AI represents an attempt to build systems that can understand and communicate with humans using natural language. There is no a priori claim that these systems model the way humans use language to communicate, although if the resulting system may be shown to do so, so much the better. Friedman (1971) expresses for computational linguistics the approach that AI researchers take in general:

> The computer scientist welcomes whatever insights about natural language are available from pure linguistics as potentially useful in his attack on the problem; however, he does not feel limited to the standard linguistic methods (1971:719).

The natural language interests of AI overlap with many other disciplines, including philosophy (semantics, pragmatics, logic), psychology (perception, vision, understanding,

cognition), linguistics (acoustic phonetics, parsing, formal grammars, discourse), computer science (formal languages and language theory), and engineering (designing "machines who think"[1]).

Natural language research has contributed to an understanding of analysis (parsing) strategies which have had applications in computer science (such as compiler theory) as well as natural languages (Hopcroft and Ullman 1979).

Several parsing schemes have proven effective for natural language syntax: rewrite rules (production systems) (Rosner 1983), chart parsers (Kay 1977; Tennant 1981; Varile 1983), formal logic (Winograd 1983), and augmented transition networks (ATN) (Johnson 1983). Each of these is capable of easily building syntactic structures. In addition, the ATN formalism provides a framework on which to build semantic information (Winograd 1983) and even discourse and pragmatic information (Reichman 1985). The literature on systems using the ATN formalism is extensive, with applications in understanding systems as well as parsing systems (Bates 1978; Bolc 1983).

3.3 Machine Translation Research

3.3.1 Brief History

The history of MT is an interesting one, reflecting the ideas of researchers in various disciplines. Some of these ideas demonstrate a notable misunderstanding of the nature of human language, while others indicate ignorance of the amount and kind of background information that a translator must bring to bear in understanding a message and making that message available in another language, as discussed in Chapter 2.

Electronic computers were invented in the late 1930's and became used extensively during World War II. The primary applications were in solving mathematical problems, such as artillery trajectories, but they also were used in cipher problems, in the areas of developing encryption schemes for coding and decoding secret messages.

Some of the researchers involved in these areas turned their attention to the possibilities of MT after the war. They believed that foreign languages were simply messages coded using different sets of symbols, and that the translation process merely involved developing a scheme for replacing the symbols used by one code scheme (i.e., a text in Russian), with the corresponding symbols from the decoding set (i.e., English) (Weaver 1955:18). Weaver presented his

ideas in a letter written in March, 1947, to Norbert Wiener, who expressed his opinion that the idea of MT was premature.

Weaver was also in contact with A. D. Booth, who suggested the possibility of a word-for-word translation that could handle some morphology, but ignored syntax. Booth and Richens were viewing the problem as one of dictionary lookup and substitution, with a morphological component, as early as 1948. In July 1949, Weaver distributed a memorandum called Translation, which was reprinted as Weaver (1955). This memorandum served to kindle interest in MT as a research activity in three universities: the University of Washington, the University of California at Los Angeles, and the Massachusetts Institute of Technology. In 1952, eighteen researchers were invited by the group at MIT to participate in several days of discussion of current research and future directions for MT. No formal conclusions were reached, but they agreed that enough was known about the linguistic and computational techniques that with further research in two areas, MT looked promising. These two areas still requiring research were:

1. statistical studies of word frequency and word translations, by language and scientific domain, and
2. syntactic studies to determine how much syntactic information was needed in order to be able to

design and build the equipment needed to do MT (Booth and Locke 1955).

What is especially interesting about Booth and Locke's (1955) introduction to their book is that their conclusions regarding directions for further research outlined some factors that have continued to influence research to the present time. These directions affect natural language research in general, not just MT. The five factors that they mentioned were:

1. Adequate input-output. They included in this area the importance of scanning devices which could directly input from a printed page, as well as spoken input and output -- areas of active AI research today;
2. Large and cheap memory. In the mid 1980's the average home computer can make use of more memory than was conceivable on the machines of their day;
3. Suitable dictionaries, including specialized dictionaries by subject domain. This is the current state of the art in MT and machine-aided translation (MAT);
4. Inflectional endings. Early MT work attempted to have full-word dictionaries, but that proved unmanageable. It is still unmanageable because productive inflectional morphology can greatly multiply

the size of the dictionary if it is necessary to include all possible forms of a full word. In agglutinating languages, such as the Tucanoan languages to which CADA is applied, it is for all practical purposes impossible to generate all possible forms for each word. Attention to morphological details in general is important to any successful MT system, and newer systems are focusing more on this area;

5. Syntactic analysis. This is one area that was largely ignored by early systems, but its importance was quickly realized.

These five areas, plus the very important one of semantic information representation, are still of interest to practical MT and other natural language applications today.

MT research continued in various university settings, where there were two types of study being considered: the linguistic aspect and the computational aspect of MT. Lehmann (1959) saw as an important part of MT the study of comparative structures of languages, and the MT researchers at the University of Texas became involved with an attempt to describe the languages they were working with as formal systems.

Early MT research was based on the assumption that the translations produced by the machine must be high-quality translations done without human intervention -- the goal was to eliminate the need for human translators (Melby 1985). Until the early 1960's, there was a general feeling that high-quality MT was feasible, and that truly successful systems were imminent. Bar-Hillel was an early researcher in MT in the 1950s, but by the early 1960s he was a confirmed skeptic (Bar-Hillel 1964b, 1964c, 1964d). As early as 1951, he was aware of some of the problems facing MT that he believed would prove insurmountable, such as resolution of semantic ambiguity (Bar-Hillel 1964a).

The United States government was one of the major sources of funding for MT research in this country. In the early 1960s, the National Academy of Sciences commissioned the Automated Language Processing Advisory Committee (ALPAC) to research the state of MT. In 1966, ALPAC released a report which concluded that MT was not feasible nor cost-effective, and the sources of government funding mostly dried up (Hutchins 1984:94). Part of the problem with the ALPAC report was that it did not take into consideration the various purposes for doing translation -- the goal was seen to be fully automatic, high-quality translation produced entirely by the computer without human assistance. This goal was not achievable with the technology of that day, nor is

it yet within reach. Present MT efforts have the more modest goal of using the computer to produce a text which is post-edited by a human translator.

In spite of the negative report from the ALPAC, various agencies that were already using existing systems to translate scientific articles from Russian to English, continued to use them, despite the poor quality of results, because some understanding was considered preferable to no understanding at all (Hutchins 1984:103).

3.3.2 Approaches

Garvin (1972) presents three approaches to MT that describe how its proponents and detractors perceived translation:

1. Brute-force: "... given a sufficiently large memory, MT can be accomplished without a complex algorithm -- either with a very large dictionary containing not only words, but also phrases, or with a large dictionary and an equally large table of grammar rules ..." (Garvin 1972:10).

   This approach has produced translations of questionable quality.

2. Perfectionist: it is impossible to attempt a MT without perfect understanding of the source and target languages, and a mathematical model of the translation process between them. This approach would never get started.

3. Middle-ground: an engineering approach which views the problem as one of acquiring as complete and extensive knowledge of the languages involved as possible, and continuing to build that knowledge through a cyclic process of application, evaluation, and adjustment. This third approach is the only one that has a chance for success.

3.3.3 Current Research

MT hit its peak of popularity in the early 1960's, then went into a decline, as it was unable to live up to the expectations created by its proponents. The goal of early MT efforts was to automatically produce high-quality translation without the need for intervention by human translators. The problems of knowledge representation, the kinds of knowledge required to completely understand a text (extralinguistic social, cultural, and physical knowledge of the real world (Pollack and Waltz 1986)), and incomplete syntactic and semantic knowledge, as well as the ambiguities inherent in natural languages proved insurmountable for completely automatic, unassisted MT systems (Bar-Hillel 1964b, 1964c; Hutchins 1984).

The ALPAC report, criticized by many as biased, affected the prestige and funding of MT efforts world-wide. As a result, later MT research was virtually ignored by most

linguists, translators, computer scientists, and the general public. In spite of this, work continued in developing and improving translation aids, as well as searching for new approaches to MT that might overcome the problems facing the early systems. One of these approaches was to fine-tune the system to produce the best results possible by machine, then submit these results to a human translator who would use the machine output as a rough draft text to be corrected and revised -- this is the approach taken by the CADA project (cf. Melby 1985, Bar-Hillel 1964b). Post-editors were used in early systems, but they were seen as a necessary evil, an indication of the shortcomings of the system, rather than viewing the MT system as a tool for translators to use, as described by Melby (1985).

The economic necessity for reducing the cost of translations required by such multinational corporations and governments as Siemens in Germany; Fujitsu, Hitachi, Toshiba, and NEC in Japan; the European Community; Japan; and Canada have provided the continued impetus and funding to continue to search for solutions in this area. At the same time, giant strides forward in computational capabilities (both hardware and software), coupled with linguistic and semantic theories sufficiently rigorous to be implemented in a computer, have allowed useful systems to be designed and implemented.

Systemic linguistics (see Berry 1975, 1976), developed by Firth and Halliday in England, provides the theoretical basis for research by Mann's group into text generation strategies (Mann and Matthiessen 1983; Mann 1981, 1983).

Junction grammar (see Bush 1976), developed by Lytle at Brigham Young University, provides the theoretical basis for the MT research carried out by the Translation Institute at that university, as well as a commercial outgrowth of this research, a CAT system called ALPS (Automated Language Processing Systems) (Slocum 1985:10). This system is similar to the one described by Melby (1985), which is of sufficient interest to be discussed below.

Lawson presented a paper to a translator's conference in 1981, stating that "Translation by computer is now a reality, with perhaps 30 machine translation (MT) systems in regular use around the world." (Lawson 1982:v). One of the most successful of these is the METEO system, used daily by the Canadian government to translate weather reports from English to French (Isabelle and Borbeau 1985; Melby 1985). A large part of its success is due to the narrow subject domain. Attempts by the group (Traduction Automatique Université de Montréal (TAUM)) that developed METEO to expand the system to translate aviation maintenance manuals has produced useful, though not cost-effective, since the cost

to revise MT output is higher than the cost to revise human translations (Isabelle and Borbeau 1985:25).

The impact that MT and MAT systems have had on translators has been varied. Eibl states that the MT system used by Siemens "will not replace the translator but it will probably change his job description" (1985:105).

Melby (1985) describes a three-level CAT system currently being developed as a research project at Brigham Young University. Level one is a word processor with access to a terminology data base, abbreviation expansion capabilities, and communication capabilities for accessing other translators on computer networks.

In level two, the system suggests possible translations for the source text. The translator is free to select from among the alternatives presented, or disregard them altogether. The translator can add or change information in the terminology data base as desired. Sharing terminology data bases among translators working on similar material allows more consistent translation. Level two also provides spelling checking capabilities while the translation is being done.

Level three adds a MT component. The translation is done automatically by an offline MT system which keeps track of the source text, translated text, and problem areas (syntactic constructions or lexical omissions). For each segment of text (typically a sentence or two), a level of tolerance is assigned by the MT system, depending on problem factors. The translator sets the desired minimum tolerance level, and interacts with segments produced by the MT system meeting these tolerance requirements. The segment can be incorporated as is, edited, or disregarded. In any case, the translator remains in complete control of everything in the target text -- the translation aids at any level are available to be used or ignored at the translator's discretion. As a result, the computer is a tool for the human translator, instead of the translator being a tool for the computer.

Although most definitions of translation given in Chapter 2 view translation as a transfer of meaning, only one MT system attempts to translate based on an understanding of the SL text (Wilks 1983). All others use as little semantic information as possible to disambiguate parsing problems that cannot be solved any other way (Hutchins 1984:126). Semantic information used by MT systems includes such categories as animacy, humanness, etc. Case frames are also used (Hutchins 1984; Samlowski 1976).

3.3.4 Method

The general approach to MT involves two or three phases: analysis, transfer, and synthesis. Twophase systems generally are designed for specific language pairs; the analysis phase produces an intermediate form which contains some of the information required to transfer to the specific TL, and the synthesis phase derives the TL-specific information from the intermediate form. The different systems vary considerably as to the inner workings of each phase, i.e., the amount of syntactic and semantic information used, internal representation, morphological parsing or whole-word dictionary search, intermediate form, etc. (Hutchins 1984).

3.4 MT and CADA

The approach taken by the CADA project involves three primary phases: analysis, transfer, and synthesis (see Chapter 5 for fuller discussion). The analysis phase does morphological parsing on a word-by-word basis -- no syntactic or semantic information is used. The intermediate form consists of all possible analyses of the word; there will be multiple analyses if the word is ambiguous. The transfer phase does lexical and phonological substitution to derive the TL root, and the synthesis phase deals with the required morpheme selection criteria in order to produce the TL word.

Notes:

[1]To quote the title of a book by McCorduck (1979).

Chapter 4

THE TUCANOAN LANGUAGES[1]

4.1 Demographic Information

The Tucanoan languages are spoken by some twenty-two different groups of people living in the tropical rain forests of southeastern and southern Colombia, northern Ecuador, northern Peru, and northwestern Brazil. Most of the groups are rather small, ranging from 180 to 4000 speakers. Grimes (1984) provides specific information on the population and geographical location of each of the various groups: Figures 4.1 and 4.2 are maps reprinted by permission from Grimes (1984).

Tucanoan languages are divided into three main branches: Eastern, Middle, and Western Tucanoan (Waltz and Wheeler 1972:128-129). The Eastern Tucanoan branch of the family has about fifteen languages or dialects, in three subgroups, while Middle has one or two (the classification of Rẽtuarã is unclear), and Western has about five, in two subgroups. The CADA experiment described by this study involved several Eastern Tucanoan languages: Tucano served as the SL, and Tuyuca as the primary TL. Some apparently promising results were also achieved for Yurutí as a second TL, (by comparing the computer-adapted output to a word-for-word translation done by a Yurutí man who was also fluent in Tucano), but a

Figure 4.1 General Geographical Location of Tucanoan Languages

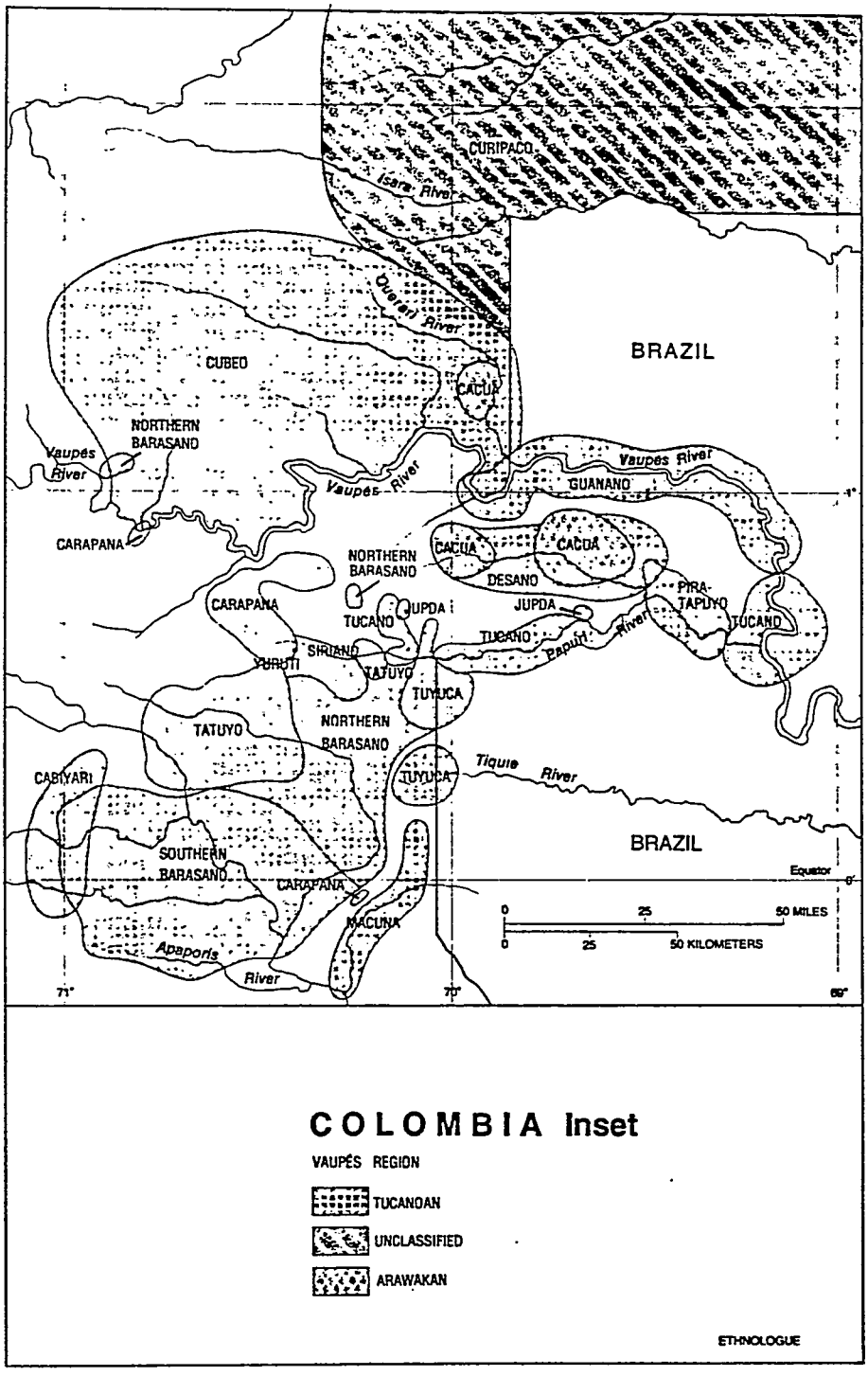


Figure 4.2 Detailed Geographical Location of Tucanoan Languages

detailed evaluation by the linguist working with the Yuruti was not possible.

An approximate classification of the Tucanoan languages is shown in Table 1. This table follows Waltz and Wheeler (1972:128-129), with some additional languages included which were not found in their classification. For these languages, I have followed Malone (personal communication).

TABLE 1

CLASSIFICATION OF TUCANOAN LANGUAGES

BRANCH	GROUP	LANGUAGE
EASTERN	NORTHERN	Tucano[2] Guanano Piratapuyo
	CENTRAL	Bará and proximates Bará (Northern Barasano)[3] Tuyuca[2] Yurutí[2] Pápiwa Desano and proximate Desano Siriano Tatuyo and proximate Tatuyo Carapana
	SOUTHERN	Macuna Barasana (Southern Barasano)[3]
MIDDLE		Cubeo Retuarã[4]
WESTERN	NORTHERN	Coreguaje Siona and proximate Angutera Macaguaje Secoya Siona Teteté
	SOUTHERN	Orejón

4.2 Sociolinguistic Information

The language (or dialect) identity of an individual is interesting from a sociolinguistic perspective. Tribal divisions align fairly closely with dialectal divisions. This is in spite of the fact that each of the dialect groups of at least the Eastern Tucanoan languages practices exogamy (except Macuna and Cubeo); members of these dialect groups must marry outside their group. Women of other language groups move to the village of their husbands. It is therefore the norm that any given village will have speakers of several languages represented.

Extended family members have traditionally lived together in a large communal house called a maloca (or long house, following Christine Hugh-Jones (1979)), although the use of the maloca is dying out in some areas which have had more contact with Western culture; in other areas the maloca is used only for ritual dances and ceremonies. As a result of living together in the same house or village with close relatives speaking other languages, a child grows up at least bilingual, with the father speaking one language, the mother speaking another, and possibly aunts speaking a third or fourth. The child is identified by the language and group of the father, although he or she may be fluent in several of the languages (cf. Tovar and Tovar 1984:157).

4.3 Linguistic Information

4.3.1 Phonological Characteristics

Phonological characteristics of the various languages are relevant to the CADA program in several ways. Some of the lexical and morphological differences between languages relate directly to differences in phonological systems, such as the loss of the glottal stop in Tuyuca and Yurutí, producing lexical items which differ between languages solely by the presence or absence of the glottal stop.

Phonological environments are necessary to restrict the occurrence of allomorphs during the analysis or recognition phase, and to provide criteria which are used to select the correct allomorph during the synthesis phase. These environments include preceding characters, which govern the recognition or selection of allomorphs indicating vowel harmony or geminate vowel reduction; preceding nasal morphemes, which indicate the need for conditionally nasalized allomorphs of following morphemes; and word boundaries, which determine wordfinal allomorph selection -- Tucano morphemes ending in a glottal stop lose the glottal stop if that morpheme occurs word-final.

Orthographic considerations are relevant because the form being analyzed (recognized) is written in the practical orthography, rather than phonemically. The synthesized form

is generated in the practical orthography of the TL. For clarity, all examples in the following sections will be presented in phonemic form, which will not coincide with the sample data in the appendices, which is in orthographic form.

4.3.1.1 Vowels

The Tucanoan languages, whether Eastern, Middle, or Western, with the sole exception of Retuarã, have six vowels, shown in Table 2, which are either oral or nasalized (see discussion of nasalization in section 4.3.1.3).

TABLE 2

TUCANOAN VOWELS

	-back	+back	
		-round	+round
+high	i	ʉ[5]	u
-high	e	a	o

4.3.1.2 Consonants

The consonant phoneme inventories of the various Tucanoan languages are far more varied than the vowel phoneme inventories are.

All Tucanoan languages have at least two of the following three series of stops: bilabial, alveolar, and velar. For most languages there is a voiced versus voiceless contrast for these stops, but by contrast Coreguaje has an aspirated versus unaspirated contrast (Gralow, Cook, and Young 1984). Some languages do not have complete series: Barasana does not have /p/ except in loan words (Smith and Smith 1976); Carapana has a limited distribution for /d/ (Metzger, personal interview); Cubeo (Salser and Salser 1976) has a voiceless alveopalatal affricate in the place of the voiced velar stop, and Retuarã (Strom forthcoming) has a voiced alveopalatal affricate in the place of the voiced velar stop.

Tucano and Guanano have a third series of stops, giving them a contrast of aspirated versus unaspirated voiceless versus voiced in the bilabial, alveolar, and velar positions (West 1980; Waltz and Waltz 1979).

Of the Eastern Tucanoan languages, Tucano, Guanano, Piratapuyo, Siriano and Desano have a glottal stop which serves as a contrastive consonant. This includes all the languages of the northern branch, and some of the languages of the central branch. Neither of the languages in the southern branch has a glottal stop phoneme.

In the Middle Tucanoan languages, Retuarã has the glottal stop and Cubeo does not have it, while both Siona and Coreguaje of Western Tucanoan have it.

The semivowels /w/ and /y/ are found in almost all of the languages: Retuarã, Cubeo and Coreguaje do not have the /y/, which in the case of Retuarã and Cubeo may coincide with the alveopalatal affricate filling the function of the missing voiced velar stop /g/.

All the Tucanoan languages except Siona have an alveolar flap phoneme, usually [r], alternating with the lateral flap [l]. The flap phoneme usually has a nasal flap allophone in nasalized morphemes.

Almost all the Tucanoan languages have one sibilant phoneme /s/. In Tatuyo, Bará, and Cubeo, this phoneme has merged with /h/. In addition to the /s/, Guanano also has a sibilant affricate /ch/, with the phonetic manifestation [c].

The Middle and Western Tucanoan languages have a larger phoneme inventory than the Eastern Tucanoan languages have. Cubeo has a voiced interdental fricative /đ/ that none of the other languages share. Coreguaje has several voiceless continuants that none of the others share, /W/ and /Ñ/, and

the voiced nasal stops lack the voiced oral counterpart found in the other languages. Coreguaje also has a voiced alveopalatal affricate /j/. Siona has a voiced sibilant fricative /z/, a voiceless alveopalatal affricate /ch/, and a complete set of labialized velar obstruants /k^{wh}/, /g^{w}/, /h^{w}/, which none of the other languages have. Wheeler's phonemic statement indicates a contrast between voiced oral and nasal stops, which are allophonic variations in the other languages.

4.3.1.3 Nasalization

Nasalization functions on a morpheme level, rather than on the segment or syllable level found in most non-Tucanoan languages. For this reason, it is often analyzed as a prosody of nasalization functioning on a morpheme level, rather than a feature of the vowel itself or the influence of an underlying nasal consonant (Kaye 1971).

Morphemes are of three types with respect to nasalization: intrinsically oral, intrinsically nasalized, or conditionally nasalized. West (1980) has charts identifying 20 intrinsically nasalized morphemes, 43 intrinsically oral morphemes, and 24 conditionally nasalized morphemes. Nasalization is progressively assimilated through a word, beginning with a nasalized root, or with the first suffix which is intrinsically nasalized, and spreading through the rest

of the word, or until it is blocked by an intrinsically oral morpheme. Consider the Tucano conditionally nasalized morphemes -gʉ 'masculine nominalizer,' and -re 'focus or accusative.' When it follows the nasalized morpheme /bõ/ [mõõ] 'not have,' the following forms occur:

(1) bõõ-gʉ̃-rẽ
not have-masc. nom.-focus

versus the oral forms following an oral root:

(2) co'te-gʉ-re
care for-masc. nom.-focus

Notice that the -re assimilates the nasalization of the preceding morpheme -gʉ, which assimilates the nasalization of the root preceding it. This is true for Tucano (West and Welch 1979; West 1980), Guanano (Waltz and Waltz 1979), Piratapuyo (Klumpp and Klumpp 1973), Bará (Stolte and Stolte 1976), Tuyuca (Barnes and Silzer 1976), Desano (Miller 1976; Kaye 1971), Siriano (Nagler and Brandrup 1979), Carapana (Metzger and Metzger 1973), Cubeo (Salser and Salser 1976), Retuarã (Strom forthcoming), and Siona (Wheeler 1970).

In addition to progressive (forward) assimilation described above, Kaye (1971) describes regressive assimilation in Desano. Welch (personal correspondence) also states that Tucano has at least one case of regressive assimilation: when the intrinsically nasalized suffix -rã 'plural' is added, the preceding vowel is also nasalized (Welch personal communication). Regressive nasal assimilation can never

affect noun or verb morphemes, only suffixes.

In addition to this morphological limitation on nasalization, several languages have phonological environments which block this nasal assimilation. In Tatuyo, it is blocked by oral stops (Whisler and Whisler 1976); in Barasana, it is blocked by oral stops in accented syllables (Smith and Smith 1976); and in Coreguaje, it is blocked by oral stops, /r/, or /s/.

4.3.1.4 Tone

The Tucanoan languages are tonal, most having two emic tones (high and low), although Guanano has three (high, mid, and low) (Waltz and Waltz 1979). Most languages have a pitch-stress correlation: stressed syllables always have high tone, but not all high-tone syllables are stressed. In these languages, monosyllabic words in isolation always have high tone: tonal contrasts are found only in polysyllabic words. It was not clear from the published descriptions whether tonal contrasts are different from stress contrasts. Some languages allow only one stress per word, while others allow multiple stress.

Tucano is one language which does not follow the pitch-stress correlation. In Tucano, words may have multiple stress, low-tone syllables may be stressed, and stressed

monosyllabic words may have low tone (Welch personal correspondence).

Considerable tone perturbation occurs, and in the practical orthography used for the Tucanoan languages, tone is generally marked only to disambiguate otherwise ambiguous forms.

Barnes (1984) describes Tuyuca as having pitch-stress, rather than tone. Tuyuca words have only one high-pitch (stress) per word, instead of the multiple high-pitch syllables that other Tucanoan languages have (West 1980; Waltz 1976).

4.3.2 Typological Characteristics

The Tucanoan languages are agglutinating languages, consisting of polymorphemic words in which the morpheme boundaries are generally recognizable (Comrie 1981:42). Grimes (1984) classifies Cubeo, Northern Barasano (Bará), Siona, Siriano, and Tucano as Subject-Object-Verb (SOV) languages; Macuna, and Southern Barasano (Barasana) as Object-Subject-Verb (OVS) languages; and leaves the other Tucanoan languages unclassified with respect to word order type.

Comrie (1981) outlines various word order parameters used in classifying languages as to word order type. These include word (or constituent) order within the clause (ordering of subject, object, and verb); constituent order within the noun phrase (order of genitives and modifiers with respect to the head noun); type of adpositions (prepositions or postpositions); constituent order within comparatives (the order of the standard with respect to the object of comparison); order of auxiliaries with respect to main verbs; and the predominant type of affixes (prefixes versus suffixes).

Comrie (1981:89) outlines four general combinations of these parameters, which can be reduced to two if the position of S (subject) is removed, distinguishing between VO languages and OV languages. The OV word order type tends to correlate with the following typological characteristics:

1. Object-Verb word order
2. Postpositions
3. Descriptive modifiers generally precede the head noun in a noun phrase
4. Genitive constructions have the genitive followed by the head noun
5. Affixes are almost exclusively suffixes.

Let us consider some Tucano examples in support of this classification. In example (3), the word order is SOV -- the object 'corn' precedes the verb.

(3) bãrĩ ti dʉ̃bʉ̃ ojócare pe'érã ejá-wʉ
we that day corn theme got-we

'We got corn that day.'

The Indo-European languages with which most of us are familiar have prepositions, adpositions which precede the noun phrase with which they are associated. OV languages have postpositions, as shown by example (4), in which the postposition follows the pronoun:

(4) yʉ'ʉ́ bẽ'rã
me with

'with me'

In OV languages, descriptive modifiers generally precede the head noun in a noun phrase. In Eastern Tucanoan languages, whether a descriptive modifier precedes or follows the head noun depends on the gender of the noun and whether or not it is marked with a classifier (see sections 4.3.3.1.2 and 4.3.4.2.2). In example (5), the adjective follows an animate noun, while in example (6), the adjective precedes an inanimate noun (West 1980):

(5) bãsã ayũrã
people good

'good people (animate)'

(6) butirí wi'i
white house

'white house'

In OV languages, genitive constructions have the genitive (possessor) followed by the head noun, as shown by example (7) (West 1980):

(7) co yagʉ́ diáyi
her poss. dog

'her dog'

The possessive marker yagʉ́ is used with animate nouns which are not kinship terms -- kinship terms omit it, marking possession by the juxtaposition of the possessor and the possessed nouns.

In OV languages, affixes are almost exclusively suffixes. Only three Tucanoan languages have any prefixes at all, and in these, the majority of affixes are suffixes. Carapana and Tatuyo prefix possessive pronouns to the possessed noun. Retuarã also prefixes possessives to nouns, but prefixes subject pronouns to verbs as well.

The language descriptions by Ferguson (forthcoming) for Cubeo, Jones (forthcoming) for Barasana, and Strom (forthcoming) for Retuarã, indicate that these Tucanoan languages fit closely into Comrie's characterizations of OV type languages. These are the only descriptions that I am aware of

that discuss typological considerations in Tucanoan languages -- Grimes (1984) mentions the typological classification of some of the languages, but does not amplify on the classification. Although word order itself is not rigid, Tucanoan languages tend to follow the characteristics of the Object-Verb (OV) word order type described by Comrie (1981), either as SOV or OVS languages.

4.3.3 Morphological Characteristics

4.3.3.1 Nouns

Nouns in Tucanoan languages are inflected for case and number. In addition, they are divided into classes depending on a variety of features. The primary class distinction is based on animate versus inanimate gender, with nouns of inanimate gender being further subdivided according to the specific suffix used to designate the categories of plural or form (shape).

4.3.3.1.1 Case

Tucanoan languages have a case-marking system for nouns that varies considerably from language to language. Some languages have only two different case marking suffixes, which are described as marking six different cases, while Siona has as many as nine. This variation in case marking systems is one potential area for adaptation problems, particularly in adapting from a language with fewer markers to

one with more markers. In the adaptation process between Tucano and Tuyuca, on which our efforts to date have focussed, case marking has not been a problem, because they both have the same mapping of case markers to case functions.

For several of the languages in question, the inventory of cases is difficult to determine precisely because of the limited information available. West (1980) is a practical grammar of Tucano, written for non-linguists, and does not deal specifically with the subject of case marking, so it is difficult to determine exactly how many case markers actually exist. Her description, as well as a study of Tucano data, points to at least the following marked cases: accusative (or discourse focus specifier) -re, benefactee -re locative (spatial and temporal) -pʉ, accompaniment me'rã, and instrument me'rã.

Waltz (1976b) is written as a series of language learning lessons for Guanano and does not deal specifically with the subject of case marking, making it difficult to determine the case marking system of Guanano. This description, as well as observation of Guanano data, points to at least the following marked cases: accusative -re, locative (spatial and temporal location) -pʉ, accompaniment me'ne, and instrument me'ne.

Metzger (1981) presents a practical grammar for people interested in learning Carapana. This grammar does not deal specifically with the subject of case marking, making it difficult to determine the case marking system of Carapana. A personal conversation with Metzger points to at least the following marked cases: accusative -re, benefactee -re, experiencer -ra, locative (temporal and spatial) -pʉ, continuative temporal locative ("from then on") -paʉ, specifier -na, accompaniment -mena, and instrument -mena. In addition, Carapana has a wide variety of post-positions indicating specific locations and directions. The Carapana are river people, and four of the thirteen post-positions that Metzger mentions deal specifically with river locations, such as 'down-river,' 'up-river,' 'at the mouth of the river,' 'at the source of the river.' These post-positions are suffixes that are attached to nouns, and may be optionally followed by the locative or accusative case markers.

Jones' description of Barasana presents two different case suffixes, though he posits more cases: accusative -re, experiencer -re, benefactee -re, accompaniment -rãka, instrument -rãka, and goal -rãka (forth-coming:90).

Strom presents five different Retuarã case suffixes for the following cases: term -re ~ -te (which he defines as referring to "humans or anything (animate) given a proper name" (forthcoming:75); locative (spatial or temporal location) -rã, goal -rã, instrument -pi, source -pi, path -pi, material -pi, comitative -ka (indicating "both co-participant and conjunction" (forthcoming:81)), benefactive -ro'si, purpose -ro'si, and the final state of a process -ro'si (forthcoming:75-84).

Ferguson's description of Cubeo also presents a five suffix case system: accusative -re, experiencer -re, source -re, genitive -i, accompaniment -ke, instrument -ke, and goal -ta. There are two different locative suffixes in Cubeo: one, -i, indicates general temporal and spatial location, including place and path, and another, -rã indicates exact location (forthcoming:43-44).

Wheeler (1970) describes suffixes for the following cases (although he does not refer to them as cases): goal -nã, -nĩ, or -de, "depending on the degree of focus given to the nominal element and whether the nominal refers to an animate or inanimate participant in the discourse" (1970:42); locative -de; accusative -nĩ, -de, or nothing; benefactive -de; nominative -ga, -bi or nothing; instrument -bi; source -bi; accompaniment

nakóni; and temporal -dɨ, -tɨ, and -to.

Case marking differences have not yet been significant problems for Tucanoan CADA since both Tucano and Tuyuca have minimal case markers. However, the problems inherent to a one-to-many mapping system would affect adaptation efforts between Tucano and a language such as Retuarã, which marks five cases distinctively.

4.3.3.1.2 Gender and class

Nouns in Tucanoan languages are divided into two primary groups, based on animate versus inanimate gender. Each of these groups is further subdivided, but the number of these divisions is language specific. All Tucanoan languages include heavenly bodies as animate nouns, along with people and animals. The greatest difference in the system of gender subdivisions is found in the number of classifier suffixes for inanimate gender nouns.

The inventory of classifiers ranges from a few in Tucano to over one hundred in Tuyuca (Barnes, personal interview) and Barasana (Jones forthcoming:18). This great variation in gender-marking systems is a potential area for adaptation problems, particularly in adapting from a language with fewer markers to one with more markers.[6] In the adaptation process between Tucano and Tuyuca, all problems

in gender marking have been left for the human post-editor to resolve. Since Tucano has few classifier suffixes for the inanimate gender and Tuyuca has over one hundred, it was decided that to attempt to generate possible Tuyuca forms would require more effort on the part of the human post-editor than to insert the required forms during revision. This problem is a good example of the one-to-many mapping differences discussed in section 1.1.2.1. The subject is worthy of attention because a successful solution would contribute to a more complete understanding of the comparative morphology of Eastern Tucanoan languages, and would simplify the task of the post-editor.

Animate nouns are generally marked for gender, distinguishing between masculine (usually marked with ʉ, oral or nasalized), feminine (usually marked with o, oral or nasalized), and plural (usually marked with a, oral or nasalized, or rã). These distinctions hold across the entire Tucanoan language family, as described by West (1980) for Tucano, Metzger (1981) for Carapana, Jones (forthcoming) for Barasana, Ferguson (forthcoming) for Cubeo, Gralow (1984) for Coreguaje, and Wheeler (1970) for Siona.

The following is a summary of the sorts of variations found in gender-matching systems of Tucanoan languages.

In Tucano, animate nouns are divided into two primary groups: kinship terms and terms referring to people, animals, and heavenly bodies (West 1980:245).

Metzger divides animate nouns in Carapana into three primary groups, based on the form of the plural suffix: some kinship terms, insects, fish, and heavenly bodies; kinship terms denoting relative age (older or younger than the speaker or referent), spirits, and monkeys; and terms requiring an irregular plural form (Metzger 1981:125-128).

Jones (forthcoming) divides nouns in Barasana into eight classes. Class one nouns are inanimate (discussed below). Class two nouns must be specified for gender (masculine or feminine) and number -- gender differences are marked only for singular nouns. These nouns refer to humans or household animals, such as cats and chickens. Class three nouns occur only in the plural, and refer to groups of people. Class four nouns end in what look like the gender suffixes -ʉ and -o. These nouns refer to animals, and although they refer to animals of either gender, they require masculine gender agreement on the verb; the ʉ and o endings do not indicate gender, as can be seen from wekʉ 'tapir,' and weko 'parrot.' Class five nouns refer to kinship relationships or religious positions generally held by men, and use only the masculine gender suffix -ʉ.

Class six nouns refer to female kinship relationships, and use only the feminine gender suffix -o. The class seven noun sũka 'baby' is uninflected for gender, but triggers verb agreement in terms of the sex of the baby.

Animate nouns in Cubeo refer to people, kinship terms, some animals, and heavenly bodies (Ferguson forthcoming). Even some animals which are marked with inanimate classifier suffixes trigger animate agreement marking on adjectives and verbs.

Wheeler (1970) describes three groups of animate nouns in Siona: terms referring to the speaker's social ingroup[7]; animals and the speaker's social outgroup; and supernatural beings, either visible (including heavenly bodies) or invisible (spirits). The third group can also be used derogatorily to refer to humans in the speaker's social outgroup.

In addition to the classification of animate nouns, Tucanoan languages tend to have a rich inventory of noun classifiers for inanimate nouns. These classifiers generally indicate shape or form, but some also indicate function.

Ferguson (forthcoming:18) provides some good examples of how productive the system of noun classifiers is in Cubeo:

> For example, the noun tãũ 'metal/glass' some times appears alone. But it is also found with many classifiers, as in such words as tãũ-tʉrava (metal-cls:disk shaped) 'coin or other round metal object,' tãũ-bẽ (metal-cls:string shaped) 'chain; wire,' tãũ-yo (metal-cls:slender cylinder with a point) 'nail,' tãũ-kʉ̃ (metal-cls:like a canoe) 'motorboat,' tãũ-bʉ (glass/metal-cls:curved like a barrel) 'bottle or can,' tãũ-ve (metal-cls:long and flat shaped) 'knife,' tãũ-jĩ-ve (metal-dimin-cls:long and flat shaped) 'knife,' tãũ-yako-rʉ (glass-eye-cls:three dimensional and rather small) 'glasses' (forthcoming:18).

Ferguson estimates that there are approximately 100 different classifiers in Cubeo.

Jones (forthcoming) has found 109 noun classifiers in Barasana to date. He divides these into ten major categories: shape, masses, designs, botanical, disassociated parts, geographical, abstract, associative, general, and those he describes as residue (those which do not fit into one of the previous categories). Of these ten categories, the one containing the largest number of classifiers is the shape category, with 48.

Retuarã has a small set of noun classifiers which are obligatory on numerals, but rarely occur elsewhere. They are occasionally found on some nominal modifiers (Strom forthcoming:17).

Wheeler (1970:95) describes thirty-two different classifier suffixes for Siona (including those for animate as

well as inanimate nouns). These classifications for inanimate nouns include such forms as 'cliff-shaped,' 'fat,' 'protruding,' 'cave,' 'lagoon,' 'meshed,' 'hollowed,' 'round flat,' 'transportation,' 'line,' 'above,' 'root,' 'hook,' 'edge,' 'containing within,' 'river,' 'tree,' 'shaft,' 'place,' 'time for,' 'time when,' 'time,' 'opposite, behind,' 'opposite, across from,' 'here,' 'underside,' 'incompleted state,' 'completed state.'

Inanimate nouns are further categorized as mass or abstract nouns versus count nouns, which themselves are classified according to the type of plural suffix they use. Abstract nouns are formed by adding nominalizers to verbal forms, which may be inflected.

Some abstract nouns may be animate, such as Tucano bu'e-gʉ 'student (masc.)'; the verb root is bu'e 'study', with a nominalizer -g- and gender ʉ 'masculine.'

Differences in classifier systems have been one of the areas where the most work has been left for the human post-editor for texts adapted to date. This is due to the limited number of classifiers in Tucano when compared with Tuyuca, which has over one hundred classifiers for inanimate nouns.

4.3.3.2 Verbs

Verbs are the locus of the morphological complexity of Tucanoan languages. Verbs are marked morphologically for evidential certainty (on independent verbs), tense, mood, aspect, switch-reference (on dependent verbs), and miscellaneous other features, such as negation, directionals (on motion verbs), dubitative, and benefactive. Tuyuca, for example, has forty verb suffixes, although at most four are allowed on any one verb.

4.3.3.2.1 Evidentials

Evidential suffixes are obligatory on independent Tucanoan verbs. They simultaneously indicate person, tense, indicative mood, and speaker certainty with respect to the state or action being talked about.

Tucanoan languages mark at least four tense distinctions: present, immediate past, regular (or remote) past, and future. Carapana marks the most tense distinctions, with five: present; immediate, regular, and historic past; and future (Metzger 1981).

Barnes describes evidentials as the way the speaker expresses how he or she obtained the information being conveyed: "visually [visual]; through a sense other than the visual [non-visual]; through evidence of the state or event

[apparent]; by being told about the state or event [second-hand]; or by assuming what happened [assumed]" (Barnes 1984:255).

Each evidential marker in Tucanoan languages has a distinct form incorporating tense and person. Evidentials are only used in the indicative mood. Person is marked as non-third person (first or second person, singular or plural are not differentiated), third person masculine singular, third person feminine singular, and third person plural. Tense may be past or present tense, except that the present tense second-hand evidential does not exist. Future tense in Tuyuca and Tucano is marked with a future tense morpheme followed by the present assumed evidential (Barnes 1984: 266), i.e., the speaker assumes that the state or event will exist; the other evidentials do not (and semantically cannot) exist for the future tense.

The present tense in Tucano has one evidential marker, indicating a current or habitual action (West 1980:24). Immediate past tense indicates an action that recently took place or a state that recently existed, or a present event out of sight of the speaker. Regular (or remote) past indicates an action that took place (or a state that existed) either more than approximately three days ago or at a location far away (i.e., remote in either space or time).

Regular past is marked only for third person. For the past tenses Tucano marks the visual, non-visual, second-hand [West calls it reportative] and assumed evidentials described by Barnes above, plus one which West describes as a state of first-hand emotional involvement on the part of the speaker. In addition, the assumed evidential in Tucano is used as the evidential of choice in recounting legends. Future tense indicates an action that has not yet occurred or a state that does not yet exist, but whose occurrence or existence is considered probable or definite. Like Tuyuca, the Tucano future tense is formed by adding the present tense evidential (since Tucano has only one) to a future marker. This forms what West calls the indefinite future. However, Tucano also has three other future forms: the future reportative (formed by adding the past reportative to the future tense marker), the future assumed (formed by adding the past assumed to the future tense marker), and the future definite tense. The first two occur only in the third person, while the third occurs only in the first person.

Carapana marks three evidentials in the present, one witnessed and two non-witnessed. Metzger (1981) calls the witnessed evidential the definite present, while the non-witnessed evidentials include the probable present and the intuitive [assumed] present. Carapana marks three degrees

of past (immediate, regular, and historic), instead of the two that are common to the other Tucanoan languages. Metzger (personal interview) discusses four evidentials for the past which he says "indicate diminishing degrees of reliability of information communicated by the speaker": witnessed, evident, reportative [second-hand], and probable [assumed]. The Tuyuca distinction between visual and non-visual is subsumed under the single category 'witnessed.' Future tense indicates an action that has not yet occurred or a state that does not yet exist, but whose occurrence or existence is considered probable or definite. Carapana distinguishes between two future evidentials: the regular [definite] future, which can occur with any person and number, and the probable future, which occurs only in the third person. Unlike Tuyuca and Tucano, which add present tense suffixes to a future tense marker to form the future, the Carapana regular future tense is unique. The probable future is formed by adding the reportative past evidential to the future tense marker.

Barasana verbs mark four basic categories of evidentials: "witnessed, entailment, irrealis, and reportative" (Jones forthcoming:69). These occur in various tenses, although not all categories can occur in all tenses. In Tucanoan languages, tense not only indicates relative time but also correlates with distance (spatial or temporal) as

well. Jones' description of present tense evidential markers implies that present tense in Barasana is used to express information that the speaker is personally witnessing. These evidentials mark present proximate, which describes events or states occurring in the immediate presence of the speaker, or for which spatial or social distance is irrelevant; present non-proximate, which the speaker uses to reflect a spatial or social distance from the event or state; and heard, which indicates that the event or state was not perceived visually. Three degrees of past are distinguished: immediate, used for events occurring earlier in the day; recent, used for events of the past few days; and remote, used for events occurring from a week to many years previously. Past tense evidentials can be of any of the four basic categories. Jones' entailment category corresponds to Barnes apparent evidential. Jones' irrealis category corresponds to Barnes assumed evidential, but Barasana appears to distinguish among the types of assumption more specifically than does Tuyuca. The reportative category is used to indicate that the speaker received the information being conveyed second-hand.

Retuarã is very different from the rest of the Tucanoan languages (it will be remembered that its classification is somewhat controversial). Strom (forthcoming) does not describe an evidential system for Retuarã, which is very

strange if Retuarã is in fact a Tucanoan language. The Retuarã lexicon indicates that it is clearly related to the other Tucanoan languages, but there are significant differences in both phonology (the lack of the high central vowel) and morphology (lack of evidentials) that cast doubt as to its rightful classification. Strom's discussion focuses on the tense system. Retuarã has two present tense suffixes, both of which are used for present events or past events with present significance. One of these is used with customary actions, while the other has less of a durative sense. There are three past tenses that specify degree of remoteness: immediate past, used to describe events up to one day previous to the time of speaking; intermediate past, from two days to approximately five months previous; and remote past, more than five months previous. There is only one future tense. The tenses in Retuarã are marked by tense suffixes, rather than groups of evidential suffixes.

To sum up, the differences in evidential systems are generally rather small across the Eastern Tucanoan languages. Some of the differences can be attributed to the collapse of the two witnessed categories (visual and non-visual) into one witnessed category for past tenses in Carapana (it is not clear whether there was originally one category which split to further specify witnessed events or states, or whether two categories collapsed). Tucano and

Tuyuca generally agree on the use of the tenses and evidentials, so evidentials have posed no major problems for the Tucanoan CADA efforts to date.

4.3.3.2.2 Mood

Tucanoan languages have four moods, most of which are indicated by specific verb suffixes: indicative, imperative, conditional and interrogative. The indicative mood is the unmarked mood, indicated by the evidentials discussed above.

The interrogative suffix can be used with the same tense suffixes as the indicative mood, when tense is marked separately from the evidential (West 1980; Metzger 1981). Jones does not include interrogative as a mood in his description of Barasana (forthcoming:34).

Conditional sentences generally have one or more subordinated clauses marked by a subordinating suffix, and an independent clause containing a verb marked with a conditional suffix, followed by the appropriate evidential or interrogative suffix. In Carapana, conditionals are allowed only in the present or past indicative tenses.

Tucanoan languages have a wide variety of imperative forms used for expressing commands (including direct, indirect, future, polite, and polite negative commands), as

well as for scolding, exhorting, giving permission, and warning (West 1980; Metzger 1981).

4.3.3.2.3 Aspect

Tucanoan languages mark aspect in various ways. One of these ways is through the evidential system (discussed in section 4.3.3.2.1), which indicates speaker certainty with respect to the action or state of the verb. Other kinds of aspect are also marked with suffixes, while still others are marked using compound, serial, or auxiliary verb constructions.

Tucano and Carapana use suffixes to mark the completive, emphatic, habitual, negative, desiderative, and frustrative aspects.

Jones (forthcoming) discusses the perfective, progressive, anticipatory, habitual (repetitive), durative, and contra-expectative [frustrative] aspect markers for Barasana, which are all marked with suffixes (though some require auxiliary verbs as well).

Ferguson (forthcoming) discusses the durative, completive, habitual, perfective, progressive and prospective aspects in Cubeo. Some of these aspects appear with simple verbs, while others appear as suffixes on auxiliary verbs.

Retuarã has fewer aspect markers than other Tucanoan languages, but it marks continuative, which the others don't mark, as well as the durative and completive that the others do mark (Strom forthcoming).

4.3.3.2.4 Direction

Many Tucanoan languages mark direction by adding suffixes to verbs of motion. In Tucano, these verbs include those defined 'to move down-river,' 'to move up-river,' 'to move down the trail,' 'to move up the trail,' 'to return,' 'to enter,' 'to leave the jungle,' and 'to take.' The directional suffixes are 'motion away from the speaker' and 'motion toward the speaker,' specifying the relative location of the speaker with respect to the person or object that is moving (West 1980). These directional suffixes also hold for Tuyuca. Carapana has neither motion verbs nor directional suffixes on verbs; Metzger (personal interview) suggests that this information may be handled by use of the directional postpositions that are suffixed to Carapana nouns.

4.3.3.2.5 Miscellaneous

In addition to the various categories described above, Tucano has verb suffixes which mark indefiniteness and change of focus (similar to a passive construction) (West 1980). As a general rule of thumb, Tucano verbs generally

consist of a root and two or three suffixes, although longer words do occur less frequently.

4.3.4 Syntactic Characteristics

4.3.4.1 Agreement

Tucanoan languages have gender and number agreement between subject noun phrases and independent verbs. As indicated in section 4.3.3.1.2, gender distinctions indicate animate versus inanimate, and if animate, masculine versus feminine. In addition, noun phrases containing inanimate nouns manifest quantifier, modifier and noun agreement with respect to noun classifiers.

4.3.4.2 Word order

4.3.4.2.1 Sentence

As mentioned in section 4.3.2, Tucanoan languages tend to have a preferred SOV word order, although word order is not rigid and other orders are frequently found in text. As discussed in section 4.3.2, Grimes (1984) classifies Cubeo, Bará, Siona, Siriano, and Tucano as SOV languages, while Macuna and Barasana are classified as OVS languages; the other Tucanoan languages are not specified for typological classification. Metzger (personal interview) stated that the most frequent word order for Carapana is OSV.

Although the normal word order for Tucano is SOV, West (1980) describes the construction in which the subject follows the verb in Tucano as an order used for emphasis or clarification. The nominal constructions used to determine word order are noun phrases. Pronouns do not always follow the same rules as noun phrases; for example, in Tucano, pronominal subjects are optional. Carapana, on the other hand, requires pronouns for first or second person subjects and these pronouns can never follow the verb, though nouns or third person pronouns may follow the verb to emphasize or clarify, as in Tucano (Metzger 1981:17).

The case marking system (section 4.3.3.1.1) allows the hearer or reader to determine the relationships between the various components of the sentence, so word order can serve to mark discourse relationships, such as focus or emphasis, rather than being required to indicate sentential relationships.

4.3.4.2.2 Noun phrase

Little variation in word order within the noun phrase exists in Tucanoan languages.

In possessive noun phrases (genitive constructions), the possessive pronoun or the possessor noun phrase always precedes the head (possessed) noun. Some languages which do

not otherwise have prefixes bind the possessive pronoun to the possessed noun, including Carapana and Tatuyo. Retuarã also prefixes possessives to nouns, but it has verb prefixes as well.

In languages having comparative constructions within noun phrases, the standard of comparison precedes the object of the comparison.

Quantifiers generally precede the head noun, except in Tucano, where they always follow it (West 1980:175).

Numerals precede the head noun and require an obligatory classifier suffix which agrees in class (for inanimate nouns) or gender-number (for animate nouns) with the head noun.

Adjectives generally precede the head noun, but the reverse order (with adjectives following the head noun) is fairly common. In Tucano, the order of adjectives with respect to head nouns depends in some cases on the gender or class of the head noun (West 1980:175), although they fit the general pattern of preceding the head noun, except for descriptive modifiers. Strom (forthcoming) believes that in Retuarã, the order with adjectives following indicates that the quality of the adjective is being predicated; however,

when multiple adjectives occur, one precedes the head noun and the rest follow it, although multiple adjective constructions are rare. Most Cubeo adjectives precede the head noun. Ferguson (forthcoming) observes that while descriptive adjectives follow the head noun, some of these contain a verb root 'to be or have,' indicating predication, rather than modification (this coincides with Strom's observation in Retuarã). Jones (forthcoming) states that Barasana modifiers may precede or follow the head noun, and does not indicate a preferred order.

Tucanoan languages do not have articles. Demonstratives function like adjectives: they generally precede the head noun, but may follow it in a predication. In Tucano (West 1980), demonstratives only precede the head noun.

In summary, the relative order of elements within a noun phrase is as follows: single modifiers generally precede the head noun; if multiple modifiers occur, one precedes the head noun, and the rest follow it. If there is a genitive (possessive) in the noun phrase, the genitive precedes the head noun, and any other modifiers follow it.

Further, the classifier system plays a concordial role within the noun phrase: adjectives, numerals, and demonstratives are generally marked with classifier suffixes (see

section 4.3.3.1.2) to agree in class or gender with the head noun.

4.3.4.2.3 Verb phrase

Verb phrases in Tucanoan languages consist of verbs and the adverbs or adverbial constructions that modify them, indicating temporal or spatial location, or manner. These modifiers appear either initial in the clause, before any explicitly marked subjects and objects, or directly following the verb.

4.3.4.2.4 Clause chaining

Tucanoan languages typically string dependent clauses in narrative discourse. Thus a sentence may consist of a series of dependent verbs with their corresponding complements to indicate actions in temporal succession, with the sentence terminating with an independent verb marked with an evidential (Smith 1977). Stringing dependent verbs together in this way is known as clause chaining.

The dependent verbs may generally be marked with any of the verbal suffixes described in section 4.3.3.2 except the evidentials. Subjects of these dependent verbs may or may not be the same as the subject of the main (matrix) clause containing the independent verb marked with the evidential. To indicate that the subject of the dependent verb differs

from the subject of the matrix clause, a suffix is attached to the dependent verb marking it as having a different subject. This system of marking same and different subject on dependent verbs is known as switch-reference.

Problem areas for adapting texts among the Tucanoan languages arise primarily due to differences in the order of constituents. There is little ordering difference for noun phrase constituents among the Tucanoan languages. Sentential constituents vary more widely, including even differences in word order type within the Eastern Tucanoan language family. These differences have not significantly affected the successful results obtained between Tucano and Tuyuca because of the close similarities between the two languages, but they will be more significant as other languages are added.

4.3.5 Discourse Characteristics

Tucanoan languages have morphological markers which function on the paragraph or discourse level to indicate thematic participants or events.

Karn (1976) discusses five Tuyuca morphemes which function above the sentence level: two of these function on the paragraph level and three on the discourse level. Of the two functioning on the paragraph level, one marks cohesion

within the paragraph, while the other indicates contrastive participants. Of the three functioning on the discourse level, one indicates a change in spatial or temporal setting, another indicates contra-expectation, and the third marks actors or events that form the theme line of the narrative discourse.

Linkage between paragraphs (and even sentences) in Tucanoan languages is often accomplished by repetition, typically repeating a verb (Whisler 1976; Waltz 1976a). The thematic participants may be identified by special suffixes indicating their status, and dependent verbs may be marked with switch-reference suffixes indicating whether the participant involved as the agent of the dependent verb is different from the agent of the independent verb.

Several of the discourse markers appear to function in similar ways between Tucano and Tuyuca, and thus have not posed significant problems for the adaptation experiment.

Notes:

[1]I thank Ronald Metzger and Betty Welch who read and commented on earlier drafts of this chapter and corrected some misunderstandings about Tucanoan languages and peoples.

[2]Languages to which the program has been applied.

[3]The names Northern and Southern Barasano came about through historical accident: the names themselves are not meaningful (Metzger personal conversation). This study will refer to them as Bará and Barasana, respectively, even

though publications cited refer to them as Northern and Southern Barasano.

[4]The Retuarã and Tanimuca people belong to separate ethnic groups. Strom (forthcoming) and Grimes (1984) state that these people speak the same language. Key (1979) and Grimes (1984) classify Tanimuca as a Western Tucanoan language, but Malone (personal correspondence) classifies Retuarã as Middle Tucanoan, and Grimes (1984) has a note indicating that Tanimuca-Retuama (a spelling variant) may possibly be Eastern Tucanoan. The classification problem is difficult because, as discussed in the text below, though the language shows a close lexical similarity to the other Eastern Tucanoan languages, certain syntactic and phonological features common to all the other Tucanoan languages, such as the highly developed evidential system and the high central vowel, are missing.

[5]Retuarã does not have this vowel.

[6]Metzger (personal interview) points out that some of these apparent differences may be merely differences in interpretation on the part of the linguists studying the various languages.

[7]The terms "ingroup" and "outgroup" are from Wheeler (1970).

Chapter 5

METHOD

5.1 Brief History

The Tucanoan CADA Project began in January, 1982, in Colombia, with a committee of five people: Joseph Grimes, who was visiting at the time, provided input from his general background in computational linguistics; Stephen Walter, the Linguistics Coordinator of the Colombia Branch of the Summer Institute of Linguistics, was responsible for overseeing the project; Terrell Malone provided information on some of the comparative linguistic issues which needed to be dealt with for the Tucanoan languages; Richard Aschmann and I were responsible for implementing the system that the project team decided on. Grimes, Walter, and Malone provided input during discussions of factors which appeared relevant for dialect adaptation in the Tucanoan languages, then left the rest of the design and the implementation up to Aschmann and me.

Robert Kasper (Kasper and Weber mss.) was concurrently in Peru working with David Weber redesigning the Quechua adaptation system that Weber and Mann (1980, 1981) had originally written in INTERLISP on a DEC-20. Kasper's implementation was written in the C programming language to run on a Digital Equipment Corporation LSI 11/23 computer.

By the summer of 1982, when Aschmann left the project to work with another language group, the CADA project had begun to bog down, primarily due to our inexperience with specifying and managing a large software project. In addition, there was a general lack of comparative lexical and syntactic information on the languages involved, though what was available through Malone's research was very helpful.

At about this time, Kasper finished designing and coding an initial version of a CADA system for the Quechuan languages of Peru, and was returning to the United States to continue his studies. He spent a week with me in Colombia, where we quickly determined that progress would be much more rapid for Tucanoan CADA if I began with his work for Quechua and continued from there. Since the Tucanoan CADA project was entirely my responsibility by that time, the decision to leave what we had begun and start over, based on a system which was working (though in a different language family), came easily.

From August of 1982 to the summer of 1984, I worked on modifying the programs[1] (described in section 5.2), and expanding and clarifying the morphological descriptions of Tucano and Tuyuca encoded in the dictionaries (described in section 5.2.2.1). The analysis phase was successfully completed in early 1984, with the transfer and synthesis phases

following later. By the fall of 1984, the system had successfully adapted approximately 100,000 words from Tucano to Tuyuca, in a form that the linguist for the Tuyucas considered an acceptable and very useful translation for human editing. As a further experiment, I supplied a partial dictionary of the Yurutí language and was able to get results after just a few hours' work, adapting from Tucano to Yurutí. This experiment suggested that an adaptation path using Tuyuca as the SL for Yurutí as the TL would be preferable to continuing adaptation from Tucano to Yurutí, but the results were encouraging none-the-less.

The modifications to the Quechua CADA programs required to produce useful results for Tucanoan languages include, among other things, those which allow the system to deal with null allomorphs of morphemes, next-word look-ahead, and generalized phonological environments. These will be discussed in more detail in section 5.2.2.1.

The dictionaries used by the CADA programs encode information relating to cross-linguistic morpheme equivalence, relative order with respect to other morphemes, and context information necessary for selecting the correct allomorph. Much of this information is not contained in dictionaries used by linguists working with one language, at least not in a form that is useful to a computer program.

The background work of developing dictionaries for CADA included preliminary studies in the comparative morphology of Eastern Tucanoan languages, determining relative ordering relations, co-occurrence restrictions, and developing lexical equivalence tables.

I left Colombia in November 1984, with the programs working and in the hands of the linguists there; some work has continued there since I left.

5.2 System Components of Tucanoan CADA

The current version of the CADA programs is written in the C programming language, and is running under the RT11 and TSX operating systems on Digital Equipment Corporation LSI 11/23 and 11/73 computers. The system is designed to operate in two passes, as depicted in Figure 5.1. For each pass, relevant data files are used to allow the linguist to specify the idiosyncratic restrictions or conditions that apply (these files will be considered in more detail in the specific sections to which they relate).[2]

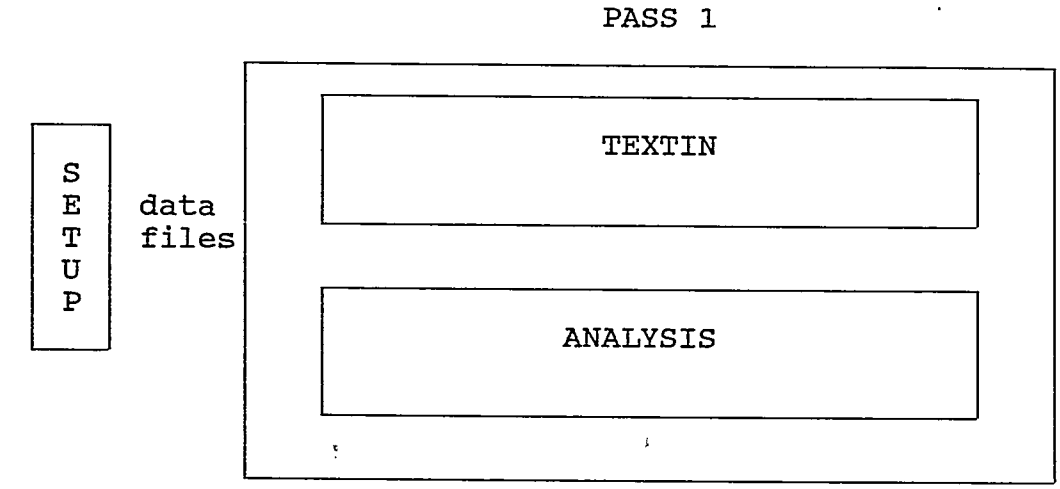
The reason for multiple passes is twofold: first, the division into an analysis phase, dealing solely with the SL text, and transfer and synthesis phases, dealing solely with the TL, allows the system to deal with one language at a time. Division into these three phases is typical of MT

systems (cf. Colmerauer et. al. 1976). This simplifies considerably the amount of interaction between the various constraints that are placed on the morphemes. Second, it is possible to optimize the results of analysis by editing the analyzed text file, a sample of which appears in Appendix A, reducing morphological ambiguities which may not collapse in the transfer or synthesis phases. Such optimization could be particularly useful if transfer and synthesis are to be done for multiple languages from the same analyzed source text. In addition, the analyzed source text is also useful in its own right, as a text glossing tool, particularly for further study or publication of linguistic data papers about relatively unknown languages.

The intermediate form between analysis and transfer-synthesis is referred to in Figure 5.1 as the 'Analyzed Source Text.' This form consists of a series of fields containing information relevant to either the analysis or the reconstruction of the TL text. These fields are:

1. the original input word from the text (optional);
2. formatting commands for typesetters or word-processors, if any appear in the source text;
3. the analyses of the word;
4. punctuation information from the source text necessary for reconstructing the target text (if any);

5. capitalization information (if any).



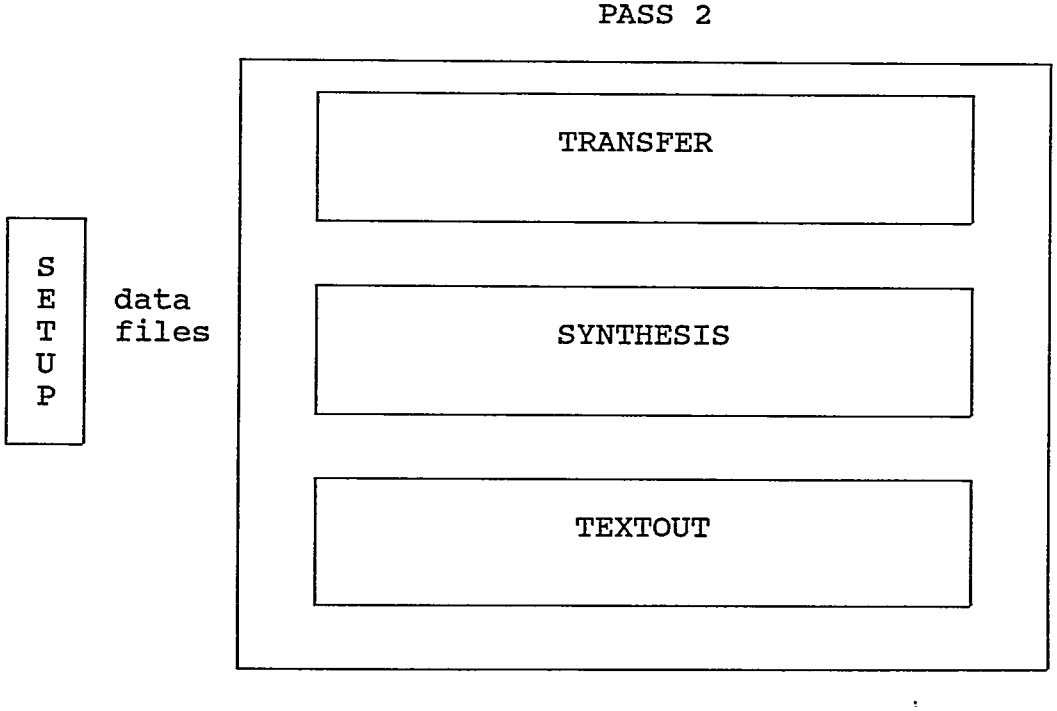


Figure 5.1 Block Diagram of CADA System

5.2.1 Textin

The textin phase functions like the lexical analyzer of a compiler (Aho, Sethi, and Ullman 1986): it reads in the SL text, strips out information that is not used by the analysis phase (such as text formatting commands used by typesetters or word processors, capitalization, and punctuation), breaking the text into words, and passes these words with their associated information to the parser, the analysis phase. The information associated with each word is not discarded, but is passed with the source text word through the various stages until the textout phase needs it to reconstruct the capitalization and punctuation for the TL text. This punctuation information is available to the analysis, transfer, and synthesis phases, but is currently only used by the textout phase. Thus, the output of the textin phase is a word to be analyzed in a form recognizable to the parser, as well as all of the other information necessary for the textout phase to reconstruct the necessary format commands, punctuation, and capitalization for the synthesized text.

Two types of control information are relevant for this phase. The first specifies which (if any) non-alphabetic characters are to be considered parts of words. For Tucanoan languages, these characters include accents, the tilde (which indicates nasalization), and hyphens (which are used

in the ʉ character). The second type of control information is contained in a file which specifies what changes should be made to the input file to regularize the orthographic data for processing. This file for Tucano looks like this:

```
INPUT orthography change for Tucano
"ʉ"     > "&" change digraph to a single character to
              simplify processing
"'"     > ""  delete tone marks (inconsistently marked in
              Tucano)
```

This regularization of data allows rules to refer to the high central vowel /ɨ/ (orthographically ʉ) as a single character, rather than the sequence of symbols -u. This is particularly relevant for preceding character context, as in the case of Tucanoan vowel harmony for this fragment of the suffix dictionary entry for the first person visual, present, evidential suffix:

```
\g vis.pres-3
\ftc a / "_
\ftc "a / a_
\ftc "e / e_
\ftc "i / i_
\ftc "o / o_
\ftc "u / u_
\ftc "ʉ / ʉ_
```

This fragment is read "The visual, present, nonthird person morpheme in Tucano has the allomorph a in the environment following a glottal stop, glottal stop (") a following an a, glottal stop e following an e, etc." This form encodes the process of vowel harmony in a cumbersome notation, but provides for easy pattern matching in comparing the surface manifestation of the morpheme with the text string. It also

provides an easy method of selecting the correct form for the morpheme during the synthesis phase -- merely look at the last character of the surface form reconstructed thus far and select as the surface form the allomorph whose environment list matches the final character. The various fields of a dictionary entry are discussed below in section 5.2.2.1; this example merely illustrates how context information is used.

5.2.2 Analysis

Analysis is the heart of the system. Analysis takes the regularized words from textin and attempts to decompose them into all possible combinations of roots and suffixes. A more complete description of the form of these roots and suffixes will be presented in section 5.2.2.1.

The essential characteristics of the Tucanoan languages have been discussed above in Chapter 4. Because most of these languages do not have prefixes, the initial implementation of the program is designed to build a list of all root entries in the dictionary which match initial substrings of each word in the input text, one word at a time. For each root in this list, the remaining substring (i.e., the rest of the word) is successively checked for initial substrings which match entries in the suffix dictionary. Each analysis for which this process successfully reaches

the end of the input word, and which passes all of the tests (discussed in section 5.2.2.2) applied to it is added to a list of possible analyses (parses) of the word. This list of possible parses is then written to the analysis field of the intermediate file, which is called the 'Analyzed Source Text' in Figure 5.1.

If the program is unable to find an entry in the root dictionary which matches an initial substring of the word, a root failure occurs. Or if the program finds a root, but is unable to find a set of one or more suffixes which allow the rest of the word to be parsed, an analysis failure occurs. These failures are written to the error log file, in the form "RF: %1word%," for root failure, or "AF: %1word%," for analysis failure. Since there was no successful analysis, a "%1word%" is written in the analysis field of the intermediate file to indicate that no parse was possible for the word, and processing continues with the next input word. Failures marked in this way are passed unchanged to the synthesis stage, which ignores them, then into the output TL text file where the human post-editor must resolve the problem.

A study of the error log file reveals several kinds of errors. The most obvious kind of errors indicated are spelling or typographical errors. By finding this kind of

error, the program provides a valuable function as a spelling checking mechanism, a function that is widely available for English and other major languages, but not generally available for minority languages.

Another kind of error indicated by the error log is a deficiency in lexical coverage. Both root and analysis failures can be due to the inability of the program to find an appropriate form in the root dictionary. In the case of a root failure, the linguist merely adds the root to the root dictionary. In the case of an analysis failure, something was found which was interpreted to be a root, but there was either no path through the word to be found in the suffix dictionary, or there were various types of restrictions placed on the occurrence of the suffixes found which were not met by the input word. The proper root form can be added, or the linguist may need to revise the required morphological description (encoded in the dictionary), in order to incorporate the information necessary to allow the correct analysis. This byproduct of the analysis phase helps to improve and clarify the knowledge of the morphology of the language.

For example, this morphological description includes information on lexical category, for instance, that a noun suffix cannot follow a verb root, or vice versa. Or it may

be stated that specific morphemes can never co-occur with certain other morphemes, so any analysis which places them together should fail. Further, certain morphemes bear a linear ordering relationship with respect to other morphemes, and any violation of these ordering constraints should cause the analysis to fail. Finally, certain morphemes can only occur word final, while others never can. These constraints are encoded in the dictionary (described in section 5.2.2.1), and are tested by a series of applicable tests selected by the linguist. When a parse which should succeed fails by one of these tests, the morphological description must be revised.

In practice, the morphological description can be developed iteratively by building a dictionary fragment containing a limited number of morphemes. Test data is analyzed, then the output is studied to determine what deficiencies must be remedied by making changes to the dictionary entries in such fields as order class, environment, category, etc. This process is applied iteratively, adding more data, until the dictionary contains a rather complete description of the morphology of the language. In this sense, a morphological description is probably never complete, since, given enough data, some form will come up which has not been faced before. The process of setting the program up for a new language involves iterating through

this process until the system produces results which begin to be acceptable to the linguist working with the TL.

The morphological parser must deal with the problems of homographemic morphemes (different morphemes that are spelled the same way) and null allomorphs (allomorphs which have no representation in the input string, but which are present in meaning). An example of homographemic morphemes in English is saw. This character string can represent three morphemes or morpheme clusters:

1. Noun -- a tool used to cut with.
2. Verb -- the act of cutting with a back and forth motion.
3. Verb -- past tense of the verb see.

A possible example of a null allomorph in English is the plural of deer or sheep. The regular plural suffix marker for English is orthographically -s, or -es, along with the irregular plural suffix -en (as is ox/oxen). An irregular form of marking plural in English is the vowel difference indicating plurality in the pairs mouse/mice, or foot/feet. In these examples, the difference between singular and plural is clearly marked. A null allomorph is one which has no surface representation: i.e., it is not possible to tell morphologically whether sheep or deer is singular or plural. The approach taken by the program is that if a context exists in which a null allomorph could occur, an analysis is

generated which includes that allomorph. The constraint tests then apply to eliminate analyses where the morpheme represented by the null allomorph would be illegal.

Two kinds of control files are used by the analysis phase. One kind is the dictionary files, and the other the specification of which tests are desired, and the preferred order of application.

5.2.2.1 Dictionaries

In the initialization phase, the program reads in at least two dictionary files, one containing suffixes and one or more containing roots. These dictionaries are kept separate because different kinds of information are required for roots than for suffixes. For example, roots in Tucanoan languages may influence nasal spreading, which is progressively assimilated through the word, but they would never themselves have conditionally nasalized allomorphs. Roots also occur word initial, so are not subject to order classes, as are suffixes.

The internal representation of the dictionary in the computer is a memory-resident trie structure, containing the relevant information from the dictionary files. A trie is a computer data structure which allows efficient use of memory for storing the character strings, and provides fast access

to the strings being looked up. The representational scheme behind a trie is as follows: for each letter which appears in initial position in an allomorph, there is a pointer to a list of all the allomorphs beginning with that letter. If any allomorphs in the list have the same second letter in common, this letter is placed on a second level, with a pointer to a list of all allomorphs which share the same first and second letters. This process of descending levels continues while there are shared initial letters in the remaining substrings. Figure 5.2 illustrates this concept for some English words. The efficiency of the trie memory representation increases considerably as the number of lexical entries increases. Rather than needing to search 1000 or more entries to find a desired allomorph, the search is limited to a list of initial characters, pointing successively to other lists of characters, until the list is found containing the desired allomorph. Associated with each allomorph is a pointer to the shared dictionary entry: some information, such as conditioning environments, order class, etc., is allomorph-specific; other information, common to all allomorphs of a morpheme, belongs in the shared dictionary entry, including morph-name, category, etc.

The form of the two dictionaries is different and the information required by the program is different for roots and suffixes. The dictionary control file specifies which

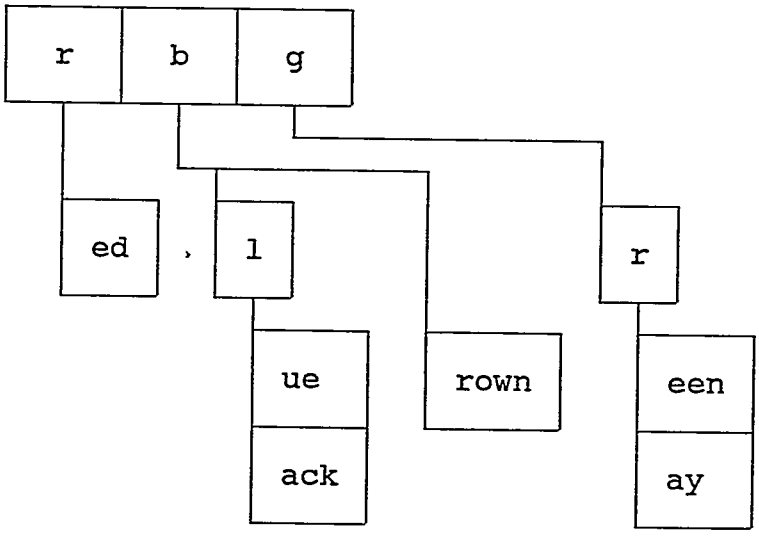


Figure 5.2 **Sample Trie for red, blue, green, gray, black, brown**

fields of a dictionary entry are to be loaded, as well as which morphemes are to be loaded. This allows a linguist to selectively load relevant parts of the dictionary, excluding, for example, long lists of plant names in texts where these would not be used.

Both the root and suffix dictionaries are organized by morphemes. These dictionaries may contain as much information as the linguist desires, but the program minimally requires the following fields, which are discussed in more depth below:

1. Morph-name
2. Category
3. Allomorphs

4. Order class (for suffixes)
5. Next word (required for morphemes requiring auxiliary verbs)

5.2.2.1.1 Morph-name

Each suffix morpheme has a morph-name, a short gloss which uniquely identifies it. The morph-name is not used for the roots. Instead, the surface form, or an etymological form, of the root is passed directly into the analyzed source text file. The reasons for this will be covered in more detail in the discussion of the transfer phase (section 5.2.3).

5.2.2.1.2 Category

Category corresponds to part of speech. The theoretical basis for the approach to lexical and state categorization is found to be an application of the notion of suffix-as-operator, developed by Weber (1976).

The notion of suffix-as-operator describes a system in which at any state in a morphological graph, it is possible to determine what kind of suffixes are required to be able to terminate the word. Quechua, as described by Weber, has nominals, adjectives, adverbs, and transitive and intransitive verbs. Weber uses the term 'place' to identify the number of category-changing suffixes required to allow the

stem to stand alone. Nominals, adjectives, and adverbs can usually stand alone, i.e., they are of place 0, requiring no further suffixes. An intransitive verb requires a suffix marking subject, which changes it from a verb of place 1 (V1), which cannot stand alone, into a verb of place 0 (V0), which can stand alone. Transitive verbs (V2) require a subject marker, which changes the category to V1, and an object marker, which changes it from a V1 to a V0, which can stand alone. Changes of category are handled neatly by this system. A nominalizer can be described as a suffix which changes the category of a verb from, say V1 to a noun (N0). Adjectivizers and adverbializers function in the same way to change the category of a stem into a stem of a different category. In Quechua, there appears to be little limit to the function of these suffix-operators. Weber gives an example of a word which starts out as a verb (V1), becomes nominalized (N0), becomes adjectivized (A0), becomes verbalized (V1), and then takes a suffix which allows it to stand alone (V0):

$\text{miku}_{V1}\text{y}_{N0}\text{niraq}_{A0}\text{ya}_{V1}\text{n}_{V0}$
eat-nom-adj-verb-obj
'it becomes like food'

The Tucanoan languages use a more restricted set of operators, and the only changes used are from V1/V0, V1/N1, N1/N0. The presence of null allomorphs indicating non-final

verbs in a clause chain indicated that it would be expedient to distinguish between verbs that could stand alone with a category of V0, verbs that could take additional suffixes after reaching the category V0, and verbs that could not take any further suffixes after reaching category V0. For these last cases, the category NOSUFF was used to indicate that no further suffixes were acceptable (suffixes having null allomorphs which could otherwise apply).

The change of category from N1/N0 allows the specification of noun stems, unmarked for gender, in the dictionary. The required gender suffix is applied to nouns of category N1 and changes them to N0, allowing them to stand alone.

For the analysis phase, the notion of input and output category is relevant. Input category is taken to be the category value that any preceding morpheme must have for the current morpheme to be applicable. For example, some suffixes, such as nominalizers, apply only to verbs in Tucanoan languages, so the input category must be a verb. The nominalizer, in the course of its work, changes the verb into a noun. This results in a change of category, i.e., the input category 'verb' is changed into an output category 'noun.'

For roots, the category is a single value which identifies whether the root is a noun, adverb, transitive verb, or

intransitive verb. This functions as the output category, since roots do not have an input category, i.e., there is no preceding morpheme to supply an input category.

For suffixes, both an input and an output category must be specified, even if they are the same. This allows for a uniform categorization system for suffixes so that those suffixes which change categories, i.e., those which function as nominalizers or verbalizers, do not require a different form of dictionary entry from those which do not.

5.2.2.1.3 Allomorphs

A morpheme is a unit of meaning or function in a language, and an allomorph (in this context) is one of possibly several character strings by which that morpheme may be identified in a surface string. Both the root and suffix dictionaries allow multiple allomorphs for each morpheme. A partial sample suffix entry illustrating the allomorphs of the English plural suffix morpheme is as follows:

```
\g PL                   Morph-name
\d plural               Definition
\c N0/N0                Category: applies to nouns
\f es / s_              Applies to sibilants
\f es / x_
\f es / ch_
\f en / _               Morphological conditioning required
                        for English cannot be specified in
                        this program in its current form.
\fen s                  Default plural marker
```

In this entry, the allomorphs are marked by "\f." As

described in section 5.2.2 above, the analysis phase takes an input word, and builds a list of all root allomorphs which match initial substrings of the word. For each root, there exists a (possibly empty) substring of suffixes. If one of the allomorphs of this morpheme matches an initial substring of the string of suffixes, the parser generates a hypothesis including this morpheme in the analysis of the word. It then applies tests to determine if the allomorph conditions, such as order, environment conditions, category restrictions, etc., are met. If all these tests succeed, the tentative analysis is considered to be an acceptable parse for the word.

For example, if the parser finds the character string es as the initial substring of a suffix string (what is left after stripping off the root allomorph), it attempts to determine whether that es can mark 'plural' in this particular case. First it checks the category of the root (or preceding suffix morpheme). Is this category N0, the input category required by this morpheme? If not, the analysis fails, since this morpheme can only apply to nouns. If the analysis has not failed, the parser next checks the phonological environment -- does the preceding allomorph end in s, x, or ch? If not, the analysis fails, since the contextual requirement is not met.

Null allomorphs are allomorphs which have no manifestation in the surface string. They are required by the Tucano language because of the process of reduction of geminate vowels. Thus a morpheme whose surface representation is normally a has no surface representation when it occurs following a morpheme ending in /a/ (a+a = a, rather than aa).

5.2.2.1.4 Order class

Order class refers to the relative order or position of a suffix with respect to all other suffixes. In this system, it is a number in the range 0-255, inclusive. This number helps to constrain ordering relations with other morphemes, and determines where a morpheme is to be inserted in the synthesis phase. For example, the directional morphemes /-ti/ and /-a/ occur immediately following the root, so they would be assigned a low relative order number (e.g. 10) such that all other ordering positions would be higher. Morphemes that must occur at the end of the word are assigned the highest possible order classes to indicate that nothing can follow them. Other morphemes are assigned order class numbers that reflect their relative position between the root and the final suffixes. This test is particularly useful in distinguishing and eliminating certain wrong analyses between homographemic forms. For example, the suffix /-ti/ 'motion toward' is a directional suffix of order class

10 -- i.e., it immediately follows the verb root. Another suffix /-ti/ 'interrogative,' has order class 250, i.e., it is always the last suffix in the word. If the parser finds a word of the form root-ti, there is no preferential basis for selecting the directional instead of the interrogative, so both analyses would result. However, if the word is of the form root-suffix-ti, the directional suffix is eliminated by the fact that it does not immediately follow the root.

Certain morphemes may appear in various orders with respect to certain other morphemes. Such morphemes with variable ordering relations are assigned an order class of zero, which indicates that the ordering test is not applicable to this morpheme. The notion of order classes of affixes is discussed in depth in Grimes (1983).

Since the Tucanoan languages used in this experiment have no prefixes, roots must be the first components of words. Because their position is fixed, they do not participate in the scheme of order classification and so are excluded from ordering tests.

5.2.2.1.5 Next word

This field is used by a limited number of morphemes to indicate that a particular word is required to follow it;

for the Tucanoan languages, this typically applies only to morphemes requiring an auxiliary verb. For these morphemes, it is necessary to know whether the next word is a possible candidate as auxiliary; in these cases, the dictionary is marked with the required verb as the content of the next-word field, and the analysis phase triggers a test of the next word to determine if one of these morphemes constitutes a possible match. In each case in Tucano, a given morpheme which requires an auxiliary occurs with only one specific auxiliary, rather than a class of auxiliary verbs. This allows reference to a specific word, rather than requiring advance knowledge of the class of the next word.

5.2.2.1.6 Sample entries

Here is a sample suffix dictionary entry which illustrates some of the features of the fields described above:

Morph-name	\g dim	Short name for program
	\d diminutive	Longer name for linguist
Category	\c N0/N0	Noun input => noun output
Tucano Allomorph	\ftc cã / a_	Conditioning environment: must follow a.
	\ftc acã	anywhere except after a
Tucano Order-class	\otc 200	occurs in middle of suffixes
Tuyuca Allomorph	\fty gã	Tuyuca language form
Tuyuca Order-class	\oty 200	
Yurutí Allomorph	\fyr ga	Yurutí language form
Yurutí Order-class	\oyr 200	

Here are two more entries which show the type of complexity permitted in the conditioning environments of allomorphs:

```
\g vis.pres-3
\d visual,pres,-3
\c V1/NOSUFF V2/NOSUFF      Apply to transitive or intransitive
                              verbs
\ftc a / "_                 Vowel harmony:  isolated case, not a
\ftc "a / a_                  general pattern in the language.
\ftc "e / e_
\ftc "i / i_
\ftc "o / o_
\ftc "u / u_
\ftc "ʉ / ʉ_
\otc 240                    Occurs word final; also marked by
                            NOSUFF category as the output
                            category.
\fty a                      Only one form for this morpheme in
                            Tuyuca.
\oty 240
\fyr a                      Only one form for this morpheme in
                            Yurutí.
\oyr 240
```

As these examples show, the dictionary form presently matches character strings in the dictionary rather than capturing the linguistic generalizations that geminate vowels reduce, as in the first example, or that vowel harmony is operative, in the second example. The tradeoff demonstrated here is that this implementation opts for ease of programming (simple string matching) versus generation of surface forms from descriptive rules (which involves a much higher computational overhead at run-time). The geminate vowel reduction situation comes about only with the a, because it is the only vowel that begins suffixes. And as indicated in the comment above, this second example is the only good example of vowel harmony in Tucano -- none of the the other languages have it either. Thus these two patterns exhibit properties which make them amenable to treatment in

terms of dictionary features.

There are several kinds of conditioning environments which are defined for Tucanoan CADA. These include a generalized preceding character environment, word final environment, and preceding nasalized morpheme environment. These environments are specified for each allomorph, i.e., they refer to the context in which that specific allomorph will be found in analysis or which selects the specific allomorph in synthesis. The following is an example of conditioning environments for allomorphs:

```
\g conflict
\d conflict focus marker
\c N0/N0
\ftc ã / [~]_          Occurs after a nasalized morpheme.
\ftc - / a_            Null allomorph; no form if after a
\ftc a
\cntc ::~              Conditionally nasalized
\otc 240
\fty ja
```

This is read "The conflict morpheme has the allomorph ã when the preceding morpheme is nasalized, no surface manifestation when the preceding morpheme ends in a, and a anywhere else." Any allomorph may have several conditioning environments simultaneously. For example, a morpheme could have an allomorph which is conditionally nasalized, occurs word-final, and must follow an allomorph of a morpheme ending in a: the dictionary entry for this allomorph could be written as

cã / [~]_ / a_ / _#

There has been no need to constrain any allomorphs this rigidly in Tucanoan CADA.

Root entries are simpler than suffix entries in the following ways:

1. the morph-name is not needed, since the actual root or etymological form is passed to the intermediate file. Phonological transfer rules apply to the root or etymological form to derive the equivalent root in the TL, if the transfer is regular; otherwise a substitution is provided in the transfer table;
2. only the output category is specified, since the category is inherent to the root; and
3. no order class is allowed, since the root is always initial.

Here is a sample root dictionary entry:

Gloss	\d say	dictionary entry name for linguist
Category	\c V1	Verb
Tucano Allomorph	\ftc ni	Tucano surface form
Tuyuca Allomorph	\fty jĩĩ	Tuyuca surface form
Yurutí Allomorph	\fyr ĩĩ	Yurutí surface form

5.2.2.2 Tests

Various kinds of tests are applied in the analysis phase to weed out spurious parses. These tests fall into two main types: successor tests and word-final tests. If

any of these tests fail, the current line of analysis is terminated.

5.2.2.2.1 Successor tests

These tests are applied to morpheme pairs as the parse is being generated. By weeding out illegal parses early on, it is possible to decrease the computation time required. At the present time, the successor tests include the following:

1. CATEGORY — Checks that the output category of the first morpheme is the same as the input category of the second morpheme.
2. PREVC — Checks to see if the second morpheme specifies a required preceding character environment, and if so, that the first morpheme ends in the required character.
3. WFINAL — Checks to see if second morpheme can occur word final.
4. ADHOC — Checks to see if this morpheme pair occurs in a list of ad hoc pairs of morphemes which the user has determined may not co-occur.
5. ORDER — Checks to see that the order class of the second morpheme is not less than the order class of the first morpheme.
6. NASAL — Checks if the second morpheme requires a

preceding nasalized morpheme. If so, it checks to see if the first morpheme does in fact contain a nasal sound.

5.2.2.2.2 Final tests

These tests are applied to the whole parse for each analysis that survives all the successor tests. Some of these are similar in function to successor tests, but are different in scope of application: whole analysis versus morpheme pair. The final tests are:

1. NEXT_WORD Checks to see if any morpheme specifies a required next word; if so, it checks the next word for a match.
2. CATEGORY Checks that the final output category of the parse is permitted word finally.
3. ORDER Checks to see that the order classes of all morphemes are in ascending order.

5.2.2.3 **Rules**

The rule mechanism allows the linguist to specify certain morphemes which may never co-occur. Rules are different from the ad hoc pairs because rules do not require that the specified morphemes be contiguous and can express generalizations involving a larger morphological context. This mechanism has not been used much for Tucanoan, but the following is an example:

"contraex1" / _... "caus"

Tuyuca has two different contra-expectation morphemes, one which occurs with causatives, and another which does not. For the purposes of simplifying the transfer phase, the previous rule is used during the analysis phase in Tucano to eliminate any analyses in which a contra-expectation1 morpheme is posited with a causative anywhere following it in the analyzed string. This rule causes the analysis containing both a contra-expectation1 morpheme and a causative morpheme to fail.

5.2.3 Transfer

The transfer phase handles the root changes that are necessary for a particular dialect pair in question, in this case, Tucano to Tuyuca. For the Tucanoan languages, these changes have been handled by direct substitution, since an adequate description of a proto-language does not exist. Given such a proto-language, it might be desirable to have the analysis phase produce the proto-form for the root and handle regular correspondences by applying phonological rules. Such an approach, in addition to indicating the linguistic motivation (showing relationships) for the transfer phase substitutions, might make more efficient use of computing resources, both in terms of computing time and memory usage. This would be different from generating analysis allomorphs from rules because in the case of transfer

rules, the forms to which the rules apply are already known and little extra effort is involved, whereas with analysis rules, many incorrect forms could be generated for each character string matched.

The transfer phase must distinguish between two kinds of roots: those which undergo phonological (or rule derived) changes and those involving lexical substitution -- forms which are either irregular (i.e., the rules would not produce the correct form) or unrelated, due to such factors as borrowing from other languages. The approach taken by the program is to attempt to substitute borrowed or irregular forms first, then apply the rules to the remaining forms. The phonological rules apply to forms marked as proto-forms (with an '*'), making it possible to replace a borrowed root in the source language with the corresponding proto-form. Such proto-forms produced by substitution in the root transfer table are indistinguishable from proto-forms produced by the analysis phase. Proto-forms undergo the phonological rules to produce the correct surface form. This approach attempts to capture the linguistic realities involved. In cases where substitution takes place, however, there is no computational advantage in taking this approach and, in fact, applying phonological rules to proto-forms substituted in the root transfer table involves more computation than direct substitution of the surface form.

The root substitution table used in the transfer phase may refer to the grammatical category of the root, allowing it to distinguish between homographemic forms of different categories, such as the Tucano lexeme masã, a root which as a noun means 'people,' but which as a verb means 'resurrect.' This particular distinction is relevant in the transfer from Tucano to Tuyuca, because Tuyuca distinguishes between them; in Tuyuca, however, the words are unrelated (basocá means 'people,' while masã means 'resurrect').

A fragment of the root substitution table follows:

```
c Tucano to Tuyuca Root Changes
'acabere'       >       "wedegɨ&"
'N0 acabiji'    >       "N0 bai"
'N0 acabijo'    >       "N0 baiyó"
'N0 acaro'      >       "N0 acatiba"
'N0 masã'       >       "N0 basocá"
```

This substitution table is read as "if the current root to be transferred is acabere, replace it with wedegɨ&; if it is the string category N0 acabiji, replace it with N0 bai; ... if it is N0 masã, replace is with N0 basocá (however, if it is the verb masã, leave it alone)." (Computationally, the program does not compare each possibility sequentially; rather it uses a faster searching method (a binary search)). This table does not incorporate changes such as

"mʉ'ʉ" > "mʉʉ"

because these may be accounted for by a general rule which deletes all glottal stops (represented orthographically in

Tucano by the apostrophe (')) from Tuyuca words. Tuyuca does not have any glottal stops (Barnes and Silzer 1976), and in many cases, the only difference between a Tucano word and the Tuyuca equivalent is the glottal stop. Deriving the Tuyuca word by a such a phonological rule captures the general linguistic process taking place and eliminates a large number of substitutions.

The transfer phase handles differences between dialect pairs; any criteria necessary to select correct allomorphs for a morpheme based on order class, phonological environment, morphological insertion, deletion, or substitution are handled by rules or other selectional devices in the synthesis phase.

5.2.4 Synthesis

The synthesis phase takes the output of the transfer phase and does the work of reconstructing the analyzed input word into the TL. Any necessary root changes have already been taken care of by the transfer phase. Synthesis carries out this process of reconstruction by using a dictionary identical to that used by the analysis phase.

The fields of the dictionary relevant for synthesis are the morph-names, which specify which morpheme is involved in the reconstruction, and the allomorph conditioning environ-

ments, which specify which allomorph is to be selected in the particular context. The possible conditioning environments include whether the preceding morpheme is oral or nasalized, whether a particular allomorph must follow a particular character, and whether a particular allomorph may occur word final or not. For example, the dictionary fragment shown below (which is also given in section 5.2.2.1.6) shows that the diminuative morpheme, with morph-name **dim**, has only one allomorph in Tuyuca, gã, but has two in Tucano. In synthesizing this morpheme into Tucano, it is necessary to determine whether the preceding morpheme ends in a or not. If so, the Tucano allomorph cã is selected; otherwise the allomorph acã is selected.

Morph-name	\g dim	Short name for program
	\d diminutive	Longer name for linguist
Category	\c N0/N0	Noun input => noun output
Tucano Allomorph	\ftc cã / a_	Conditioning environment: must follow a.
	\ftc acã	anywhere except after a
Tucano Order-class	\otc 200	occurs in middle of suffixes
Tuyuca Allomorph	\fty gã	Tuyuca language form
Tuyuca Order-class	\oty 200	
Yurutí Allomorph	\fyr ga	Yurutí language form
Yurutí Order-class	\oyr 200	

Synthesis rules are useful in reconstructing words which involve changes in the string of morph-names produced by the analysis phase. These changes include morpheme reordering, insertion, deletion, and substitution. Morpheme reordering occurs where the required order in the TL is different from the order in which the morphemes were

analyzed in the SL. Any morpheme insertion necessary is done by order class, which allows the morpheme to be inserted into its proper place in the word. It is important to note that these synthesis rules do not make any claims as to the historical changes that took place in the development of these languages, although they may reflect such changes directly; instead, they express in rule form the changes that must be made to the morphological structure of an analyzed word from one language to derive the corresponding word in another.

The synthesis phase involves everything necessary to produce the correct word according to the rules of the TL. In many cases where there is a multiple parse in the analysis phase, the resulting ambiguities collapse during synthesis. This phenomenon is due to the similarities between the dialects or languages. In those cases where multiple analyses are synthesized differently, all the possible alternative forms are passed to the output, separated by '%', so that the human post-editor can decide which is preferable in the particular context. Appendix C shows the degree of collapse of multiple parses between the analysis and synthesis phases.

A special editor program, written by Alex Waibel as part of the Quechua adaptation project, allows the human

editor to deal solely with textual problems marked by the synthesis phase. These problem forms are marked in the TL text marked by the '%' and include either words which the analysis phase was unable to analyze (typically typographical errors or forms not in the root dictionary), or forms which produced ambiguities in analysis which did not completely collapse during synthesis. Separating the cleanup phase from the revision phase allows the human editor to deal with low-level details in one quick pass, while postponing the decisions requiring more thought relating to other textual corrections to be part of the revision phase. This editor program has been used extensively for the Quechuan languages, but the linguist for Tuyuca opted to use a standard text editor program instead, combining cleanup and revision in one slower pass.

5.2.5 Textout

The textout phase inserts the formatting commands, capitalization, and punctuation marking required to reconstruct the output text. This is done from the information that was originally stripped away from each word by the textin phase. The regularization of orthographic representation is also undone, so that the orthographic conventions of the TL are implemented. The result is a text containing all the formatting information of the original, as well as the newly synthesized text, with analysis failures and

synthesis ambiguities marked by '%'.

5.3 System Output of Tucanoan CADA

Appendix B shows a line by line comparison of the Tucano input text (indicated by TC), the result of the output phase (indicated by CD), and the final cleaned up Tuyuca text (indicated by TY). A careful look at this appendix will give some indication of what the program is able to do (by comparing the TC lines to the CD lines), and what remains for the human translator to do (by comparing the CD and TY lines).

5.4 Other Languages

The basic approach to morphological analysis and synthesis presented here has been applied to three different families of languages in South America: the Quechuan languages of central Peru and Quichuan languages of Ecuador, the Campa languages of eastern Peru, and the Tucanoan languages of Colombia. What these languages have in common is a complex morphological system with many kinds of interaction (and co-occurrence) restrictions. For these languages, dealing with allomorph selection, morpheme reordering, deletion, insertion, and substitution has resulted in text material in a TL that is from 80-95% equivalent to the SL text from which it was adapted. This has been accomplished with no attempt to understand the semantic content

of the SL text directly, although the rules utilized may make use of certain kinds of information making reference to semantic classes, such as verbs of motion.

The parsing mechanism has been successfully applied to Spanish text data, but brief attempts to adapt this analyzed text to Portuguese have indicated that much more syntactic processing is required in order to produce useful results. Among the differences between these two languages that are immediately observable is the structure of possessive noun phrases: in Portuguese, the possessive noun phrase requires an article, and the possessive pronoun must agree in gender with the possessed noun, while Spanish does not allow articles in possessive noun phrases, and gender is not marked on possessive pronouns; e.g. Spanish su casa versus Portuguese a sua casa 'his house.'

Such examples indicate that a different approach is required when attempting to adapt text between languages having complex syntax and syntactic differences from an approach which is successful in dealing with morphological complexity operating on the word level.

Notes:

[1]These modifications include changes to the dictionary loading routines to deal with the information required to correctly parse Tucanoan languages. Much of this information is different from the Quechuan languages, including the

kinds of tests which are significant, environment specification (this was generalized from characters built into the program, in the Quechua implementation, to a mechanism which allows the linguist to specify which characters are relevant environments for phonological conditioning of allomorphs in the specific SL or TL.)

The textin and textout phases were modified to handle orthographic characters of the Tucanoan languages, some components of which include character sequences which singly are word-building characters, but in sequences are not. For example, a single hyphen is a component of the u̲ character; a double hyphen is a dash). Requirements of Tucanoan diacritic sequences involved modification of the functions which strip capitalization away from the SL text (in textin) and replace it in the TL (in textout).

The look-ahead requirement of certain Tucanoan suffixes (to determine whether the next word is an auxiliary verb) required modification of the textin and analysis phases.

The analysis phase was modified by incorporating the Tucanoan language-specific tests for allomorph selection based on phonological context criteria, either direct context such as a specific preceding character, or determining whether the preceding allomorph was nasalized.

The transfer phase was modified to incorporate a more efficient lexical substitution mechanism. The Quechuan approach emphasized the derivation of TL forms from etymological (proto) roots, by applying TL-specific phonological rules. This approach meant that lexical substitution was only required for a limited number of irregular roots. In the Tucanoan languages, which lacked the detailed research in comparative linguistics, such correspondence rules were unavailable, placing a much larger burden on the lexical substitution tables. This in turn meant that efficiency in this part of the program was crucial.

The synthesis phase was modified by changes and additions in tests for allomorph selection.

[2]The system allows the linguist to include information in these files, especially the dictionaries, that may be of linguistic interest, but which is ignored by this program.

Chapter 6

CONCLUSIONS

6.1 Results

Using a computer program to adapt text material between two dialects of the Tucanoan language family has produced exciting and useful results. The texts adapted to date include almost 60% of the New Testament of the Bible (approximately 100,000 words), and the results are considered beneficial and worthwhile by the linguist working with the Tuyuca people who has been responsible for post-editing the program output. There are no quantifiable data to determine the effectiveness of the program, but a careful study of Appendix B will show the program output and the post-edited output, indicating that the program is able to produce most of the correspondences uniquely, and provide alternatives, one of which is correct, in many other cases.

Another benefit that has come out of the experiment has been a better and more complete understanding of the source and target languages (especially in the morphology, which is the focus of this particular program), due to the rigorous definition required to represent this knowledge in a computer. This understanding includes ordering relationships between morphemes, and a clarification of co-occurrence restrictions holding between morphemes, etc. It has thus contributed to the comparative study of Tucanoan languages,

work which is still going on and is expected to be formalized in the near future.

Initial attempts to adapt to a third language (Yurutí) produced results indicating that, within the Eastern branch of the Tucanoan language family, the kinds of correspondences between languages that produced useful results for Tuyuca also held for Yurutí. Some of the problems encountered in adapting between Tucano and Tuyuca could be eliminated by adapting from Tuyuca to Yurutí, which are more closely related lexically and morphologically. These observations were made after only two days of experimentation with Yurutí, by comparing the output of the program with a literal translation of the same text by a man fluent in both Tucano and Yurutí.

6.2 Conclusions

Machine translation of natural language texts is desirable for many reasons, but the goal of perfect translation by machine is not yet attainable, due to lack of complete understanding of all that is involved in the translation process, and inadequate, incomplete descriptions of any human language.

In spite of these limitations, computers can assist human translators in many ways, including adapting text

material between closely related languages, as described in this study. The type of language being considered for adaptation is a very important factor. To date, our attempts have focussed on languages having complex morphology, but relatively simple syntax. The approach taken has proven effective for this type of language, but a different approach would be required for languages such as those of Southeast Asia, parts of Africa, and the Pacific, having monomorphemic (or at most bimorphemic) words, and a complex syntax. The degree of success of the type of program used in this study is proportional to the closeness of the relationship between dialects, but, among the Tucanoan languages considered, and in the Quechuan languages of Peru, where a similar experiment is in almost daily use, the results have been encouraging.

Success in scientific research frequently depends as much on the formulation of the correct questions as the discovery of the correct answers. The questions asked of a problem depend largely on the framework or paradigm within which the researcher has been trained or in which the problem is cast (Kuhn 1970).

One of the conclusions reached in this study is that the capability for producing output of the kind produced by CADA was possible in the early 1960s when the ALPAC report

concluded that automatic translation by machine was not and could not be foreseen to be cost-effective. No application for adapting text between closely related languages was considered important at that time, or this type of work might have been considered worthwhile much earlier.

This research has made no significant contribution to translation theory, but has demonstrated that a system based on substituting equivalent linguistic forms can have an acceptable, though limited, degree of success, if the source and target languages of the translation are sufficiently closely related. This indicates that in spite of the fact that translation is largely a transfer or reconstruction of the meaning of a SL text into a different TL, the linguistic form of the TL can correspond closely to the form of the SL if the languages are closely related, though mutually unintelligible. The success rate of a system substituting equivalent linguistic forms can be improved by adding contextual selectional restrictions incorporating larger and larger contexts. The success of the system depends largely on the accuracy of the equivalence tables. Since translation is meaning-based, a system such as the one described here has limits which cannot be overcome without the incorporation of semantic information. Therefore, the quality of translation will eventually only be further improved by adopting a completely different approach.

Machine-assisted translation systems are indispensable for dealing with the increasing volume of translation required in today's world. Their usefulness will continue to improve as the automatic MT component improves, to the point where eventually, the human translator may be able to accept almost all of the translation produced by the machine. A system of the type described by Melby (1985), where the translator uses the computer as a word-processor with dictionary lookup and automatic translation suggestion components, will allow translators to better cope with the volume of work and to produce higher quality work in less time. While not working as interactively as the system that Melby describes, the CADA system provides a step in that direction for translators working with minority languages.

The theory of machine parsing of natural languages has taken various directions. Some parsers, including the morphological parser used by CADA, attempt to parse syntactic structures in parallel, producing all possible parses which are grammatical. This approach could easily overwhelm the synthesizer and post-editor with alternatives, if it were not the case that many analytical ambiguities collapse during synthesis due to the similarities between languages. Other parsers, including most augmented transition network grammars (ATNs), produce only one analysis of a word or sentence, beginning with the most likely analysis first.

This results in loss of information which may be important if the structural ambiguity is intentional, and may result in the selection of the wrong form where the less likely of two ambiguous alternatives is the form desired in a particular context. In the design of CADA, it was decided that the human post-editor should make the decisions in cases of ambiguity.

6.3 Further Research

Areas for further research include developing a generalized syntactic parser incorporating the morphological parser used here as a sub-component. This approach could expand the number of languages between which adaptation would be productive. The incorporation of semantic information, in order to do semantically based translation, while desirable, quickly becomes unmanageable, particularly among the minority languages with which the Summer Institute of Linguistics works. These languages differ from the major world languages in that few have more than two or three linguists studying them. By contrast, major world languages, such as English, Russian, German, and French, have been studied for years by large numbers of grammarians, philologists, and linguists, and still no comprehensive, computable grammar or dictionary yet exists for any of them. Problems of compiling, representing, and cross-referencing lexical information are difficult. The real

difficulties, however, lie in determining the nature and content of the information that humans bring to bear on the process of understanding and interpreting what they read or hear, and formalizing that information.

In the end, successful machine understanding and meaning-based translation systems will depend on the ability to represent and apply the knowledge that human beings use in understanding texts that they read or hear. This process of understanding is knowledge-based, and cannot be achieved until such knowledge is available. However, this knowledge can be discovered incrementally, and the success of some systems in understanding small problem domains gives hope for extending this knowledge to increasingly larger domains.

APPENDIX A

SAMPLE ANALYSIS FILE

This is a sample of the output of the analysis phase -- the text referred to as the "analyzed source text" or intermediate form. The meanings of the fields are:

w: the original input form of the word (unused by the program, but available to the linguist for reference) [optional if desired]

f: any preceding format markers or typesetting commands [if needed]

a: the analysis field, containing the category, root and suffixes as analyzed. A "%1" indicates a word for which analysis failed, either a root failure (the root was not in the dictionary), or an analysis failure (a root was found, but no path was possible through the end of the word). A "%" followed by a number greater than 1 indicates the number of alternate analyses which succeeded for the word. The list of analyses follows the number, each separated from the others by "%."

n: non-alphabetic characters following the word. These include punctuation, spaces and "\n," -- newline (carriage return). This field is omitted when the only following non-alphabetic character is a space.

c: capitalization. This field is omitted when the word is in lowercase, 1 if the word is initially capitalized, and 2 if the word is all capitalized.

```
f \\cid tcac1s.ana: from tcac1s.e1a on 05/APR/85
a header
n \n

w jesucristo
f \\b
a N0 jesucristo
c 1

w besecũ"cãrãrẽ
a V1 bese hab+p Plnom+p obj

w espíritu
a N0 espiritu
c 1

w santu
a N0 santu
c 1

w wetamo"que
a V1 wetamo massnom

w ni"i
a V1 ni vis.pres-3
n \n

w hechos
f \\st
a N0 hechos
c 1

w de
a N0 de

w los
a N0 los

w apóstoles
a N0 apostoles
n \n

w espíritu
f \\c 1\n\\s
a N0 espiritu
c 1

w santu
a N0 santu
c 1
```

```
w a"tiatjere
a V1 a"ti FUTiPl obj

w jesu
a N0 jesu
n \n
c 1

w cɯ̃
f \\s2
a N0 c&

w wereyu"que
a V1 were beforehand massnom

w ni"i
a V1 ni vis.pres-3
n \n

w teófilo
f \\p\n\\v 1\n
a N0 teofilo
n ,
c 1

w mɯ"ɯrɛ̃
a N0 m&"& obj

w ni"cã
a N0 ni"cã

w pũrĩ
a %3N0 pũrĩ%N0 pũ Pl%V1 pũrĩ -final%

w to
a N0 to

w dɯporopɯre
a %2N0 d&poro loc obj%V1 d&po place-p loc obj%

w ojawɯ
a V1 oja vis+p+r-3
n .

w ti
a %2N0 ti%V1 ti -final%
c 1

w pũrĩpɯre
a %2N0 pũrĩ loc obj%N0 pũ Pl loc obj%
n \n
```

```
w nipe"tise
a V1 ni all2 nomSg

w jesu
a N0 jesu
c 1

w masãrẽ
a %2N0 masã obj%V1 masã mass-pPl%

w bu"e"quere
a V1 bu"e massnom obj
n ,

w cʉ̃
a N0 c&

w tutuaro
a V1 tutua place-p

w me"rã
a N0 me"rã

w weeĩ"o"quere
a V1 weeĩ"o massnom obj
n \n

w werewʉ
a V1 were vis+p+r-3
n .

w ne
a N0 ne
c 1

w waro
a N0 waro

w cʉ̃
a N0 c&

w bu"enʉ"cã"quere
a V1 bu"e begin massnom obj
n ,\n

w cʉ̃
f \\v 2\n
a N0 c&
```

```
w bu"e
a V1 bu"e -final

w yapada"reo"quere
a V2 yapada"reo massnom obj
n ,

w cʉ̃
a N0 c&

w ʉ"mʉsepʉ
a N0 &"m&se loc

w mʉjã"que
a V1 m&jã massnom

w quẽ"rãrẽ
a N0 quẽ"rã obj
n \n

w werewʉ
a V1 were vis+p+r-3
n .

w jesu
a N0 jesu
c 1

w ʉ"mʉsepʉ
a N0 &"m&se loc

w mʉjãse
a %2V1 m&jã nomSg%V1 m&jã before%

w dʉporo
a %2N0 d&poro%V1 d&po place-p%

w cʉ̃
a N0 c&

w bu"erã
a %2V1 bu"e ifwhenPl%V1 bu"e animnomPl%
n \n

w besecũ"cãrãrẽ
a V1 bese hab+p Plnom+p obj

w dutiro
a V1 duti place-p
```

```
w cũcʉ
a V2 cũ resultM

w niwĩ
a V1 ni vis+p+r3M

w na
a N0 na

w weeatjere
a V1 wee FUTiPl obj
n .\n

w espíritu
a N0 espiritu
c 1

w santu
a N0 santu
c 1

w masĩse
a V1 masĩ nomSg

w o"oró
a %2N0 o"oro%V1 o"o place-p%

w me"rã
a N0 me"rã

w tere
a %2N0 te obj%V1 te mass-pPl%

w duticʉ
a V1 duti resultM

w niwĩ
a V1 ni vis+p+r3M
n .\n

w cʉ̃
f \\v 3\n
a N0 cʉ̃
c 1

w wẽrĩca
a %2V1 wẽrĩ after%V1 wẽrĩ rem.past.Sg%

w be"ro
a N0 be"ro
```

```
w peje
a N0 peje

w tiri
a %6N0 ti Pl%V1 ti vis+p+rint%V1 ti genSg
        %V1 ti genSg -final%V1 ti int%V1 ti cautionimp%

w cuarenta
a N0 cuarenta

w nɯmɯrĩ
a N0 n&m& Pl

w jesu
a N0 jesu
c 1

w bajuanu"cũcã"cɯ
a %4V1 bajua state emph nom+pM%V1 bajua state emph resultM
        %V1 baju motionaway state emph nom+pM
        %V1 baju motionaway state emph resultM%
n \n

w niwĩ
a V1 ni vis+p+r3M

w narẽ
a N0 na obj
n .

w tojo
a N0 tojo
c 1

w weerã
a %2V1 wee ifwhenPl%V1 wee animnomPl%

w cɯ̃rẽ
a N0 c& obj

w ĩ"arã
a %2V1 ĩ"a ifwhenPl%V1 ĩ"a animnomPl%

w marĩ
f --
a %3N0 marĩ%N0 ma Pl%V1 marĩ -final%
c 1

w wẽrĩa
a %6V1 wẽrĩa completive%V1 wẽrĩa -final
        %V1 wẽrĩ completive%V1 wẽrĩ imp
        %V1 wẽrĩ motionaway completive
```

```
        %V1 wẽrĩ motionaway -final%

w wa"ami
a %2V1 wa"a vis+p-r3M%V1 wa"a 3M%

w ní"cʉ
a V1 ni nom+pM
n \n

w catimi
a V1 cati 3M
n ,

w nicãrã
a V1 ni pastPl

w niwã
a V1 ni vis+p+r3Pl
n .

w narẽ
a N0 na obj
c 1

w yʉ"ʉ
f --
a N0 y&"&
c 1

w pacʉ
a %2N0 pac&%V1 pa resultM%

w nipe"tirã
a %2V1 ni all2 ifwhenPl%V1 ni all2 animnomPl%

w masã
a %2N0 masã%V1 masã -final%

w wiogʉ
a %3N0 wiog&%V1 wio animMnom%V1 wio ifwhenM%

w nimi
a V1 ni 3M
n ,\n

w nise
a V1 ni nomSg

w quetire
a N0 queti obj
```

```
w werecɑ
a V1 were resultM

w niwĩ
a V1 ni vis+p+r3M
n .\n
```

APPENDIX B

COMPARISON OF INPUT, SYNTHESIZED, AND EDITED SYNTHESIZED FILES

LINE BY LINE COMPARISON: Input, Output, and Post-edited Output files. The input file in Tucano is marked with TC, the output produced by the CADA program for Tuyuca is marked CD, and the post-edited final form for Tuyuca is marked TY. The effectiveness of the program can be seen by comparing the differences between the TC and CD lines, and the similarities between the CD and TY lines. The amount of post-editing required can be seen by comparing the CD and TY lines.

TC: \id TCAC1.E1A 5/JUN/84
CD: \id TYAC1S.E1A 05/APR/85

TC: \b Jesucristo besecũ'cãrãrẽ Espíritu Santu wetamo'que
CD: \b Jesucristo beserucurirare Espiritu Santo tiapurigue
TY: \b Jesucristo beserucurirare Espiritu Santo tiapurigue

TC: ni'i
CD: niia
TY: niia

TC: \st Hechos de los apóstoles
CD: \st Hechos de los apostoles
TY: \st Hechos de los apostoles

\c 1
TC: \s Espíritu Santu a'tiatjere Jesu
CD: \s Espiritu Santo atiadarere Jesus
TY: \s Espiritu Santo atiadarere Jesus

TC: \s2 cũ wereyu'que ni'i
CD: \s2 cuũ wedeyurigue niia
TY: \s2 cuũ wedesuguerigue niia
\p
\v 1
TC: Teófilo, mu'urẽ ni'cã pũrĩ to
CD: Teofilo, muure sica %3=pũri%pũuri%puni% too
TY: Teófilo, muure sicatuti too

TC: duporopure ojawu.
CD: %2sugueropure%duporopure% jóawu.
TY: sugueropure jóawu.

TC: Ti pũrĩpɯre nipe'tise Jesu
CD: Ti %2=pũripɯre%pũuripɯre% niipetire Jesus
TY: Títutipɯre niipetire Jesus

TC: masãrẽ bu'e'quere, cɯ̃ tutuaro me'rã
CD: %2basocáre%masãre% bueriguere, cɯɯ̃ tutuaro mena
TY: basocáre bueriguere, cɯɯ̃ tutuaromena

TC: weeĩ'o'quere werewɯ. Ne waro cɯ̃ bu'enɯ'cã'quere,
CD: tiañoriguere wedewɯ. Ne peti cɯɯ̃ buenɯcãriguere,
TY: tiañoriguere wedewɯ. Sicato cɯɯ̃ buesɯgueriguere,

\v 2
TC: cɯ̃ bu'e yapada'reo'quere, cɯ̃ ɯ'mɯsepɯ mɯjã'que quẽ'rãrẽ
CD: cɯɯ̃ bue yapacɯtiriguere, cɯɯ̃ ɯmɯcaseropɯ mɯarigue =cãre
TY: cɯɯ̃ bueyapacɯtiriguere, cɯɯ̃ ɯmɯáropɯ mɯarigue =cãre

TC: werewɯ. Jesu ɯ'mɯsepɯ mɯjãse dɯporo
CD: wedewɯ. Jesus ɯmɯcaseropɯ %2mɯare%mɯa% %2sɯguero%dɯporo%
TY: wedewɯ. Jesus ɯmɯáropɯ mɯawaadari sɯguero

TC: cɯ̃ bu'erã besecũ'cãrãrẽ dutiro cũcɯ niwĩ na
CD: cɯɯ̃ buera beserucurirare dutiro cũurigɯ niiwi cɯɯ̃a
TY: cɯɯ̃ buerá besecũrirare dutiro cũurigɯ niiwi cɯɯ̃a

TC: weeatjere. Espíritu Santu masĩse o'oró me'rã
CD: tiiadarere. Espiritu Santo masĩre %2coori%ticoro% mena
TY: tiiádariguere. Espiritu Santo masĩré ticocoriguemena

TC: tere duticɯ niwĩ.
CD: teere dutirigɯ niiwi.
TY: teeré dutirigɯ niiwi.

\v 3
TC: Cɯ̃ wẽrĩca be'ro peje tiri cuarenta nɯmɯrĩ Jesu
CD: Cɯɯ̃ %2diaca%diagã% síiro pee tiri cuarenta búrecori Jesus
TY: Cɯɯ̃ diari siro peecõro cuarenta búrecori Jesus

TC: bajuanu'cũcã'cɯ niwĩ narẽ. Tojo
CD: %2bauwárucujãrigɯ%bauwarucujãrigɯ% niiwi cɯɯ̃are. Teero
TY: bauwárucujãrigɯ niiwi cɯɯ̃are. Teero

TC: weerã cɯ̃rẽ ĩ'arã --Marĩ
CD: tiira cɯɯ̃re iñara --%2=jari%Mani%
TY: tiirá cɯɯ̃re iñara --Mani

TC: wẽrĩa wa'ami
CD: %5dia%diajoã%diaya%diawajoã%diawa% %2wáaawĩ%wáai%
TY: diajoãwi

TC: ní'cʉ catimi, nicãrã niwã. Narẽ --Yʉ'ʉ
CD: niirigʉ catii, niirira niiwa. Cʉ̃are --Yʉʉ
TY: jiĩrigʉ catii, jiĩrira niiwa. Cʉ̃are --Yʉʉ

TC: pacʉ nipe'tirã masã wiogʉ
CD: %2pacʉ%páarigʉ% niipetira %2basocá%masã% %2opʉ̃%quiogʉ%
TY: pacʉ niipetira basocá opʉ̃

TC: nimi, nise quetire werecʉ niwĩ.
CD: niii, niire quetire wederigʉ niiwi.
TY: niii, jiĩre quetire wederigʉ niiwi.

APPENDIX C

LOG FILES

The following files are produced to show what the analysis and synthesis processes are doing and how hard they have to work.

ANALYSIS Log File:

Tucanoan CADA, 1.01 November '84

PASS 1: ANALYSIS FROM TEXT, Fri Apr 05 15:41:08 1985

Any root or analysis failures are printed here.

Input file: tcac1s.e1a
Output file: tcac1s.ana

ANALYSIS STATISTICS:

WORDS processed: 100
Processing time: 00:02:21
Ambiguity levels

```
  0 words with  0 analyses.
 78 words with  1 analyses.
 16 words with  2 analyses.
  3 words with  3 analyses.
  1 words with  4 analyses.
  0 words with  5 analyses.
  2 words with  6 analyses.
  0 words with  7 analyses.
  0 words with  8 analyses.
  0 words with  9 analyses.
  0 words with 10 analyses.
  0 words with 11 analyses.
  0 words with 12 analyses.
  0 words with 13 analyses.
  0 words with 14 analyses.
  0 words with 15 analyses.
```

Successor Tests:

```
CATEGORY_SP called 2057 times, failed 1682.
   PREVC_SP called 375 times, failed 92.
  WFINAL_SP called 283 times, failed 6.
   ADHOC_SP called 277 times, failed 6.
   ORDER_SP called 271 times, failed 0.
   NASAL_SP called 271 times, failed 0.
```

```
Final Tests:
NEXT_WORD_FP called 169 times, failed 23.
 CATEGORY_FP called 146 times, failed 0.

Rule Tests: called 146 times, failed 0 times.


SYNTHESIS Log File:

Tucanoan CADA, 1.01 November '84

PASS 2:  TRANSFER-SYNTHESIS, Fri Apr 05 15:48:06 1985

Input file:  tcac1s.ana
Output file: tyac1s.e1a

SYNTHESIS STATISTICS:

WORDS processed:  100
Processing time:  00:00:11
Ambiguity levels

        0 words with  0 ambiguities.
       85 words with  1 ambiguities.
       13 words with  2 ambiguities.
        1 words with  3 ambiguities.
        0 words with  4 ambiguities.
        1 words with  5 ambiguities.
        0 words with  6 ambiguities.
        0 words with  7 ambiguities.
        0 words with  8 ambiguities.
        0 words with  9 ambiguities.
        0 words with 10 ambiguities.
        0 words with 11 ambiguities.
        0 words with 12 ambiguities.
        0 words with 13 ambiguities.
        0 words with 14 ambiguities.
        0 words with 15 ambiguities.
```

APPENDIX D

PARTIAL SOURCE LISTING

This appendix contains the definition files for system variables and structures, as well as the sources for the major component modules of the Tucanoan CADA project. Small functions and functions normally found in system function libraries are omitted. The compiler and programming language used in this project was DECUS C for the RT11 operating system.

As indicated in Chapter 4, the Tucanoan CADA programs are based on earlier work done by Robert Kasper and David Weber for the Quechuan languages of central Peru. Almost all modules required modification for Tucanoan, some extensive. The work presented here was developed under the auspices of the Summer Institute of Linguistics in Peru and Colombia. Although the code here presented is the source code for a currently working system, it is not complete (as indicated above) and no representation is made as to usefulness in other applications.

```
/* DEFS.H  INPUT/OUTPUT functions 26-APR-82 Robert T Kasper
 ************************************************************
 *                                                          *
 *  Modified for Tucanoan by Robert B Reed                  *
 *    1-Oct-82 -- Template changed to allow for look ahead  *
 *   14-Oct-82 -- ALDEFAULT changed for Tucanoan languages  *
 *   19-Nov-83 -- hyphen added to format marker for quote   *
 *                  dash                                    *
 *    7-Jan-83 -- BUFSIZE reduced from 100 to 90            *
 *   17-Feb-83 -- NEG changed from '~' to '!' to allow      *
 *                  nasal vowels                            *
 *                                                          *
 ************************************************************/

#define VERSION "Tucanoan CADA, 1.0 July '84"
#define CADA
#define ANALONLY
#define TRUE    1
#define FALSE   2

/* define only for text input (controlled by ANRT) */
/* only one of these may be defined */
/* #define DTB_INPUT */
#define TXT_INPUT

/* for ufopen */
#define READ "r"
#define WRITE "w"

/* max number of open files */
#define NFILES 3

/* changed from Quechua to allow nasal vowels */
#define NEG '!'

#define DELIM '\"'
#define FORMINIT "@\\-"
#define ALDEFAULT "&\"~-\'"
#define BREAKC " \t\n"

#define BUFSIZE 90
#define ORTHOTAB 250

#define NOCAP 0
#define INITCAP 1
#define ALLCAP 2

/*************************** template ***********************
 * word-format template for input and output
 */

struct template {
```

```
  char *format;
  char *word;
  char *non_alpha;
  int capital;
  char *new_word;
  char buffer[BUFSIZE];
};

/* record definitions */

#define EOR '\030'  /* ctrl-X */
#define RECSIZE 200
#define FIELDTERM '\n'

/* allocation alignment parameters */
#define BYTES 0
#define INTS  1

/************************** strlist ***********************
 * linked list of strings  used in analysis and synthesis
 */

struct strlist {
  char *string;
  struct strlist *slink;
};

/* high level of ambiguity for stats */
#define MAXAMBIG 16
/* maximum length of morpheme */
#define MAXMORPH 15

/* time buffer */

struct TIME {
  int year;
  int month;
  int day;
  int hour;
  int minute;
  int second;
  int tick;
  int tsec;
};
```

```
/* DICT.H  defs for dictionary structures  Robert T  Kasper
 ************************************************************
 *                                                          *
 *   Modified for Tucanoan Languages by Robert B Reed       *
 *   28-Sep-82  Lowering and shortening references deleted  *
 *    1-Oct-82  Previous character categories expanded      *
 *   11-Oct-82  Nextword added to allomorph                 *
 *   14-Oct-82  ONEV is changed to a non-final category     *
 *   19-Nov-82  Prev char category added to Analysis        *
 *                Code Table (ACODETAB)                     *
 *   14-Jan-83  Join added to the Analysis Code Table       *
 *                (ACODETAB)                                *
 *   14-Jan-83  ONSHEV is changed to EV                     *
 *   14-Jan-83  Rearranged to group structures with related *
 *                definitions                               *
 *   25-Feb-83  Transfer Code Table (TRCODETAB) added       *
 *    8-Jun-83  Data Base Code Table (DTBCODETAB) added     *
 *    1-Sep-83  SD_suffix modified to allow prefixes,       *
 *                indicated by setting the sign bit.        *
 *                Prefixes are indicated by a negative      *
 *                order class, in which the order will      *
 *                descend to the root, then increase for    *
 *                suffixes.  A separate trie will be        *
 *                maintained for prefixes.                  *
 *                                                          *
 ************************************************************/

/* field code table for dictionary databases */
#define CODETAB    200
#define ACODETAB   "ah\0A\0pc\0C\0fp\0F\0j\0J\0ru\0R\0sp\0S"
#define ANUMCODES  6
/* Codes in the Analysis Data Record:
 *   adhoc pairs    -- A
 *   previous char  -- C
 *   final pred     -- F
 *   rules          -- R
 *   join           -- J     join current and next words
 *   successor pred -- S
 */

#define TRCODETAB       "cl\0C\0in\0I\0pc\0P\0su\0S"
#define TRNUMCODES      4
/* Codes in the Transfer Data Record:
 *   class             -- C
 *   insertion rule    -- I
 *   previous char     -- P
 *   substitution rule -- S
 */

#define DTBCODETAB      "w\0W\0f\0F\0a\0A\0n\0N\0c\0C"
#define DTBNUMCODES     5
```

```
/* Codes in the Word Data Base Record:
 *   word                -- w
 *   format markers      -- f
 *   analysis            -- a
 *   nonalphabetics      -- n
 *   capitalization      -- c
 */

#define NCATS   8

/********************** SD Suffix Structure ****************
 */

struct SD_suffix {
  char *morphname;
  char category[NCATS];
  unsigned props;
    /* orderclass        -- low 8 bits + sign bit
     * flags             -- high 8 bits
     *  CONDNASAL 256    flag for conditional nasalization
     *   not presently used {
     *     512
     *    1024
     *    2048
     *    4096
     *    8192
     *   16384
     *   }
     *   PREFIX 32768  indicates that the affix is a prefix
     */
};

/* Properties masks for SD_suffix:
 *   USAGE: in SETUP: allo->dp.morpheme->props |= mask-value
 *          in test:  allo->dp.morpheme->props & mask-value
 */

#define ORDERCLASS  255+32768  /* mask for low 8 bits +
                                        sign bit */
#define CONDNASAL   256         /* conditional nasalization
                                   flag */
#define PREFIX      32768      /* flag for prefix */

/*************************** allomorph *********************
 */

struct allomorph {
  char *rest_key;
  /* conds[1] = (low 4 bits)  = TOCLASS1 for root entries
   *            (high 4 bits) = SFX for suffix entries
   *                            TOCLASS2 or NULL for root
                                entries.
```

```
    * conds[0] = allomorph specific conditions:
    *   PREVCHAR   1-15  -- Environment list of previous
                            characters
    *   WFINAL       16  -- May occur word final
    *   NWFINAL      32  -- May not occur word final
    *   NASALFORM    64  -- Allomorph is nasal (conditions
                            nasalization)
    *   LOAN        128  -- ROOTS ONLY: loan word.
    *   NASALREQ    128  -- AFFIXES ONLY: requires a
                            preceding nasal allomorph
    *                       (for conditional nasalization)
    */
   char conds[1];
   union {
     /* for ROOTS: proto form, NULL if morphname = string */
     char *morphname;

     /* for SUFFIXES */
     struct SD_suffix *morpheme;
   } dp;

   /* next word, if required */
   char *nextword;

   /* for linked list of allos at this node of the TRIE */
   struct allomorph *alink;
};

/* Conditions for all allomorphs:
 *   USAGE is allo->ACONDS &  NEXTWORD ?   [in test]
 *     after  allo->ACONDS |= NEXTWORD     [in SETUP]
 */

#define PREVCHAR  15    /* Mask subscripts in env array */
#define WFINAL    16
#define NWFINAL   32
#define NASALFORM 64    /* allomorph is nasal */
#define LOAN     128    /* root is a loan word */
#define NASALREQ 128    /* requires a preceding nasal */

#define ACONDS  conds[0]
#define RCATEG  conds[1]
#define ACLASS  conds[1]

/*************************** TRIE ***************************
 */

struct TRIE {
  char *letters;          /* string of letters at this node */
  struct TRIE *subtries;  /* linked list of subtries */
  struct TRIE *tlink;     /* link in list of subtries */
  struct allomorph *amorphs; /* linked list of  allomorphs */
```

```
};

/*************************** amlist ************************
 * amlist produced by copy_entries
 */

struct amlist {
  struct allomorph *amp;
  int alen;                    /* length of allomorph string */
  struct amlist *amlink;
};

/* category masks: USAGE is (CLASS x) FROM == value ?
 *                         after    x = SETFROM value
 */

#define CLASS   0377&

/* high 4 bits of byte */
#define SETFROM 0377&16*
#define FROM    /16      /* for suffixes */
#define TO2     /16      /* for roots */
#define ATYPE   /16

/* low 4 bits of byte */
#define SETTO   0377&1*
#define TO      &017     /* for suffixes */
#define TO1     &017     /* for roots */

/* in (CLASS ap->conds[1]) ATYPE field of allomorph entry */
/*    indicates whether entry is a suffix or root */
#define SFX     15
#define ROOTS   0

/* allowable categories for roots */
#define V2      1
#define N2      2
#define R2      3
#define V1      4
#define N1      5
#define R1      6
#define ONEV    7

/* may be word final categories */
#define FINALCATS 8

#define V0      8
#define N0      9
#define R0      10
#define NOSUFF  11
#define EV      12       /* used for evidentials in Tucanoan */
#define SOUND   13       /* not used in Tucanoan */
```

```
#define P0      14      /* prefix */

/***************************** cat_tab *********************
 * category table
 */

struct cat_tab {
  char *catname;
  int catnum;
};

/***************************** TD_allo *********************
 * for synthesis
 */

struct TD_allo {
  char *astring;
  unsigned taconds;
    /*   PREVCHAR  1-15  -- Environment list of prev chars
     *   WFINAL      16  -- May occur word final
     *   NWFINAL     32  -- May not occur word final
     *   NASAL      128  -- Nasal allomorph (for conditional
                           nasalization)
     */
  struct TD_allo *talink;   /* link in list of allomorphs */
};

/* Available bits in TD_allo taconds
   available   256
   available   512
   available  1024
   available  2048
   available  4096
   available  8192
   available 16384
   available 32768
 */

/***************************** TD_suffix *******************
 */

struct TD_suffix {
  char *morphname;
  unsigned tprops;
    /* orderclass       -- low 8 bits
     * suffix flags     -- high 8 bits
     *   CONDNASAL 256  -- flag for conditional nasalization
     */
  struct TD_allo *ams;      /* head of list of allomorphs */
  struct TD_suffix *tdlink; /* link in list of suffixes */
};
```

```
/* Available bits in TD_suffix props
   available   512
   available  1024
   available  2048
   available  4096
   available  8192
   available 16384
   available 32768
 */

/****************************** tdnode ***********************
 */

struct tdnode {
  char *morphname;
  unsigned tprops;
    /* orderclass       -- low 8 bits
     * suffix flags     -- high 8 bits
     *   CONDNASAL 256  -- flag for conditional nasalization
     */
  struct TD_allo *ams;  /* head of linked list of allomorphs */
  struct tdnode *left;  /* link in list of suffixes */
  struct tdnode *right; /* link in list of suffixes */
};

/* Available bits in tdnode props
   available   512
   available  1024
   available  2048
   available  4096
   available  8192
   available 16384
   available 32768
 */
```

```
/* ADEFS.H definitions for analysis */

/************************** headlist ***********************
 * morphemes are added to the head as an analysis is made.
 * the head is a linked list with links from right to left.
 */

struct headlist {
  struct allomorph *allop;  /* each allomorph contains a
                             * pointer to a SD_suffix or a
                             * root morphname. */
  int categ;                /* copied from the dictionary */
  unsigned prop;            /* always 0 for ROOTS */
  int strsize;              /* accumulated string length of
                             * morphnames in head */
  struct headlist *hlink;
};

/* output from analysis is a linked list of strings
 * (strlist) each string is a sequence of morphnames
 * separated by spaces.  The number of elements in the
 * list is the ambiguity level.
 */

/************************** pairlist ***********************
 * list of paired strings for adhoc pairs
 */

struct pairlist {
  char *lstring;
  char *rstring;
  struct pairlist *plink;
};

/************************** fntab ***************************
 * function tables used to fill in fnlist for successorp and
 *   finalp
 * predname = the external name used to identify the
 *   predicate
 * fname    = a pointer to the actual function
 */
struct fntab {
  char *predname;
  int (*fname) ();
};

struct functab {
  char *predname;
  int (*fname) ();
  int *calls;
  int *fails;
};
```

```
/* RDEFS.H   12-Oct-83: definitions for rule structures
 * Rulelists are used in ANALYSIS to constrain the occurrence
 * of specific morphemes, in TRANSFER to specify necessary
 * conditions for insertion  and replacement.
 *
 * SYNTAX of rulelists:
 *
 * RULE:       "mlist1" ("mlist2") (CONDITION)* <CR>
 * CONDITION: / (TERM)* (REFPOINT) (TERM)*
 * REFPOINT:   _ | ..._ | _... | ..._...
 * TERM:       mname | [class] | ~TERM
 *
 * RULES ARE ORDERED!
 *
 * CLASSDEF:  [class] (mname)*
 * class definitions and references are conditionally
 * compiled for TRANSFER.
 */

/* term flags */
#define TERMTYPE 3
#define NODIR 0
#define LTERM 1
#define RTERM 2
#define CTERM 3

/* conditions on terms */
#define CONTIG 4
#define NEGATED 8

/* for class flags (all but low 4 bits) */
#define CLASSBASE 16

/*************************** term ************************
 */

struct term {
  char *mname;
  int flags;
  struct term *termlink;
};

/*************************** condit **********************
 * conditions
 */

struct condit {
  struct term *tlist;
  struct condit *clink;
};
```

```
/*************************** rule ************************
 * rule structure
 */

struct rule {
  char *kstring;        /* analysis or replacement key */
  char *istring;        /* insertion string */
  struct condit *condlist;
  struct rule *rlink;
};

/* classes for transfer rules */
#define MAXCLASS 12     /* number of bits for classes */

/*************************** ref_tab *********************
 * class reference table, a linked list,
 * each entry is a rule term which is a class reference.
 * The class number is to be filled in after the entire
 * data record has been processed
 */

struct ref_tab {
  char *cl_name;
  struct term *cl_ref;
  struct ref_tab *rtlink;
};

/* delimiters for key strings */
#define DELIMS '\"'
```

```
/*  TXTIN.C  25-JULY-82 Robert T Kasper
 ************************************************************
 *                                                          *
 *  INPUT:   text                                           *
 *  PROCESS: reads text a word at time, filling template    *
 *  OUTPUT:  file in the following database format.         *
 *           Each word is a record with fields:             *
 *           w = original word                              *
 *           f = preceding format marks                     *
 *           a = analysis (ambiguities and failures marked)*
 *           n = trailing nonalphabetics                    *
 *           c = capitalization                             *
 *                                                          *
 ************************************************************
 *                                                          *
 *   Modified for Tucanoan Languages by Robert B Reed       *
 *   23-Sep-82 -- look-ahead added                          *
 *   21-Jan-83 -- dash added to correctly interpret dashes  *
 *                  as format markers instead of parts of   *
 *                  words                                   *
 *   25-Jan-83 -- num_words changed to unsigned             *
 *   25-Jan-83 -- template changed to point to locations in*
 *                  internal buffer instead of allocating   *
 *                  and freeing up external space.          *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"

extern char *ssalloc();

/* word structures */
unsigned num_words;

/* special char lists */
extern char nonalpha[], word_init[];

#ifdef CADA

/* input orthochange table */
extern char *in_chg_tab;
extern int in_num_chgs;
extern char *change();

#endif

/***************************** fill_template ***************
 * fill_template parses input text and forms norm_word for
 * each word and its context.
 * input text:
 *   (<format mark> <break char>)* <alpha> (<nonalpha>)
```

```
 */

struct template *
fill_template(infp)
  FILE *infp;
{
  register struct template *norm_word;
  register char *textp, *cp;

  /* count of bytes left in inbuf */
  int buf_left, c;

  /* clear buffer for new template, textp at beginning of
   *   buffer
   * Allocate space for word template
   */
  norm_word = (struct template *)
                ssalloc(sizeof(struct template), INTS);

  textp = norm_word->buffer;
  buf_left = BUFSIZE;
  norm_word->non_alpha = NULL;
  norm_word->capital = NOCAP;
  if ( (c = getc(infp)) == EOF) {
     /* free buffer space */
     ssfree(norm_word, sizeof(struct template));
     return(NULL);                          /* no word */
  }

  /* Is character a format marker (@\-)? */
  if (index(FORMINIT, c) > -1) {
     if (c == '-' && !dash(infp)) /* part of a word */
        norm_word->format = NULL;
     else {
       norm_word->format = textp;
       /* Get all consecutive format markers */
       while (index(FORMINIT, c) > -1) {
         if (c == '-' && !dash(infp)) /* part of a word */
             break;
         else {
           *textp++ = c;
           --buf_left;

           if (c == '-' && dash(infp)) { /* part of a word */
              *textp++ = getc(infp);
              --buf_left;
              textp += get_chars_upto(word_init, &buf_left,
                        infp, textp);
           }
           else
             /* Read in chars up to white space */
             textp += get_chars_upto(BREAKC, &buf_left,
```

```
                        infp, textp);
            /* Read in chars up to next alpha or format
             *  marker
             */
            textp += get_chars_upto(word_init, &buf_left,
                        inf textp);
            c = getc(infp);
          }
        }
        if (norm_word->format == textp)
            norm_word->format = NULL;
        else {
          *textp++ = '\0';
          --buf_left;
        }
      }
    }
    else norm_word->format = NULL;

    norm_word->word = textp;
    --buf_left;
    if (index(word_init, c) > -1) {
       *textp++ = c;
       textp += get_chars_upto(nonalpha, &buf_left, infp,
                   textp) + 1;
       norm_word->non_alpha = textp;
    }
    else { /* leading nonalphabetics */
      *textp++ = '\0';              /* End of word */
      if (feof(infp)) {
         ssfree(norm_word,sizeof(struct template));
         return(NULL);
      }
      *textp = c;
      norm_word->non_alpha = textp++;
      --buf_left;
    }
    /* Get following non-alphabetic chars */
    textp += get_chars_upto(word_init, &buf_left, infp, textp)
               + 1;

    /* fill in capital field */
    cp = norm_word->word;
    while (*cp == '~' || *cp == '-' || *cp == '\'' ||
           *cp == '\"')
      cp++; /* Skip over initial special alphabetics */
    if (isupper(*cp)) {
       if (strlen(cp) == 1 || isupper(*(cp+1)))
          norm_word->capital = ALLCAP;
       else norm_word->capital = INITCAP;
    }
    else norm_word->capital = NOCAP;
```

```
   /* word in all lower case */
   for (; *cp; cp++)
       if (isupper(*cp))
          *cp = tolower(*cp);

   norm_word->new_word = textp;

   /* Return pointer to word template */
   return(norm_word);

} /* end fill_template */


/***************************** dash ************************
 * test for --; called if first hyphen is found
 */

dash(infp)
  register FILE *infp;
{
  register int c;

  c = getc(infp);        /* get the next char */
  ungetc(c,infp);        /* restore it to input stream */
  return(c == '-');      /* return whether a dash or not */
}


/***************************** TXTIN ************************
 * read a word and its related information
 */

struct template *
TXTIN(infp)
  register FILE *infp;
{
  register struct template *norm_word;

  if (feof(infp)) {
     fprintf(stderr, "\nINPUT: %u words.\n", num_words);
     return(NULL);
  }
  else {
    norm_word = fill_template(infp);

#ifdef CADA                                  /* discard ID line */
    if (norm_word) {
       if (sindex(norm_word->format,"\\id") > -1) {
          /* discard up to the newline */
          while (index(norm_word->non_alpha,'\n') == -1) {
            ssfree(norm_word, sizeof(struct template));
            norm_word = fill_template(infp);
```

```
            }
            /* discard the newline */
            ssfree(norm_word, sizeof(struct template));
            norm_word = fill_template(infp);
         }
      }
#endif

      /* orthography change if a word was found
       *   word -> original word
       *   new_word -> changed word
       */
      if (norm_word) {
         norm_word->new_word = change(norm_word->word,
                                      norm_word->new_word,
                                      in_chg_tab,in_num_chgs,FALSE);
         num_words++;
      }
      return(norm_word);
   }
}
```

```
/* ANPROC.C   2-SEP-83        Robert T Kasper
 ************************************************************
 *                                                          *
 * ANALYSIS                                                 *
 *   INPUT: taken from text if TXT_INPUT is defined,        *
 *          from database otherwise.  The format of the     *
 *          database is given below under output.           *
 *                                                          *
 *   PROCESS: analyzes text a word at time, using diction-  *
 *            ary and  rules to restrict analyses.          *
 *                                                          *
 *     OUTPUT:  file in the following database format.      *
 *       Each word is a record with fields:                 *
 *       w = original word                                  *
 *       f = preceding format marks                         *
 *       a = analysis (ambiguities and failures marked)     *
 *       n = trailing nonalphabetics                        *
 *       c = capitalization                                 *
 *       Accepts multiple files, each with separate output  *
 *         file.                                            *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Data Structures:                                        *
 *    *dict -> TRIE:                                        *
 *               char letters;                              *
 *               struct TRIE *subtries;                     *
 *               struct TRIE *tlink;                        *
 *               *amorphs-> allomorph:                      *
 *                          char *rest_key;                 *
 *                          char conds[1];                  *
 *                          union                           *
 *                            char *morphname;              *
 *                            *morpheme--> SD_suffix:       *
 *                                         char             *
 *                                           *morphname;    *
 *                                         char             *
 *                                           category[];    *
 *                                         unsigned         *
 *                                           props;         *
 *                                 char *nextword;          *
 *                                 struct allomorph *alink; *
 *                                                          *
 *    ctab[] cat_tab:        SPtab[], FPtab[] fntab:       *
 *      char *catname;         char *predname;             *
 *      int catnum;            int (*fname) ();            *
 *                                                          *
 *    *adhoc_list --> pairlist:                             *
 *                      char *lstring;                      *
 *                      char *rstring;                      *
 *                      struct pairlist *plink;             *
 *    *arules-> rule:                                       *
```

```
 *                 char *kstring;                                  *
 *                 char *istring;                                  *
 *                 condlist --> condit:                            *
 *                                  tlist -> term:                 *
 *                                             char *mname;        *
 *                                             int flags;          *
 *                                             struct term         *
 *                                               *termlink;        *
 *                                                                 *
 *    Change tables:  Format: <key>0<substitution>0                *
 *      char *in_chg_tab;                                          *
 *      int in_num_chgs;                                           *
 *                                                                 *
 *******************************************************************
 *                                                                 *
 *   Modified for Tucanoan Languages by Robert B Reed              *
 *     5-Oct-82 -- Separated out of ANRT.C to save space           *
 *    14-Oct-82 -- Discard header record produced by TEXT          *
 *                   analysis                                      *
 *    15-Oct-82 -- Process timer added                             *
 *    16-Dec-82 -- Counter variables changed to unsigned           *
 *    28-Apr-83 -- File header added                               *
 *    26-Sep-83 -- Prefix handling added                           *
 *                                                                 *
 *******************************************************************/

#include <stdio.h>
#include "defs.h"
#include "dict.h"
#include "adefs.h"
#include "rdefs.h"

extern FILE *ufopen();
extern ANALYSIS();
extern char *cpystr(), *encode();

/* for storage overflow message */
extern char errmsg[];

/* TRIE for SD dictionary */
extern struct TRIE dict;
extern struct TRIE prefdict;

/* word structures */
extern unsigned num_words;
unsigned awords;

#define NSPFNS 6
#define NFPFNS 3

/* ordered list of SP and FP functions
 * SP -- Successor Predicate
```

```
 * FP -- Final Predicate
 */
extern struct fntab SPtab[], FPtab[];
extern int (*SPlist[]) ();
extern int (*FPlist[]) ();
extern int spfuns, fpfuns;

/* statistical counts for each function */
long SPcalls[NSPFNS], SPfails[NSPFNS];
long FPcalls[NFPFNS], FPfails[NFPFNS];
unsigned ru_calls, ru_fails;

/* cumulative ambiguity counts */
unsigned ambig_list[MAXAMBIG];

/* output from analysis is a linked list of strings */
struct strlist *anlist = NULL;

/* Word structure pointers */
#ifndef TXT_INPUT
extern struct template norm_word;
#endif
struct template *cur_word, *next_word;
extern struct template *TXTIN();

extern char idline[];
extern int $$exst;      /* RT11 exit status; used to indicat
e system errors */

/****************** PROCESS *****************************/

PROCESS()
{
  struct TIME tbuf;
  FILE *infp, *outfp;
  extern FILE *$$flow;
  char fname[20];
  register int i, orig;
  char *infile;
  int sthrs, stmins, stsecs;   /* Used for process timing */

  infile = "\n\nInput file:  %s";

  fprintf(stderr,"\n\n%s\n\n%s ",
     "You may now remove the program tape.",
          "First INPUT file:");

  fgetss(fname, 20, stdin);

  do { /* process each input file to separate output files */

     /* open input file */
```

```
infp = ufopen(fname, READ);
printf(infile,fname);

/* open output file */
outfp = ufopen(NULL, WRITE);
printf("\nOUTPUT file: %s\n", outfp->io_name);
fprintf(stderr,
      "\nInclude original word in output (Y/N)? ");
fgets(fname, 20, stdin);
orig = (toupper(fname[0]) == 'Y');

/* initialize counts */
awords = num_words = 0;
for (i = 0; i < MAXAMBIG; i++)
    ambig_list[i] = 0;
for (i = 0; i < NSPFNS; i++)
    SPcalls[i] = SPfails[i] = 0;
for (i = 0; i < NFPFNS; i++)
    FPcalls[i] = FPfails[i] = 0;
ru_calls = ru_fails = 0;

/* get starting time */
rtime(&tbuf);
sthrs = tbuf.hour;
stmins = tbuf.minute;
stsecs = tbuf.second;

/* output file header */
fprintf(outfp,
     "f \\\\cid  %s:  Analyzed from %s on %s\\n\n",
      outfp->io_name,infp->io_name,idline);
fprintf(outfp,"a header\n");
fputs("n \\n\n",outfp);
fputs("\030\n",outfp);   /* EOR */

/* process text one word at a time */
fprintf(stderr,"\nStarting time: %02d:%02d:%02d\n",
                tbuf.hour,tbuf.minute,tbuf.second);

/* input from text: all words must be analyzed */

cur_word = TXTIN(infp);     /* Get first word */
showprog(num_words);
while (cur_word) {          /* While there are words */
  /* ANALYSIS parses words into strings of morphemes */
  /* Allow one word look ahead */
  next_word = TXTIN(infp);
  ANALYSIS(&prefdict, &dict, &dict);
  showprog(num_words);
  awords++;

  /* print results of analysis */
```

```
        DTBOUT(outfp, orig);
        sprintf(errmsg,"%u",num_words);

        cur_word = next_word;
        if (floc(outfp)) {        /* file too long */
           printf(
                "\nANALYSIS OUTPUT FILE TOO LONG: SORRY!!!\n");
           fmkdl(outfp);
           $$exst = 4;            /* notify system of error */
           goto err;
        }
      } /* next word */

      /* get ending time */
      rtime(&tbuf);
      fprintf(stderr,"\nEnding time: %02d:%02d:%02d\n",
                     tbuf.hour,tbuf.minute,tbuf.second);

     /* output statistical counts */
     outstats( tbuf.hour-sthrs,tbuf.minute-stmins,
           tbuf.second-stsecs);

     /* get next input file */
     fclose(infp);
     fclose(outfp);
     fprintf(stderr,
            "\nNext INPUT file (or RETURN if no more): ");
     fgetss(fname, 20, stdin);

   } while (fname[0]); /* until return */

err:
   /* pause to reinsert system tape */
   do {
      fprintf(stderr,
           "\nPress return when system tape is in place. ");
      fgetss(fname, 20, stdin);
   } while (*fname);              /* repeat until <RETURN> */

   /* set scrolling region from top to bottom */
   fprintf(stderr,"\033[1;24r");

   exit();
} /* end anproc */


/********************** outstats ************************
 * show cumulative statistics on stdout
 */

outstats(hour,min,sec)
   int hour,min,sec;  /* elapsed time */
```

```
{
  register int i, j;
  char buffer[10];

  printf("\n%s\n\n%s%u\n%s%s\n%s\n",
         "ANALYSIS STATISTICS:",
         "WORDS processed:  ", awords,
         "Processing time:  ", cnvtime(hour,min,sec,buffer),
         "Ambiguity levels");
  for (i = 0; i < MAXAMBIG; i++)
      printf("\n   %5u words with %2d analyses.",
          ambig_list[i],i);

  printf("\n\nFunction counts\n\nAdjacency Tests:\n");
  for (i = 0; SPlist[i]; i++) {
      /* find function in table */
      for (j = 0; SPtab[j].fname != SPlist[i] && j < NSPFNS;
               j++)
          ;
      printf("\n   %20s called %lu times, failed %lu.",
             SPtab[j].predname, SPcalls[i], SPfails[i]);
  }
  printf("\n\nFinal Tests:\n");
  for (i = 0; FPlist[i]; i++) {
      /* find function in table */
      for (j = 0; FPtab[j].fname != FPlist[i] && j < NFPFNS;
               j++)
          ;
      printf("\n   %20s called %lu times, failed %lu.",
             FPtab[j].predname, FPcalls[i], FPfails[i]);
  }
  printf(
    "\nRule Tests: called %u times, failed %u times.\n\f",
        ru_calls, ru_fails);
} /* end outstats */
```

```
/*   ANAL.C  25-JULY-82   Robert T Kasper
 ************************************************************
 *                                                          *
 *   Parses words into strings of morphemes                 *
 *   INPUT:  word field of template                         *
 *   OUTPUT: list of analyses for the word.                 *
 *           statistics for the analysis.                   *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"
#include "dict.h"
#include "adefs.h"
#include "rdefs.h"

extern char *ssalloc(), *cpystr(), *separate();
extern struct amlist *get_entries();
extern struct strlist *merge_results();

extern char *encode();

/* word from input */
extern struct template *cur_word;

/* output from analysis */
extern struct strlist *anlist;
int ambig = 0;

/* analysis rule list */
extern struct rule *arules;

/* cumulative ambiguity counts */
extern unsigned ambig_list[];

/* category table */
extern struct cat_tab ctab[];

/***************************** rulep ******************
 */

extern unsigned ru_calls, ru_fails;

rulep(anal)
  char *anal;
{
  register struct rule *rlp;
  register char *left, *right;
  char *sep_anal;
  int pos, stop;

/* initialize */
```

```
    ru_calls++;
    sep_anal = ssalloc(strlen(anal) + 1, BYTES);
    cpystr(sep_anal, anal);
    sep_anal = separate(sep_anal);
    stop = FALSE;

    /* check all rules in arules */
    for (rlp = arules; rlp && !stop; rlp = rlp->rlink) {
        /* apply constraints for all occurrences of key string */
        right = sep_anal;
        while ((pos = sindex(right, rlp->kstring)) > -1) {
          left = right + pos;
          right = left + strlen(rlp->kstring) - 1;
          if (!ruletest(sep_anal, left, right, rlp, NULL)) {
             ru_fails++;
             stop++;
             break;
          }
        }
    }
    ssfree(sep_anal, strlen(sep_anal) + 1);
    return(!stop);

} /* end rulep */

/*************************** head_create ******************
 * head_create fills a head structure from a root allomorph.
 */

struct headlist *
head_create(rhead, ap, cnum)
  register struct headlist *rhead;
  register struct allomorph *ap;
  register int cnum;
{
  if (cnum == 1 && (CLASS ap->conds[1]) TO1)
     rhead->categ = (CLASS ap->conds[1]) TO1;
  else
    if (cnum == 2 && (CLASS ap->conds[1]) TO2)
       rhead->categ = (CLASS ap->conds[1]) TO2;
  else /* category does not exist */
    return(NULL);

  /* category exists */
  rhead->allop = ap;
  rhead->prop = 0;
  rhead->strsize = strlen(ap->dp.morphname) + 1 +
                   strlen(ctab[(rhead->categ) - 1].catname);
  rhead->hlink = NULL;
  return(rhead);

} /* end head_create */
```

```
/****************************** form_anal ******************
 * makes a string of morphnames from a headlist
 * called recursively until headlist is exhausted
 * RETURNS: pointer to analysis string.
 */

char *
form_anal(head, anal)
  register struct headlist *head;
  char *anal;  /* points to beginning of analysis string,
                * NULL on first call */
{
  register char *cp;
  register int len;

  if (!anal)  /* first call, allocate space for string */
      anal = ssalloc(head->strsize + 1, BYTES);

  if (head->hlink) {  /* suffix */
     len = strlen(form_anal(head->hlink, anal));
     *(cp = anal + len) = ' ';
     cp = cpystr(++cp, head->allop->dp.morpheme->morphname);
  }
  else {  /* root */
    /* copy category name */
    cp = cpystr(anal, ctab[(head->categ) - 1].catname);
    *cp++ = ' ';
    /* copy morphname */
    cp = cpystr(cp, head->allop->dp.morphname);
  }
  return(anal);

} /* end form_anal */


/***************************** ANALYSIS *********************
 */

ANALYSIS (pdict, rdict, sdict)
   struct TRIE *pdict;          /* prefixes */
   register struct TRIE *rdict; /* roots    */
   register struct TRIE *sdict; /* suffixes */
{
   register int rootfound;

   anlist = NULL;
   ambig = 0;

   /* the word may begin with a root or prefix string */
   rootfound = root_anal (NULL,cur_word->new_word, rdict,
          sdict);
   if (pdict)          /* only do if there is a prefix list */
```

```
      prefix_anal (NULL, cur_word->new_word, pdict,
          rdict, sdict);

   /* output failures */
   if (ambig == 0) {              /* nothing analyzed */
      if (!rootfound)             /* no roots */
         printf("\nRF: %s",cur_word->word);
      else
        printf("\n%20s %s","AF:",cur_word->word);
   }
   /* update ambiguity list */
   if (ambig < MAXAMBIG)
      (ambig_list[ambig])++;
   else (ambig_list[MAXAMBIG])++;

} /* end ANALYSIS */


/* dummy headlist for prefixes */
struct headlist pref_head =
{
   NULL, P0, ORDERCLASS, 0, NULL
};


/************************** prefix_anal ******************
 * attempts to match one or more prefixes from pdict with
 *   an initial substring of wordstr
 * calls root_anal to analyze the remainder of the word
 * after the prefix.
 */

prefix_anal (plist, wordstr, pdict, rdict, sdict)
   struct headlist *plist;  /* last prefix analyzed */
   char *wordstr;
   struct TRIE *pdict;  /* prefixes */
   struct TRIE *rdict;  /* roots    */
   struct TRIE *sdict;  /* suffixes */
{
   struct headlist newhead;
   struct headlist *head;
   struct amlist *pfxset, *pfxp, *np;
   register struct allomorph *ap;
   register int i;
   char *newtail;

   /* create list of all prefixes whose string matches */
   if (strlen (wordstr))
      pfxset = get_entries (wordstr, SFX, pdict);
   else
      pfxset = NULL;
```

```
    /* use dummy prefix if none precede */
    head = (plist ? plist : &pref_head);

    /* for each prefix in pfxset, can it follow plist */
    for (pfxp = pfxset, ap = pfxp->amp;
         pfxp;
         pfxp = np, ap = pfxp->amp) {

         /* for each from/toclass pair */
         for (i = 0; ap->dp.morpheme->category[i]; i++) {
             newhead.categ = CLASS
                                 ap->dp.morpheme->category[i];
             newhead.prop = ap->dp.morpheme->props;
             newhead.allop = ap;

             if (successorp(head, &newhead, wordstr,
                     pfxp->alen) ) {

                 /* add newhead to head and form newtail */
                 newhead.hlink = (plist ? head : NULL);
                 newhead.strsize = head->strsize + 1 +
                   strlen(ap->dp.morpheme->morphname);
                 newtail = wordstr + pfxp->alen;
                 root_anal (&newhead, newtail, rdict, sdict);
                 prefix_anal (&newhead, newtail, pdict,
                     rdict, sdict);
             }
          }

         /* give back prefix set node */
         np = pfxp->amlink;
         ssfree(pfxp, sizeof(struct amlist));
      }

} /* end prefix_anal */


/************************* root_anal **********************
 * attempts to match one or more roots from rdict with an
 *   initial substring of wordstr
 * calls root_given_anal to analyze the remainder of word
 *   after the root
 */

root_anal (plist, wordstr, rdict, sdict)
   struct headlist *plist;  /* last prefix analyzed */
   char *wordstr;  /* portion of word still to be analyzed */
   struct TRIE *rdict;  /* roots    */
   struct TRIE *sdict;  /* suffixes */
{
  register struct allomorph *ap;
  register int i;
```

```
    struct amlist *rootlist, *rp, *np;
    struct headlist root_head;

  /* build list of root allomorphs which
   *   match an initial substring of the word (amlist).
   */

    rootlist = get_entries (wordstr, ROOTS, rdict);

    if (!rootlist)                    /* no roots found */
       return(FALSE);

    for (rp = rootlist, ap = rootlist->amp;
         rp;
         rp = rp->amlink, ap = rp->amp) {

        for (i = 1; i < 3; i++) {
            if (head_create(&root_head, ap, i) &&
               (!plist || successorp(plist, &root_head,
                      wordstr, rp->alen)))
            {
               root_head.hlink = plist;
               root_given_anal (&root_head,
                                wordstr + rp->alen, sdict,
                                TRUE);
            }
        }
    }

    /* rootlist exhausted */
    for (rp = rootlist; rp; rp = np) {
        np = rp->amlink;
        ssfree(rp->amp->dp.morphname,
            strlen(rp->amp->dp.morphname) + 1);
        ssfree(rp->amp,sizeof(struct allomorph));
        ssfree(rp, sizeof(struct amlist));
    }
    return(TRUE);                     /* roots were found */
  } /* end root_anal */


  /***************************** root_given_anal *************
   * finds all suffixes that match an initial part of the rest
   *   of the word.
   * called recursively to exhaust the word.
   * ARGUMENTS:
   *    head = list of morphemes already analyzed (headlist)
   *    tail = remainder of word string to be parsed
   *    sdict = suffix dictionary
   * RETURNS: number of successful analyses.
   */
```

```
root_given_anal(head, tail, sdict, add_null)
  struct headlist *head;
  char *tail;
  struct TRIE *sdict;
  int add_null;  /* FALSE: if no null morphemes may be added
 */
{
  struct headlist newhead;
  char *newtail, *astring;
  register int i;
  register struct allomorph *ap;
  struct amlist *sfxp, *sfxlist, *np;

  /* add more suffixes when:
   *    (1) more of the word remains to be analyzed.
   *    (2) a final category is needed and no previous
   *          attempts have been made to add null morphemes.
   */
  if (strlen(tail) || (!catFP(head) && add_null)) {
     /* create list of all suffixes whose string matches */
     sfxlist = get_entries(tail, SFX, sdict);
     add_null = (strlen(tail) != 0);

     /* for each allomorph in sfxlist */
     for (sfxp = sfxlist, ap = sfxp->amp;
          sfxp;
          sfxp = np, ap = sfxp->amp) {

         /* for each from/toclass pair */
         for (i = 0; ap->dp.morpheme->category[i]; i++) {
             newhead.categ = CLASS
                              ap->dp.morpheme->category[i];
             newhead.prop = ap->dp.morpheme->props;
             newhead.allop = ap;

             if (successorp(head, &newhead, tail,
                   sfxp->alen) ) {
                /* add newhead to head and form newtail */
                newhead.hlink = head;
                newhead.strsize = head->strsize + 1 +
                                  strlen(ap->dp.morpheme->
                                  morphname);
                newtail = tail + sfxp->alen;
                root_given_anal(&newhead, newtail, sdict,
                    add_null);
             }
         }

         /* give back suffix list node */
         np = sfxp->amlink;
         ssfree(sfxp, sizeof(struct amlist));
     }
```

```
    }

    else /* tail is null */
      if (finalp(head) &&
            rulep(astring = form_anal(head, NULL)))
          /* analysis successfully completed */
          anlist = merge_results(astring, anlist, &ambig);

  } /* end root_given_anal */
```

```
/*  APREDS.C  25-JULY-82          Robert T Kasper
 ************************************************************
 *                                                          *
 *  ANALYSIS test predicates                                *
 *  Two groups: SP = successor predicates                   *
 *              FP = final predicates                       *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Modified for Tucanoan by Robert B Reed                  *
 *   1-Oct-82 -- Specific characters for prevc tests added  *
 *   5-Oct-82 -- NextwFP added                              *
 *  16-Dec-82 -- prevc array added to predicate tests       *
 *  10-Jan-83 -- Tests arranged in alphabetical order       *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"
#include "dict.h"
#include "adefs.h"

extern char prevc[];

/****************************** successorp ***************
 * successorp -- the adjacency tests:
 *   left and right are headlist elements
 *   strp is the beginning of the right allomorph in the
 *     word string
 *   len is the length of the right allomorph
 * RETURNS: TRUE if all tests enabled in SPlist succeed.
 */

extern struct fntab SPtab[];
extern int (*SPlist[]) ();
extern int spfuns;

/* statistical counts for each function */
extern long SPcalls[], SPfails[];

successorp(left, right, strp, len)
  register struct headlist *left, *right;
  char *strp;
  int len;
{
  register int i;

  /* fail if any pred in SPlist fails,
   *   succeed if all succeed */
  for (i = 0; SPlist[i]; i++) {
      (SPcalls[i])++;
      if ( (*SPlist[i]) (left, right, strp, len) == FALSE )
```

```
{
            (SPfails[i])++;
            return(FALSE);
        }
    }
    return(TRUE);

} /* end successorp */

/************************** the adjacency tests ************
 ********************************* adhocSP ******************
 * fail if <left, right> is a pair on the adhoc list
 * should probably be last SPtest
 *
 * adhoc list is a linked pairlist
 */

extern struct pairlist *adhoc_list;

adhocSP(left, right, strp, len)
    struct headlist *left, *right;
    char *strp;
    int len;
{
    register struct pairlist *plp;
    register int val;

    for (plp = adhoc_list; plp; plp = plp->plink) {
        if ( ((CLASS left->allop->conds[1]) ATYPE) == SFX )
            val = streq(plp->lstring,
                        left->allop->dp.morpheme->morphname);
        else val = streq(plp->lstring,
                        left->allop->dp.morphname);

        if (val && streq(plp->rstring,
                        right->allop->dp.morpheme->morphname) )
            return(FALSE);
    }
    return(TRUE);
}

/********************************* catSP ********************
 * TOCLASS of left must be compatible with FROMCLASS of right
 */

catSP(left, right, strp, len)
    register struct headlist *left, *right;
    register char *strp;
    int len;
{
    if ( (left->categ TO) == (right->categ FROM) )
        return(TRUE);
```

```
    else
      if ( (left->categ TO) == V2 && (right->categ FROM) == V1
)
         return(TRUE);
      else
        return(FALSE);
}

/****************************** ordSP *********************
 * orderclasses must be nondecreasing from left to right
 */

ordSP(left, right, strp, len)
  register struct headlist *left, *right;
  register char *strp;
  int len;
{
  if (!(right->prop & ORDERCLASS) ||
        (left->prop & ORDERCLASS) <=
          (right->prop & ORDERCLASS) )
     return(TRUE);
  else
    return(FALSE);
}

/****************************** prevcSP ******************
 * checks whether right is preceded by an allowable char.
 */

prevcSP(left, right, strp, len)
  struct headlist *left;
  register struct headlist *right;
  register char *strp;
  int len;
{
  register int pcond;

  pcond = right->allop->ACONDS & PREVCHAR;
  --strp;      /* point to previous character */
  if (pcond)  /* previous character condition specified */
     return(*strp == prevc[pcond]);
  else return(TRUE);  /* no character condition specified */

}

/****************************** wfinalSP ******************
 * RETURNS: FALSE if right is WFINAL or NWFINAL and
 *   environment is incorrect, TRUE otherwise.
 */

wfinalSP(left, right, strp, len)
   struct headlist *left;
```

```
    register struct headlist *right;
    register char *strp;
    register int len;
{
    if (right->allop->ACONDS & WFINAL)
       return(*(strp + len) == '\0');
    else if (right->allop->ACONDS & NWFINAL)
           return(*(strp + len) != '\0');
         else return(TRUE);
}


/********************************** final tests *************
 * tests which apply to a whole string of morphemes
 * RETURNS: TRUE if all tests enabled in FPlist succeed.
 */

extern struct fntab FPtab[];
extern int (*FPlist[]) ();
extern int fpfuns;

/* statistical counts for each function */
extern long FPcalls[], FPfails[];

finalp(anal)
  register struct headlist *anal;
{
  register int i;
  register int j;

  /* fail if any pred in FPlist fails,
   *   succeed if all succeed */
  for (i = 0; FPlist[i]; i++) {
      (FPcalls[i])++;
      if ( (*FPlist[i]) (anal) == FALSE ) {
         (FPfails[i])++;
         return(FALSE);
      }
  }
  return(TRUE);
} /* end finalp */

/****************************** catFP ***********************
 * checks that TOCLASS of the rightmost morpheme in anal is
 * a member of the final category list
 * SHOULD BE FIRST FINAL TEST
 *
 */

catFP(anal)
   register struct headlist *anal;
{
```

```
    if ((anal->categ TO) >= FINALCATS)
       return(TRUE);
    else return(FALSE);
}

/************************** nextwFP ************************
 * checks that any morpheme requiring a specific following
 *   word has that requirement met.
 */
nextwFP (anal)
  register struct headlist *anal;
{
  register int test;
  extern struct template *next_word;

  if (next_word == NULL)          /* next_word is NULL for DTB
 input */
     return(TRUE);
  for (; anal; anal = anal->hlink) {
      if (anal->allop->nextword) { /* there is a next word */
         test = sindex(next_word->new_word,
                       anal->allop->nextword);
         if (test == 0)    /* word found */
            continue;
         else
           return(FALSE); /* required next word not found */
      }
  }
  return(TRUE);
}

/************************** ordFP ************************
 * checks that the orderclass sequence for a string of
 *   morphemes is nondecreasing. morphemes with undefined
 *   orderclass are ignored.
 */
ordFP (anal)
  register struct headlist *anal;
{
  register int last;
  register int this;

  for (last = 0; anal; anal = anal->hlink) {
      this = anal->prop & ORDERCLASS;
      if (last > this && this != 0)
         return(FALSE);
      else
        if (this)
           last = this;
  }
  return(TRUE);
}
```

```
/*  TSPROC.C  31-JULY-82   Robert T Kasper
 ************************************************************
 *                                                          *
 *  TRANSFER and SYNTHESIS with TD dictionary stored as     *
 *    linked list                                           *
 *                                                          *
 *    INPUT: each word is a record with fields:             *
 *      w = original word                                   *
 *      f = preceding format marks                          *
 *      a = analysis (ambiguities and failures marked)      *
 *      n = trailing nonalphabetics                         *
 *      c = capitalization                                  *
 *                                                          *
 *    PROCESS: apply TRANSFER and SYNTHESIS functions to    *
 *             each word                                    *
 *                                                          *
 *    OUTPUT: text file -- format restored from norm_word   *
 *            template                                      *
 *    Accepts multiple input files, each with separate      *
 *    output file.                                          *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Data Structures:                                        *
 *    *tdict -> TD_suffix:                                  *
 *                char *morphname;                          *
 *                unsigned tprops;                          *
 *                ams  ----------> TD_allo:                 *
 *                                   char *astring;         *
 *                                   unsigned taconds;      *
 *                                   struct TD_allo         *
 *                                     *talink;             *
 *                struct TD_suffix *tdlink;                 *
 *                                                          *
 *    *classes[MAXMORPH] -> strlist:                        *
 *                            char *string;                 *
 *                            struct strlist *slink;        *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Modified for Tucanoan Languages by Robert B Reed        *
 *    15-Feb-83 -- Modified to write file header            *
 *    15-Feb-83 -- Discard header record produced by        *
 *                 analysis                                 *
 *    15-Feb-83 -- Process timer added                      *
 *    15-Feb-83 -- Counter variables changed to unsigned    *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"
#include "dict.h"
```

```
#include "adefs.h"
#include "rdefs.h"

extern FILE *ufopen();
extern char *DTBIN(), *get_record();
extern FILE *ufopen();


/* dictionary and character storage uses ssalloc and ssfree:
 * storage overflow message
 */
extern char errmsg[];
extern int $$exst;      /* RT11 exit status; used to indicate
                         *   system errors */

/* linked list for TD dictionary */
extern struct TD_suffix *tdict;
extern unsigned num_words;
extern struct template norm_word;

/* cumulative ambiguity counts */
int syn_ambig_list[MAXAMBIG];

/* test counts */
extern unsigned wfincall, wfinfail, prevcall, prevfail,
     nascall, nasfail;
struct strlist *anlist = NULL;

/************************** PROCESS ********************
 */

PROCESS()
{
  long tbuf;
  FILE *infp, *outfp;
  register char *recp, *cp;     /* pointer to input record */
  register int i;
  char filedate[30];            /* file id date */
  char fname[20];               /* filename */

  /* get date for file id line */
  getdate(filedate);


  fprintf(stderr,"\n\n%s\n\n%s ",
          "You may now remove the program tape.",
          "First INPUT file:");

  fgetss(fname, 20, stdin);

  do { /* process each input file to separate output files */
     /* open input file */
```

```
infp = ufopen(fname, READ);
printf("\n\nInput file:  %s",fname);

/* open output file */
fname[0] = '\0';
outfp = ufopen(fname, WRITE);

for (cp = fname; *cp; cp++)
    *cp = toupper(*cp);     /* file name in all caps */

printf("\nOutput file: %s\n", fname);

/* initialize counts */
num_words = 0;
for (i = 0; i < MAXAMBIG; i++)
    syn_ambig_list[i] = 0;

wfincall = wfinfail = prevcall = prevfail =
     nascall = nasfail = 0;
/* get starting time */
tbuf = time(0);

/* output file header */
fprintf(outfp,"\\id %s %s\n",fname,filedate);

fprintf(stderr,"\nStarting time: %s\n",
     cnvtime(tbuf,fname));

/* discard analysis header record */
if (recp = get_record(infp))
   recfree(recp);

/* process text one word at a time */
while (recp = DTBIN(infp)) {
  TRANSFER(tdict);
  SYNTHESIS(tdict);
  TXTOUT(outfp);
  recfree(recp);
  sprintf(errmsg,"%u",num_words);
  if (floc(outfp)) {       /* file too long */
     printf("\nOUTPUT FILE TOO LONG: SORRY!!!\n");
     $$exst = 4;           /* notify system of error */
     goto err;
  }
} /* next word */

/* get ending time */
fprintf(stderr,
     "\nEnding time: %s\n",cnvtime(time(0),fname));

/* output statistical counts */
outstats(time(0) - tbuf);
```

```
        /* get next input file */
        fclose(infp);
        fclose(outfp);
        fprintf(stderr,
             "\nNext INPUT file (or RETURN if no more): ");
        fgetss(fname, 20, stdin);

    } while (fname[0]); /* until return */

err:
    /* pause to reinsert system tape */
    do {
        fprintf(stderr,
             "\nPress return when system tape is in place. ");
        fgetss(fname, 20, stdin);
    } while (*fname);              /* repeat until <RETURN> */

    /* set scrolling region from top to bottom */
    fprintf(stderr,"\033[1;24r");

    exit();
} /* end tsproc */


/******************************** outstats *****************
 * show cumulative statistics on stdout
 */

outstats(eltime)
    long eltime;  /* elapsed time */
{
    register int i;
    char buffer[10];

    printf("\n%s\n\n%s%u\n%s%s\n%s\n",
           "SYNTHESIS STATISTICS:",
           "WORDS processed:  ", num_words,
           "Processing time:  ", cnvtime(eltime,buffer),
           "Ambiguity levels");
    for (i = 0; i < MAXAMBIG; i++)
        printf("\n   %5u words with %2d ambiguities.",
            syn_ambig_list[i],i);
    putchar('\n');          /* double space */
    putchar('\n');
    printf("\nTests:\n");
    for (i = 1; i <= 3; i++)
        printf(
       "\n%12s environment test called %5u times, failed %5u.",
          (i == 1 ? "Nasal" : (i == 2 ? "Preceding" :
            "Word final")),
          (i == 1 ? nascall : (i == 2 ? prevcall : wfincall)),
          (i == 1 ? nasfail : (i == 2 ? prevfail : wfinfail)));
```

```
  putchar('\n');
  putchar('\f');
} /* end outstats */


/*************************** getdate(buf) *****************
 * convert time string: Thu Nov 15 14:38:32 1984
 *   to date: 15/NOV/84
 */

getdate(buf)
 char *buf;
{
  char timebuf[30];
  register char *cp;

  cpystr(timebuf,ctime(0));

  for (cp = &timebuf[4]; *cp; cp++)  /* start with month */
      if (*cp == ' ')                /* terminate strings */
         *cp = '\0';
      else
        *cp = toupper(*cp);

  sprintf(buf,"%s/%s/%s",
          &timebuf[8],&timebuf[4],&timebuf[22]);
}
```

```
/*  TRANSF.C  30-JULY-82   Robert T Kasper
 ************************************************************
 *                                                          *
 *  TRANSFER functions to modify analysis strings (anlist)  *
 *    (sets of strings of morphnames)                       *
 *                                                          *
 ************************************************************
 *                                                          *
 *   INPUT:   list of analyses                              *
 *   PROCESS: for each analysis in list apply any changes   *
 *              whose conditions are met:                   *
 *                  lexical changes                         *
 *                  rule changes                            *
 *   OUTPUT:  Transfered list of analyses                   *
 *                                                          *
 ************************************************************
 *  Data Structures:                                        *
 *    *trules -> rule:                                      *
 *                char *kstring;                            *
 *                char *istring;                            *
 *                condlist ---> condit:                     *
 *                                  tlist --> term:         *
 *                                              char *mname;*
 *                                              int flags;  *
 *                                              struct term *
 *                                                *termlink;*
 *                                  struct condit *clink;   *
 *                struct rule *rlink;                       *
 *                                                          *
 *    *anlist ------------> strlist:                        *
 *    *classes[MAXMORPH] -> strlist:                        *
 *                             char *string;                *
 *                             struct strlist *slink;       *
 *                                                          *
 *    Change tables:  Format:  <key>0<substitution>0        *
 *      char *rt_chg_tab    (root change table)             *
 *      char *lex_chg_tab   (lexical substitution table)    *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"
#include "dict.h"
#include "adefs.h"
#include "rdefs.h"

extern char *change(), *salloc(), *ssalloc(), *strcpy(),
     *cpystr();
extern char *separate(), *unseparate();
extern struct TD_suffix *get_morph();

/* pointer to input and output sets */
```

```
extern struct strlist *anlist;

/* ordered set of insertion, deletion and substitution rules
 */
extern struct rule *trules;
/* class lists */
extern struct strlist *classes[];
/* classes enabled for current analysis (each is a bit) */
int triggers = 0;

/* substitution tables */
extern char *rt_chg_tab;
extern int rt_num_chgs;
extern unsigned *root_tab;

/******************************** rtchange ****************
 * consistent changes on roots
 */

char *
rtchange(in,out,table,count,pos)
  char *in, *out;
  char *table[];                    /* root table */
  int count;                        /* number of entries */
  int pos;
{
  char root[BUFSIZE];           /* INPUT: |Cat|root|(sufs)| */
  register char *cp, *restp;
  register int i, j;

  /* copy category and root */
  cp = in + 1;
  cpystr(root,cp);
  if ( (i = index(root,'|')) > -1)
     root[i] = ' ';
  if ( (i = index(root,'|')) > -1)
     root[i] = '\0';
  restp = cp + strlen(root);

  /* find a match with or without category */
  for (j = 0; j > -1; j = index(&root[j],' ')) {
      if (j > 0)                    /* j points to a space */
         j++;                       /* skip the space */
      if ((i = find(&root[j],table,count)) > -1) {
         cp = cpystr(&root[j],table[i]+strlen(table[i])+ 1);
         cp = cpystr(cp,restp);
         *out = '|';
         cpystr(*(out + 1),root);
         while ( (j = index(out,' ')) > -1)
           *(out + j) = '|';
         return(out);
      }
```

```
    }
    return(in);
}


/********************************** rootchg ****************
 * RETURNS: stringp modified according to root change table
 *    (rt_chg_tab).
 * format of analysis: <CATEGORY> <ROOT> (<SFX>)*
 * all changes must begin no further right than the
 *    beginning of the root
 * a category may also be matched in a change.
 */

char *
rootchg(stringp)
  register char *stringp;  /* pointer to original string */
{
  register char *rbegin;
  register char *newstringp;     /* pointer to new string */
  char nroot[BUFSIZE];
  int pos;

  /* root change, no change past begin of root */

  pos = 2 + index(stringp + 1, '|');
  rbegin = rtchange( stringp, nroot, root_tab, rt_num_chgs,
              pos);

  /* free old string space if change was made */
  if (rbegin == nroot) {
     newstringp = ssalloc( strlen(rbegin) + 1, BYTES);
     cpystr( newstringp, rbegin);
     ssfree( stringp, strlen(stringp) + 1);
  }
  else newstringp = stringp;

  return(newstringp);
} /* end rootchg */

/********************************** findpos ****************
 * determines position for insertion of morphnames by
 *    orderclass.
 * mstring must be separated ("|" between morphnames)
 * ARGUMENTS:
 *    mstring  = morphname string: <CATEGORY> <ROOT> (<SFX>)*
 *    instring = string to be inserted
 *    sdict = TD suffix dictionary
 * RETURNS: pointer to site of insertion
 */

char *
```

```
findpos( mstring, instring, sdict)
  register char *mstring, *instring;
  struct TD_suffix *sdict;
{
  struct TD_suffix *imorph, *de;
  register char *isite;
  char *last;
  int iorder, pos, i;

  /* find last morphname in instring */
  last = instring++;
  while ( (pos = index(instring, '|')) > -1) {
    last = instring;
    instring += pos + 1;
  }

  /* get order class of last */
  iorder = 0;
  if (imorph = get_morph( last, sdict))
     iorder = imorph->tprops & ORDERCLASS;
  if (!iorder)
     iorder = 255;

  /* pass over mstring left to right:
   * position of insertion is before first morpheme with
   *   orderclass > iorder.
   */
  isite = mstring + 1;
  for (i = 0; (pos = index( isite, '|')) > -1; i++) {

      /* next morphname */
      isite += pos + 1;
      /* skip over cat and root */
      if (i > 0 && *isite &&
         (de = get_morph( isite, sdict)) &&
         (de->tprops & ORDERCLASS) > iorder)
            break;
  }

  return(isite - 1);

} /* end findpos */


/****************************** insertion *****************
 * ARGUMENTS:
 *   anal = analysis string
 *   sdict = TD suffix dictionary
 *   rlp = rule to be applied
 * RETURNS: new analysis string
 */
```

```
char *
insertion( anal, sdict, rlp)
  char *anal;
  struct TD_suffix *sdict;
  struct rule *rlp;
{
  int pos, val, i;
  char *cp;
  char *site;
  char *newcp;
  char *oldcp;
  char *nstringp;

  /* skip root and apply test between each morphname of anal
   */
  for (i = 0, cp = anal + 1; (pos = index( cp, '|')) > -1;
          i++) {
      /* next morphname */
      cp += pos;
      if (i > 0 && (val = ruletest( anal, cp, cp, rlp,
                    &triggers))) {
         /* valid insertion */
         if (val == TRUE)
            /* insert by orderclass */
            site = findpos( anal, rlp->istring, sdict);
         else /* insert at cp */
           site = cp;

         /* insert at site */
         nstringp = ssalloc( strlen(anal) +
                        strlen(rlp->istring) - 1, BYTES);
         for (oldcp = anal, newcp = nstringp;
              oldcp < site; *newcp++ = *oldcp++)
             ;
         newcp = cpystr( newcp, rlp->istring);
         cpystr( newcp, ++oldcp);
         ssfree( anal, strlen(anal) + 1);
         anal = nstringp;
         break;          /* only one insertion per rule */
      }

      /* advance past site */
      cp++;
  } /* no more sites */

  return(anal);
} /* end insertion */
```

```
/************************************ rulechg ***************
 * RETURNS: anal modified according to trules
 */

char *
rulechg( anal, sdict)
  register char *anal;
  struct TD_suffix *sdict;
{
  char *left, *right;
  int pos;
  char *site, *newcp, *oldcp, *nstringp;
  struct rule *rlp;

  /* check all rules in trules */
  for (rlp = trules; rlp; rlp = rlp->rlink) {
      if (rlp->kstring) {  /* replacement */
         /* check all sites where key is found in anal */
         site = anal;
         while ( (pos = sindex(site, rlp->kstring)) > -1) {
           /* key is present, find left and right
            *  boundaries */
           left = site + pos;
           right = left + strlen(rlp->kstring) - 1;
           /* check constraints */
           if (ruletest( anal, left, right, rlp, &triggers))
{
              /* replace kstring with istring */
              nstringp = ssalloc( strlen(anal) +
                                  strlen(rlp->istring)
                                  - strlen(rlp->kstring) + 1,
                                  BYTES);
              for (oldcp = anal, newcp = nstringp;
                   oldcp < left; *newcp++ = *oldcp++)
                  ;
              newcp = cpystr( newcp, rlp->istring);
              right = strcpy( newcp, right + 1);
              ssfree( anal, strlen(anal) + 1);
              anal = nstringp;
           }
           site = right;
         }
      } /* end replacement section */

      else  /* insertion */
        anal = insertion( anal, sdict, rlp);

  } /* next rule */
  return(anal);

} /* end rulechg */
```

```
/******************************** TRANSFER *****************
 */

TRANSFER( sdict)
  register struct TD_suffix *sdict;
{
  register struct strlist *stringlp;
  register char *cp;

  /* modify each analysis in anlist */
  for (stringlp = anlist; stringlp; stringlp =
        stringlp->slink) {
      cp = separate( stringlp->string);
      triggers = set_trig( cp);
      cp = rootchg( cp);
      cp = rulechg( cp, sdict);
      stringlp->string = unseparate( cp);
  }
} /* end TRANSFER */

/***************************** set_trig ********************
 * RETURNS: integer with bits set for each class represented
 in
 * morpheme string beginning at stringp
 */
set_trig( stringp)
  char *stringp;
{
  register struct strlist *stringlp;
  register int i;
  register int clbits;

  clbits = 0;

  /* check each class */
  for (i = 0; classes[i]; i++) {
      /* enable class if any key name is present */
      for (stringlp = classes[i]; stringlp; stringlp =
                    stringlp->slink) {
          if (sindex( stringp, stringlp->string) > -1) {
             clbits |= (1 << i);
             break;
          }
      }
  }
  return(clbits);

} /* end set_trig */
```

```
/****************************** find ********************
 */

find(root,table,high)
  char *root;                       /* root to look up */
  char *table[];                    /* root table */
  register int high;                /* number of entries */
{
  register int low, mid, cond;

  low = 0;
  --high;
  while (low <= high) {
    mid = (low+high) / 2;
    if ( (cond = strcmp(root,table[mid])) < 0)
        high = mid - 1;
    else if (cond > 0)
            low = mid + 1;
    else
      return(mid);                  /* index */
  }
  return(-1);
}
```

```
/*  SYNTH.C  22-JUL-82  Robert T Kasper
 ************************************************************
 *                                                          *
 *  SYNTHESIS functions                                     *
 *    INPUT:  anlist (strlist of morphnames)                *
 *    OUTPUT: synlist (strlist of words)                    *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Data Structures:                                        *
 *    *sdict -> TD_suffix:                                  *
 *                char *morphname;                          *
 *                unsigned tprops;                          *
 *                  ORDERCLASS                              *
 *                  CONDNASAL                               *
 *                ams  ------------> TD_allo:               *
 *                                     char *astring;       *
 *                                     unsigned taconds;    *
 *                                       PREVCHAR           *
 *                                       WFINAL             *
 *                                       NWFINAL            *
 *                                       NASALREQ           *
 *                                     struct TD_allo       *
 *                                       *talink;           *
 *                struct TD_suffix *tdlink;                 *
 *                                                          *
 *    *anlist, *synlist --> strlist:                        *
 *                            char *string;                 *
 *                            struct strlist *slink;        *
 *                                                          *
 *    Change tables:  Format:  <key>0<substitution>0        *
 *      char *reg_chg_tab   (regular sound change table)    *
 *      char *lex_chg_tab   (lexical substitution table)    *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Modified by Robert B Reed                               *
 *  17-Feb-83 -- Deleted functions not needed for Tucanoan  *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"
#include "dict.h"

extern struct strlist *merge_results();
extern char *change(), *salloc(), *ssalloc(), *strcpy();
extern struct TD_suffix *get_morph();

extern struct strlist *anlist;
struct strlist *synlist;
extern unsigned syn_ambig_list[];
```

```
extern unsigned num_words;

/* for lexical change */
extern char *lex_chg_tab;
extern int lex_num_chgs;

/* for regular sound change */
extern char *reg_chg_tab;
extern int reg_num_chgs;

/****************************** strike_allo ***************
 * eliminates allomorphs for which conditions do not hold,
 * taking first possible choice.
 * ARGUMENTS:
 *   tap      = pointer to TD_allo allomorph
 *   stringp  = beginning of morphname in analysis string
 *   entry    = entry in TD_suffix dict
 * RETURNS: string of allomorph chosen.
 *
 * Modifications by Robert B Reed:
 *   7-Mar-83 -- pass current entry rather than the entire
 *     dictionary
 */

char *
strike_allo( tap, stringp, entry)
  register struct TD_allo *tap;
  register char *stringp;
  register struct TD_suffix *entry;
{
  if ( (tap->talink == NULL) || conditionP( tap, stringp,
          entry) )
     return( tap->astring);
  else return( strike_allo( tap->talink, stringp, entry));
}


/* buffer space for single word */
char synword[BUFSIZE];

/****************************** genword ******************
 * creates a surface string from a list of morphnames
 * ARGUMENTS:
 *   mstring  = morphname string: <CATEGORY> <ROOT> (<SFX>)*
 *   sdict = TD suffix dict
 * passes over mstring left to right
 *   (1) genroot generates root form
 *   (2) strike_allo selects allomorphs for each morpheme
 *   (3) concatenates allomorphs to form TD word in synword
 * RETURNS: copy of synword string.
 */
```

```
char *prevmnamep;          /* pointer to previous morpheme */

char *
genword( mstring, sdict)
  char *mstring;
  struct TD_suffix *sdict;
{
  /* dictionary entry containing current morphname */
  struct TD_suffix *dictentry;
  char *astringp, *wp, *cp;
  register int i, pos;
  register char *mnamep;

  /* synword contains category and root */
  genroot (mstring);

  /* select allomorphs from left to right */
  for (i = 0, mnamep = mstring; *mnamep; i++) {
      /* skip over cat and root */
      if (i == 1)                  /* root */
         prevmnamep = mnamep;  /* needed to check for nasal
                                  * root  */
      if (i > 1) {
         /* get morpheme from dict */
         if (dictentry = get_morph( mnamep, sdict)) {
            /* eliminate bad allomorphs */
            astringp = strike_allo( dictentry->ams,
                        mnamep,dictentry);
            prevmnamep = synword + strlen(synword);
            strcpy(prevmnamep, astringp);
         }
         /* if not found, use morphname as string */
         else {
           cp = mnamep;
           prevmnamep = synword + strlen(synword);
           /* trail pointer to previous morpheme */
           wp = prevmnamep;
           while (*cp != ' ' && *cp != '\0')
             *wp++ = *cp++;
           *wp = '\0';
         }
      }
      /* next morphname */
      if ((pos = index( mnamep, ' ')) > -1)
         mnamep += pos + 1;
      else break;
  }

  /* allocate space for word string */
  wp = strcpy( ssalloc( strlen(synword) + 1, BYTES),
               synword);
  return( wp);
```

```
} /* end genword */

/****************************** SYNTHESIS ******************
 * for each alternative in anlist:
 *      (1) select allomorphs to form word
 *      (2) merge results into synlist
 */

SYNTHESIS( sdict)
  struct TD_suffix *sdict;
{
  register struct strlist *synthlp, *nextlp;
  register char *outp;
  int ambig;

  ambig = 0;
  synlist = NULL;

  for (synthlp = anlist; synthlp; synthlp = nextlp) {
      nextlp = synthlp->slink;
      outp = genword( synthlp->string, sdict);
      synlist = merge_results( outp, synlist, &ambig);
      /* give back anlist space */
      ssfree( synthlp->string, strlen(synthlp->string) + 1);
      ssfree( synthlp, sizeof(struct strlist));
  }

  /* update uncollapsed ambig counts */
  showprog(num_words);                /* echo on terminal */
  if (ambig < MAXAMBIG)
     (syn_ambig_list[ambig])++;
  else (syn_ambig_list[MAXAMBIG])++;

} /* end SYNTHESIS */

/****************************** genroot ******************
 * generates surface form of a root morpheme
 * ARGUMENTS:
 *    catroot = beginning of morphname string:
 *              <CATEGORY><space><ROOT-NAME><space>...
 */

genroot(catroot)
  char *catroot;
{
  /* root with lexical changes applied */
  char lexrtname[BUFSIZE];
  /* pointer to old and new forms of root */
  char *newrtname, *oldrtname;
  int len, pos1, pos2, n;
  register char *newp, *oldp;
```

```
    pos1 = index( catroot, ' ');  /* end of category */
    pos2 = ( (n = index( catroot + pos1 + 1,' ')) > -1 ?
                  n + pos1 + 1 :
                  strlen(catroot));    /* end of category */

    /* copy category and rootname */
    oldrtname = salloc( pos2 + (pos2 % 2));  /* align */
    for (oldp = catroot, newp = oldrtname, len = pos2;
             len; len--)
        *newp++ = *oldp++;
    *newp = '\0';

    /* apply lexical changes */
    oldrtname = change(oldrtname,lexrtname,lex_chg_tab,
                       lex_num_chgs,pos1 + 1);

    /* skip category */
    while (*oldrtname++ != ' ')
      ;

    /* apply reg sound change, if proto-form */
    if ( *oldrtname == '*') {
       /* eliminate proto-form mark */
       while (*oldrtname == '*')
         oldrtname++;
       newrtname = salloc( strlen(oldrtname) * 2);
       newrtname = change(oldrtname,newrtname,reg_chg_tab,
                      reg_num_chgs, FALSE);
    }
    else newrtname = oldrtname;
    strcpy( synword, newrtname);
} /* end genroot */

/********************************* conditionP ***************
 * ordered list of tests, each governed by a condition in
 * the taconds field for each allomorph.
 * ARGUMENTS:
 *    tap       = pointer to allomorph being tested
 *    stringp   = beginning of mname in analysis string
 *    entry     = pointer to entry in TD_suffix dict
 * RETURNS: false if any predicate fails, true if all
 *    succeed.
 */

conditionP (tap,stringp,entry)
  struct TD_allo *tap;
  register char *stringp;
  register struct TD_suffix *entry;
{
  register int pos;

  /* stringp on next morphname to right */
```

```
   if ( (pos = index(stringp,' ')) > -1)
      stringp += pos + 1;
   else
     stringp += strlen(stringp);   /* end of mname string */

   /* fail if any pred fails, succeed if all succeed */
   if (nasalCP(tap, stringp, entry) == FALSE )
      return(FALSE);
   if (prevcCP(tap, stringp, entry) == FALSE )
      return(FALSE);
   if (wfinalCP(tap, stringp, entry) == FALSE )
      return(FALSE);
   return(TRUE);

} /* end conditionp */


/********************************** nasalCP ***************
 */

unsigned nascall, nasfail;

nasalCP (tap, stringp, entry)
  struct TD_allo *tap;
  register char *stringp;
  struct TD_suffix *entry;
{
  register int curnasal, prevnasal;

  nascall++;
   /* Is current allomorph nasal? */
  curnasal = (tap->taconds & NASALREQ);

  /* Is previous morpheme nasal? */
  prevnasal = nasalP(prevmnamep);

  /* if morpheme is conditionally nasal *
  if (entry->tprops & CONDNASAL)/
     /* both morphemes must agree in nasalization */
     if ((prevnasal && !curnasal) ||
           (!prevnasal && curnasal)) {
        nasfail++;
        return(FALSE);       /* Fail-- nasal required */
     }
  return (TRUE);             /* Nasalization irrelevant */
} /* end nasalCP */

/********************************** prevcCP ****************
 */

unsigned prevcall, prevfail;
```

```
prevcCP (tap, stringp, entry)
  struct TD_allo *tap;
  register char *stringp;
  struct TD_suffix *entry;
{
  register char *last;
  register int pcond;
  extern char prevc[];

  prevcall++;
  /* point to last char in previous morpheme */
  last = prevmnamep;

  /* point to last char */
  while (*last != '\0' && *last != ' ')
    last++;
  --last;

  /* get index of character */
  pcond = (tap->taconds & PREVCHAR);
  if (pcond) {      /* previous char condition specified */
     if (*last == prevc[pcond])
        return (TRUE);           /* return result of test */
     else {
       prevfail++;
       return (FALSE);           /* return result of test */
     }
  }
  /* test succeeds if no condition specified */
  else return (TRUE);
}

/****************************** nasalP ********************
 * test string to see if it contains a nasal
 * return results
 */

nasalP(string)
  register char *string;
{
  for ( ;*string && *string != ' '; string++)
      switch (*string) {
        case 'm':
        case 'n':
        case '~': return (TRUE);
      }
  return (FALSE);
}

/****************************** wfinalCP ******************
 * conditions: stringp = next morpheme to right
 */
```

```
unsigned wfincall, wfinfail;

wfinalCP( tap, stringp, entry)
  register struct TD_allo *tap;
  register char *stringp;
  register struct TD_suffix *entry;
{
  wfincall++;
  if ( (tap->taconds & WFINAL) || (tap->taconds & NWFINAL)) {
     if ( *stringp) {                          /* not word final */
        if (!tap->taconds & NWFINAL) {
           wfinfail++;
           printf("WFINAL fails\n");
        }
        return(!tap->taconds & NWFINAL);
     }
     else {                                    /* word final */
       if (!tap->taconds & WFINAL) {
          wfinfail++;
           printf("WFINAL fails\n");
       }
       return(!tap->taconds & WFINAL);
     }
  }
  else return(TRUE);
}
```

```
/*  TXTOUT.C  24-JUL-82   Robert T Kasper
 ************************************************************
 *                                                          *
 *  TXTOUT reformats text from norm_word template created   *
 *    by INPUT.                                             *
 *  for CADA: uses synlist (a list of alternatives for      *
 *    each word)                                            *
 *                                                          *
 ************************************************************
 *                                                          *
 *  INPUT: format template containing the following         *
 *    information:                                          *
 *    w = original word  (unused by program)                *
 *    f = preceding format marks (encoded)                  *
 *    a = analysis (ambiguities and failures marked)        *
 *    n = trailing nonalphabetics (encoded)                 *
 *    c = capitalization                                    *
 *    Synthesis list.                                       *
 *  PROCESS: reconstruct text from template and synthesis   *
 *    list.                                                 *
 *  OUTPUT:  text reconstructed from format, capitaliza-    *
 *    tion, and punctuation information contained in        *
 *    template.                                             *
 *                                                          *
 ************************************************************
 *                                                          *
 *  Data Structures:                                        *
 *    template:                                             *
 *      char *format;                                       *
 *      char *word;                                         *
 *      char *non_alpha;                                    *
 *      int capital;                                        *
 *      char *new_word;                                     *
 *      char buffer[BUFSIZE];                               *
 *                                                          *
 *    *synlist --> strlist:                                 *
 *                   char *string;                          *
 *                   struct strlist *slink;                 *
 *                                                          *
 *    Change tables:  Format:  <key>0<substitution>0        *
 *      char *out_chg_tab    (output orthography change     *
 *                            table)                        *
 *                                                          *
 ************************************************************/

#include <stdio.h>
#include "defs.h"

extern char *change();
extern struct template *norm_word;

#ifdef CADA
```

```
/* list of synthesized words to be output */
extern struct strlist *synlist;

/* output orthochange table */
extern char *out_chg_tab;
extern int out_num_chgs;
extern char *change();

#endif

/***************************** TXTOUT ***********************
 * reconstruct format, punctuation, and capitalization
 *   information
 */

TXTOUT(outfp)
  register FILE *outfp;      /* output on outfp */
{

#ifdef CADA
  struct strlist *slp, *nlp;
  register int n;
#endif

  /* push parts of template onto output file */
  if (norm_word->format)
#ifdef CADA
     decode(norm_word->format, outfp);
#else
     fputs(norm_word->format, outfp);
#endif

#ifdef CADA
  if (synlist) {
     if (synlist->slink) {  /* ambiguity */
        /* count ambiguity level */
        for (n = 0, slp = synlist; slp; slp = slp->slink,
              n++)
           ;
        fprintf(outfp, "%%%1d", n);
        for (slp = synlist; slp; slp = nlp) {
           nlp = slp->slink;
           /* orthography change on word only */
           slp->string = change( slp->string,
                                 norm_word->buffer,
                                 out_chg_tab, out_num_chgs,
                                 FALSE);
           recap(slp->string);
           fprintf(outfp, "%s%%", slp->string);
           ssfree( slp->string, strlen(slp->string) + 1);
           ssfree( slp, sizeof(struct strlist));
        }
```

```
        }
        else {  /* only one alternative */
          /* orthography change on word only */
          synlist->string = change( synlist->string,
                                    norm_word->buffer,
                                    out_chg_tab, out_num_chgs,
                                    FALSE);
          recap(synlist->string);
          fputs( synlist->string, outfp);
          ssfree(synlist->string,strlen(synlist->string) + 1);
          ssfree( synlist, sizeof(struct strlist));
        }
    }
    else {  /* illegal word */
      norm_word->word = change( norm_word->word,
                                norm_word->buffer,
                                out_chg_tab, out_num_chgs, FALSE);
      recap(norm_word->word);
      fprintf(outfp, "%%1%s%%", norm_word->word);
    }

    if (norm_word->non_alpha)
       decode(norm_word->non_alpha, outfp);
    else putc(' ', outfp);

#else
    recap(norm_word->word);
    fputs( norm_word->word, outfp);

    if (norm_word->non_alpha)
       fputs( norm_word->non_alpha, outfp);
#endif

} /* end TXTOUT */

/***************************** recap ***********************
 * recap uses capital field of norm_word to reimpose
 * capitalization as it was in the input text.
 *
 * Modified by RBR
 *   15-Feb-83 -- allow diacritics to begin word, but
 *                capitalize the first real letter
 */

recap(cp)
  register char *cp;
{
  while (*cp == '~' || *cp == '-' || *cp == '\'' ||
          *cp == '\"')
    cp++;     /* skip special alphabetics */
  if (norm_word->capital == INITCAP)
     *cp = toupper(*cp);
```

INTERLIBRARY LOAN 09/01/87

PATRON # ZIEGLER VAL
ADDRESS # 2/9A/EP

ILL# 4442495
AUTHOR REED, ROBERT BRUCE, 1954-
TITLE COMPUTER ASSISTED DIALECT ADAPTATION : THE TUCANOAN EXPERIMENT /

YEAR
VERIFIED OCLC

This book was BORROWED from a source outside the Eastman Kodak Company. It was LOANED to us under the condition it be RETURNED to Carol Schiffler, Information Materials Processing Center, Building 82A, x78337 on or before the due date listed below. If you do not wish to lose your borrowing privileges, PLEASE comply with our request.

****
* September 25, 1987 *
****

PLEASE LEAVE THIS FORM IN THE BOOK

```
    else
      if (norm_word->capital == ALLCAP)
         for (; *cp; cp++)
             if (islower(*cp))
                *cp = toupper(*cp);
     /* otherwise no action */
  } /* end recap */

  #ifdef CADA

  /***************************** decode ***********************
   * restores special graphic chars in format fields of
   * template
   */

  decode(cp,outfp)
    register char *cp;
    register FILE *outfp;
  {
    while(*cp != '\0') {
      if (*cp == '\\')
         switch (*++cp) {
           case 'b':
             putc('\b',outfp);
             break;
           case 'f':
             putc('\f',outfp);
             break;
           case 'n':
             putc('\n',outfp);
             break;
           case 't':
             putc('\t',outfp);
             break;
           case '\0':
             continue;
           default:
             putc( *cp,outfp);
             break;
         } /* end switch */
      else putc( *cp,outfp);
      cp++;
    }

  } /* end decode */

  #endif
```

REFERENCES

Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. 1986. Compilers: principles, techniques, and tools. Reading, MA: Addison-Wesley Publishing Company.

Aleksander, Igor. 1984. Designing intelligent systems: an introduction. New York: Unipub.

Appelt, Douglas E. 1985. Planning English sentences. Cambridge: Cambridge University Press.

Andersen, Paul Kent. 1983. Word order typology and comparative constructions. Amsterdam: John Benjamins Publishing Company.

Anderson, Stephen R. 1985a. Inflectional morphology. Language typology and syntactic description, III: grammatical categories and the lexicon, ed. by Timothy Shopen, 150-201. Cambridge: Cambridge University Press.

-----. 1985b. Typological distinctions in word formation. Language typology and syntactic description, III: grammatical categories and the lexicon, ed. by Timothy Shopen, 150-201. Cambridge: Cambridge University Press.

Andrew, A. M. 1983. Artificial intelligence. Tunbridge Wells, Kent, England: Abacus Press.

Andreyev, N. 1967. The intermediary language as the focal point of machine translation. Machine translation, ed. by A. D. Booth, 1-28. Amsterdam: North-Holland Publishing Company.

Antilla, Raimo. 1972. An introduction to historical and comparative linguistics. New York: McMillan Publishing Company, Inc.

-----. 1978. Formalization as degeneration in historical linguistics. Readings in historical phonology: chapters in the theory of sound change, ed. by Philip Baldi and Ronald N. Werth, 348-376. University Park, PA: The Pennsylvania State University Press.

Arlotto, Anthony. 1972. Introduction to historical linguistics. Boston: Houghton Mifflin Company.

Bar-Hillel, Yehoshua. 1964a. The state of machine translation in 1951. Language and information: selected essays on their theory and application, ed. by Yehoshua Bar-Hillel, 153-165. Reading, MA: Addison-Wesley Publishing Company.

-----. 1964b. Aims and methods in machine translation. Language and information: selected essays on their theory and application, ed. by Yehoshua Bar-Hillel, 166-173. Reading, MA: Addison-Wesley Publishing Company.

-----. 1964c. A demonstration of the nonfeasability of fully automatic high quality translation. Language and information: selected essays on their theory and application, ed. by Yehoshua Bar-Hillel, 174-179. Reading, MA: Addison-Wesley Publishing Company.

-----. 1964d. The future of machine translation. Language and information: selected essays on their theory and application, ed. by Yehoshua Bar-Hillel, 180-184. Reading, MA: Addison-Wesley Publishing Company.

Barnes, Janet. 1977. Relaciones entre las proposiciones en la lengua tuyuca. Estudios Tucanos II, ed. by Carol Heinze, 99-128. (Serie Sintáctica 6). Bogotá: Ministerio de Gobierno.

-----. 1984. Evidentials in the Tuyuca verb. International Journal of American Linguistics 50:255-271.

Barnes, Janet, and Sheryl Takagi Silzer. 1976. Fonología del tuyuca. Sistemas fonológicos de idiomas colombianos, III, 123-138. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Barnwell, Katharine G. L. 1974. Introduction to semantics and translation. Horsleys Green, England: Summer Institute of Linguistics.

Barr, Avron, and Edward A. Feigenbaum. 1981. Understanding natural language. The Handbook of Artificial Intelligence, I, 223-321. Los Altos, CA: William Kaufmann, Inc.

Bates, Madeleine. 1978. The theory and practice of augmented transition network grammars. Natural language communication with computers, ed. by Leonard Bolc, 191-260. (Lecture Notes in Computer Science 63). Berlin: Springer-Verlag.

de Beaugrande, Robert. 1978. Factors in a theory of poetic translating. Assen, The Netherlands: Van Gorcum.

Beekman, John. 1965. Lexical equivalence involving consideration of form and function. Notes on translation 14:1-20. Santa Ana, CA: Summer Institute of Linguistics.

Beekman, John, and John Callow. 1974. Translating the Word of God. Grand Rapids, MI: Zondervan Publishing House.

Berry, Margaret. 1975. An introduction to systemic linguistics, 1: structures and systems. New York: St. Martin's Press.

-----. 1976. An introduction to systemic linguistics, 2: levels and links. New York: St. Martin's Press.

Bolc, Leonard. 1983. The design of interpreters, compilers, and editors for augmented transition networks. Berlin: Springer-Verlag.

Booth, A. Donald, and William N. Locke. 1955. Historical introduction. Machine translation of languages, ed. by William N. Locke and A. Donald Booth, 1-14. New York: The Technology Press of the MIT and John Wiley & Sons, Inc.

Booth, Kathleen H. V. 1967. Machine aided translation with a post-editor. Machine translation, ed. by A. D. Booth, 51-76. Amsterdam: North-Holland Publishing Company.

Borko, Harold. 1967. Automated language processing. New York: John Wiley and Sons, Inc.

Brislin, Richard W. 1976. Introduction. Translation: applications and research. New York: Gardner Press, Inc., and John Wiley & Sons, Inc.

Brower, Reuben A. 1959. On translation. Cambridge, MA: Harvard University Press.

Buhler, K. 1965. Sprachtheorie. Stuttgart: Fischer Verlag, as quoted by Juliane House.

Bush, Charles D. 1976. Fundamentals of junction grammar. (Revised and edited by Rey L. Baird and Brent Thompson.) Provo, Utah: Brigham Young University.

Bybee, Joan L. 1985. Morphology: a study of the relation between meaning and form. Amsterdam: John Benjamins Publishing Company.

Bynon, Theodora. 1977. Historical linguistics. (Cambridge Textbooks in Linguistics.) Cambridge: Cambridge University Press.

Carroll, John B. 1956. Introduction. Language, thought, and reality: selected writings of Benjamin Lee Whorf, ed. by John B. Carroll, 1-34. New York: The Technology Press of the Massachussetts Institute of Technology and John Wiley and Sons, Inc.

Carstairs, Andrew. 1981. Notes on affixes, clitics, and paradigms. Bloomington, IN: Indiana University Linguistics Club.

Casad, Eugene H. 1974. Dialect intelligibility testing. (Summer Institute of Linguistics Publications in Linguistics and Related Fields 38). Norman, OK: Summer Institute of Linguistics.

Casagrande, J. 1954. The ends of translation. International Journal of American Linguistics 20:335-340.

Catford, John C. 1965. A linguistic theory of translation. London: Oxford University Press.

Ceccato, Silvio. n.d. Linguistic analysis and programming for mechanical translation. Milan: Giangiacomo Feltrinelli Editore.

Charniak, Eugene, and Drew V. McDermott. 1985. Introduction to artificial intelligence. Reading, MA: Addison-Wesley Publishing Company.

Charniak, Eugene, Christopher K. Riesbeck, and Drew V. McDermott. 1980. Artificial intelligence programming. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Chomsky, Noam. 1965. Aspects of the theory of syntax. Cambridge, MA: The MIT Press.

Colmerauer, A., J. Dansereau, B. Harris, R. Kittredge, and M. van Caneghem. 1976. An English-French MT prototype. Papers in computational linguistics, ed. by Ferenc Papp and Gyorgy Szépe, 433-445. (Janua Linguarum, Series Maior, 91.) The Hague: Mouton.

Comrie, Bernard. 1981. Language universals and linguistic typology: syntax and morphology. Chicago: University of Chicago Press.

Comrie, Bernard, and Sandra A. Thompson. 1985. Lexical nominalization. Language typology and syntactic description, III: grammatical categories and the lexicon, ed. by Timothy Shopen, 150-201. Cambridge: Cambridge University Press.

Coughlin, Josette. 1985. Should CAT be taught in academic institutions. American Translators Association Conference-- 1985. Proceedings of the 26th Annual Conference of the American Translators Association, Miami, Florida, October 16-20, 1985, ed. by Patricia E. Newman, 253-257.

Crystal, David. 1981. Directions in applied linguistics. New York: Academic Press.

Delavenay, Emile. 1960. An introduction to machine translation. New York: Frederick A. Praeger, Publishers.

Denison, Norman. 1978. On pluralingualism and translation. Theory and practice of translation, ed. by Lillebill Grahs, Gustav Korlén, and Bertil Malmberg, 33-48. Nobel Symposium 19, Stockholm, September 6-10, 1976. Bern: Peter Lang.

Dostert, L. E. 1955. The Georgetown-IBM Experiment. Machine translation of languages, ed. by William N. Locke and A. Donald Booth, 124. New York: The Technology Press of the MIT and John Wiley & Sons, Inc.

Edmundson, H. P., ed. 1961. Proceedings of the National Symposium on Machine Translation. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Eibl, Georg. 1985. Current work on expert systems and natural language processing at Siemens. Artificial intelligence: towards practical application. Proceedings of the Joint Technology Assessment Conference of the Gottlieb Duttweiler Institute and the European Coordinating Committee for Artificial Intelligence, Ruschlikon, Zurich, Switzerland, 12-13 April 1984, ed. by Thomas Bernold and Gunter Albers, 97-106. Amsterdam: North-Holland.

Ferguson, Judy. A sketch of Cubeo grammar. Forthcoming.

Finch, C. A. 1969. An approach to technical translation: an introductory guide for scientific readers. Pergamon Press.

Fishman, J. A. 1973. The Whorfian hypothesis. Readings for applied linguistics, I, ed. by J. P. B. Allen and S. Pit Corder, 114-125. (The Edinburgh Course in Applied Linguistics). London: Oxford University Press.

Forster, Leonard. 1958. Translation: an introduction. Aspects of translation. (Studies in communication 2: The Communications Research Centre, University College, London.) London: Secker and Warburg.

Franco García, Germán, and José Raúl Monguí Sánchez. Gramática Yebamasa: linguística aplicada. Bogotá: Editorial Stella.

Francis, Winthrop Nelson. 1983. Dialectology: an introduction. London: Longman.

Frawley, William. 1984. Prolegomenon to a theory of translation. Translation: literary, linguistic, and philosophical perspectives, ed. by William Frawley, 159-178. Newark: University of Delaware Press.

Friedman, Joyce. 1971. Computational linguistics. A survey of linguistic science, ed. by William Orr Dingwall, 718-756. College Park, MD: University of Maryland.

Fuller, Frederick. 1984. The translator's handbook (with special reference to conference translation from French and Spanish). University Park, PA: The Pennsylvania State University press.

Garvin, Paul L. 1963. Natural language and the computer. (University of California Engineering and Sciences Extension Series). New York: McGraw-Hill Book Company.

-----. 1972. On machine translation. (Janua Linguarum Series Minor 128). The Hague: Mouton.

-----. 1976. Machine translation in the seventies. Papers in computational linguistics, ed. by Ferenc Papp and Gyorgy Szépe, 445-459. (Janua Linguarum, Series Maior, 91.) The Hague: Mouton.

Givón, Talmy. 1978. Universal grammar, lexical structure and translatability. Meaning and translation: philosophical and linguistic approaches, ed. by Franz Guenthner and M. Guenthner-Reutter, 235-272. New York: New York University Press.

Gonzalez, Rafael C., and Michael G. Thomason. 1978. Syntactic pattern recognition: an introduction. Reading, MA: Addison-Wesley Publishing Company.

Gralow, Frances. 1984. Ko'rehuaja chʉ'ore cutuñu: Hablemos Coreguaje, I. Lomalinda, Meta, Colombia: Editorial Townsend.

Gralow, Frances, Dorothy Cook, and Carolyn Muller Young. 1984. Fonología del coreguaje. Sistemas Fonológicos de idiomas colombianos, V, 59-79, ed. by Valentim Gonzáez B. Translated by Melva Hernández. Lomalinda, Meta, Colombia: Editorial Townsend.

Grimes, Barbara F. 1984. Languages of the World: Ethnologue. Tenth Edition. Dallas: Wycliffe Bible Translators.

Grimes, Joseph E., ed. 1975. Network grammars. (Summer Institute of Linguistics Publications in Linguistics and Related Fields 45). Norman, OK: Summer Institute of Linguistics.

-----. 1983. Affix positions and cooccurrences: The PARADIGM program. Summer Institute of Linguistics Publications in Linguistics 69. Dallas, TX: Summer Institute of Linguistics.

Guy, J. B. M. 1980a. Experimental glottochronology: basic methods and results. (Pacific Linguistics, Series B, 75). Canberra: The Australian National University.

-----. 1980b. Glottochronology without cognate recognition. (Pacific Linguistics, Series B, 79). Canberra: The Australian National University.

Hays, David G. 1977. Machine translation and abstract terminology. Studies in descriptive and historical linguistics: Festschrift for Winfred P. Lehman, ed. by Paul J. Hopper, 95-108. (Amsterdam Studies in the theory and history of linguistic science IV: Current issues in linguistic theory 4. Edited by E. F. K. Koerner.) Amsterdam: John Benjamins B. V.

Henisz-Dostert, Bozena, R. Ross Macdonald, and Michael Zarechnak. 1979. Machine translation. Trends in Linguistics 11. The Hague: Mouton Publishers.

Hjelmslev, Louis. 1970. Language: an introduction. Translated from Danish by Francis J. Whitfield. Madison: The University of Wisconsin Press.

Hoenigswald, Henry M. 1960. Language change and linguistic reconstruction. Chicago: The University of Chicago Press.

Hooper, Joan B. 1976. An introduction to natural generative phonology. New York: Academic Press.

Hopcroft, John E., and Jeffrey D. Ullman. 1979. Introduction to automata theory, languages, and computation. Reading, MA: Addison-Wesley Publishing Company.

House, Juliane. 1977. A model for translation quality assessment. Tubingen: TBL Verlag Gunter Narr.

Hugh-Jones, Christine. 1979. From the Milk River: spatial and temporal processes in Northwest Amazonia. Cambridge: Cambridge University Press.

Hugh-Jones, Stephen. 1979. The palm and the Pleides: initiation and cosmology in Northwest Amazonia. Cambridge: Cambridge University Press.

Hunt, Earl B. 1975. Artificial intelligence. New York: Academic Press.

Hutchins, W. J. 1984. Machine translation and machine-aided translation. Translation: literary, linguistic, and philosophical perspectives, ed. by William Frawley, 93-149. Newark: University of Delaware Press.

Ilek, Bohuslav. 1970. On translating images. The nature of translation: Essays on the theory and practice of literary translation, ed. by James S. Holmes, 135-138. The Hague: Mouton and Company and the Publishing House of the Slovak Academy of Sciences, Bratislava.

Isabelle, Pierre, and Laurent Borbeau. 1985. TAUM-AVIATION: its technical features and some experimental results. Computational linguistics 11:18-27. January-March 1985

Jeffers, Robert J. and Ilse Lehiste. 1979. Principles and methods for historical linguistics. Cambridge, MA: The MIT Press.

Johnson, Orville, and Catherine Peeke. 1962. Phonemic units in the Secoya word. Studies in Ecuadorian Languages, I, ed. by Benjamin Elson, 78-95. Norman, OK: Summer Institute of Linguistics.

Johnson, Roderick. 1983. Parsing with transition networks. Parsing natural language, ed. by Margaret King, 59-72. New York: Academic Press.

Jones, Wendell. A sketch of Southern Barasano grammar. Forthcoming.

Karn, Gloria Jean. 1976. Tuyuca plot structure-- a pilot study. M.A. Thesis presented to the Faculty of the University of Texas at Arlington.

Kasper, Robert and David Weber. Programmer's Guide to the CQAP. Unpublished manuscript.

-----. User's Guide to the CQAP. Unpublished manuscript.

Kay, Martin. 1977. Morphological and syntactic analysis. Linguistic structures processing, ed. by Antonio Zampolli, 131-234. (Fundamental Studies in Computer Science 5). Amsterdam: North-Holland Publishing Company.

Kaye, Jonathan D. 1971. Nasal harmony in Desano. Linguistic Inquiry 2.37-56.

Key, Mary Ritchie. 1979. The grouping of South American Indian languages. Ars Linguistica 2: Commentationes analyticae et criticae. Tubingen: Gunter Narr Verlag.

-----. 1981. Intercontinental linguistic connections. (Humanities inaugural lecture series.) Irvine: University of California.

Kinch, Rodney A. 1977. El enfoque temático vs. el enfoque no temático en yurutí. Estudios Tucanos II, ed. by Carol Heinze, 129-173. (Serie Sintáctica 6). Bogotá: Ministerio de Gobierno.

King, Robert D. 1969. Historical linguistics and generative grammar. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Klumpp, James, and Deloris Klumpp. 1973. Sistema fonológico del piratapuyo. Sistemas Fonológicos de Idiomas colombianos, II, 107-120. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Koerner, E. F. K. 1977. The Humboldtian trend in linguistics. Studies in descriptive and historical linguistics: Festschrift for Winfred P. Lehmann, ed. by Paul J. Hopper, 145-158. (Amsterdam Studies in the theory and history of linguistic science IV: Current issues in linguistic theory 4. Edited by E. F. K. Koerner.) Amsterdam: John Benjamins B. V.

Kreckel, Marga. 1981. Communicative acts and shared knowledge in natural discourse. New York: Academic Press.

Kress, Guntter R. 1976. Halliday: system and function in language. London: Oxford University.

Kuhn, Thomas S. 1970. The structure of scientific revolutions. Second edition, enlarged. Chicago: University of Chicago Press.

Kulagina, O. S., and I. A. Mel'cuk. 1967. Automatic translation: some theoretical aspects and the design of a translation system. Machine translation, ed. by A. D. Booth, 137-172. Amsterdam: North-Holland Publishing Company.

Lamb, Sydney M. 1973. Linguistic and cognitive networks. Readings in stratificational linguistics, ed. by Adam Makkai and David G. Lockwood, 60-83. University, AL: The University of Alabama Press.

-----. 1973. Stratificational linguistics as a basis for machine translation. Readings in stratificational linguistics, ed. by Adam Makkai and David G. Lockwood, 34-59. University, AL: The University of Alabama Press.

Larson, Mildred. 1984. Meaning-based translation: a guide to cross-language equivalence. Lanham, MD: University Press of America.

Lawson, Veronica. 1982. Practical experience of machine translation. Amsterdam: North-Holland Publishing Company.

Lehmann, W. P. 1959. Machine Language Translation Study: First Quarterly Progress Report -- 1 May 1959 - 31 July 1959. Fort Monmouth, NJ: U.S. Army Signal Engineering Laboratories.

Lightfoot, David W. 1979. Principles of diachronic syntax. Cambridge: Cambridge University Press.

Loukotka, Cestmír. 1968. Classification of South American Indian languages. Los Angeles: University of California.

Mann, William C. 1982. Text generation. American journal of computational linguistics, 8:62-69. April-June 1982.

-----. 1983. A linguistic overview of the NIGEL text generation grammar. The Tenth LACUS forum 1983, ed. by Alan Manning, Pierre Martin, and Kim McCalla, 255-265. Columbia, SC: Hornbeam Press, Inc.

Mann, William C., and Christian Matthiessen. 1983. NIGEL: a systemic grammar for text generation. Marina del Rey, CA: University of Southern California Information Sciences Institute.

McCorduck, Pamela. 1979. Machines who think. San Francisco: W. H. Freeman.

Masterman, Margaret. 1979. The essential skills to be acquired for machine translation. Translating and the computer: proceedings of a seminar, London, 14th November, 1978, ed. by Barbara M. Snell, 159-180. Amsterdam: North-Holland Publishing Company.

Matthews, Peter H. 1974. Morphology: an introduction to the theory of word-structure. Cambridge: Cambridge University Press.

Melby, Alan K. 1985. Machine translation vs. translator aids: a false dichotomy. Humans and machines, ed. by Stephanie Williams, (Delaware Symposium 4), 45-53. Proceedings of the 4th Delaware Symposium on Language Studies, October 1982, The University of Delaware. Norwood, NJ: Ablex Publishing Company.

Metzger, Ronald G. 1981. Gramática popular del carapana. Bogotá: Ministerio de Gobierno.

Metzger, Ronald G., and Lois Metzger. 1973. Fonología del carapana. Sistemas fonológicos de idiomas colombianos, II, 121-132. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Miller, Marion. 1976. Fonología del desano. Sistemas fonológicos de idiomas colombianos, III, 105-112. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Mishkoff, Henry C. 1985. Understanding artificial intelligence. Dallas: Texas Instruments.

Moto-oka, Tohru. 1982. Challenge for knowledge information processing systems: (preliminary report on Fifth Generation computer systems). Fifth Generation computer systems, Proceedings of the international conference on Fifth Generation computer systems, Tokyo, Japan, October 19-22, 1981, ed. by Tohru Moto-oka, 3-93. Amsterdam: North-Holland Publishing Company.

Nagler, Christine, and Beverly Brandrup. 1979. Fonología del siriano. Sistemas fonológicos de idiomas colombianos, IV, 101-126. Translated by Luis Emiro Henríquez Girón. Bogotá: Ministerio de Gobierno.

Nida, Eugene A. 1949. Morphology: the descriptive analysis of words. Ann Arbor: The University of Michigan Press.

-----. 1964. Toward a science of translating, with special reference to principles and procedures involved in Bible translating. Leiden: E. J. Brill.

-----. 1976. A framework for the analysis and evaluation of theories of translation. Translation: applications and research, ed. by Richard W. Brislin, 47-91. New York: Gardner Press, Inc., and John Wiley and Sons, Inc.

Nida, Eugene A. and William D. Reyburn. 1981. Meaning across cultures. Maryknoll, NY: Orbis Books.

Nida, Eugene A., and Charles R. Taber. 1974. The theory and practice of translation. Leiden: E. J. Brill.

Peat, David. 1985. Artificial intelligence: how machines think. New York: Baen Enterprises.

Petrick, S. R. 1977. On natural language based computer systems. Linguistic structures processing, ed. by Antonio Zampolli, 313-340. (Fundamental Studies in Computer Science 5). Amsterdam: North-Holland Publishing Company.

Pinchuk, Isadore. 1977. Scientific and technical translation. Boulder, CO: Westview Press.

Pollack, Jordan, and David L. Waltz. 1986. Interpretation of natural language. BYTE 11:189-198, February 1986.

Raffel, Burton. 1971. The forked tongue: a study of the translation process. The Hague: Mouton.

Reed, Robert B. 1985. CADA: An overview of the Tucanoan experiment. Notes on Computing, 11:6-19. August 1985.

Reichman, Rachel. 1985. Getting computers to talk like you and me: discourse context, focus, and semantics (An ATN model). Cambridge, MA: The MIT Press; a Bradford Book.

Rhodes, Ida. 1967. The importance of the glossary storage in machine translation. Machine translation, ed. by A. D. Booth, 429-449. Amsterdam: North-Holland Publishing Company.

Rich, Elaine. 1983. Artificial intelligence. New York: McGraw-Hill Book Company.

Rieger, Charles J. 1978. Computational linguistics. A survey of linguistic science, ed. by William Orr Dingwall, Second edition, 97-134. Stamford, CT: Greylock Publishers.

Ritchie, Graeme D. 1980. Computational grammar: an artificial intelligence approach to linguistic description. Sussex: The Harvester Press.

Rosner, Michael. 1983. Production systems. Parsing natural language, ed. by Margaret King, 35-58. New York: Academic Press.

Sacerdoti, Earl D. 1977. A structure for plans and behavior. New York: Elsevier.

Sager, Juan C. 1979. Multilingual communication: chairman's introductory review of translating and the computer. Translating and the computer: proceedings of a seminar, London, 14th November, 1978, ed. by Barbara M. Snell, 1-27. Amsterdam: North-Holland Publishing Company.

Sager, Naomi. 1981. Natural language information processing: a computer grammar of English and its applications. Reading, MA: Addison-Wesley Publishing Company.

Salser, J. K., and Neva Salser. 1976. Fonología del cubeo. Sistemas fonológicos de idiomas colombianos, III, 71-81. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

-----. 1977. Some features of Cubeo discourse and sentence structure. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part II, ed. by Robert E. Longacre and Frances Woods, 253-272. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

Samlowski, Wolfgang. 1976. Case grammar. Computational semantics: an introduction to artificial intelligence and natural language comprehension, ed. by Eugene Charniak and Yorick Wilks, 55-72. Amsterdam: North-Holland Publishing Company.

Schank, Roger C., and Robert P. Abelson. 1977. Scripts, plans, goals and understanding: an inquiry into human knowledge structures. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Selver, Paul. 1966. The art of translating poetry. Boston: The Writer, Inc.

Sigurd, Bengt. 1978. Machine translation -- state of the art. Theory and practice of translation, ed. by Lillebill Grahs, Gustav Korlén, and Bertil Malmberg, 33-48. Nobel Symposium 19, Stockholm, September 6-10, 1976. Bern: Peter Lang.

Simmons, Robert F. 1984. Computations from the English: A procedural logic approach for representing and understanding English texts. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Simons, Gary F. 1983. Language variation and limits to communication. Dallas: Summer Institute of Linguistics.

Slocum, Jonathan. 1981. A practical comparison of parsing strategies for machine translation and other natural language processing purposes. Unpublished Ph. D. dissertation: The University of Texas at Austin.

-----. 1985. A survey of Machine Translation: its history, current status, and future prospects. Computational Linguistics, 11:1-17, January-March 1985.

Slype, G. Van, J. F. Guinet, F Seitz, E. Benejam. 1983. Better translation for better communication: A survey of the translation market, present and future, prepared for the Commission of European Communities, Directorate-General Information Market and Innovation, by Bureau Marcel van Dijk, Brussels, and PA Consieller de Direction, Paris. New York: Pergamon Press.

Small, Steven Lawrence. 1980. Word expert parsing: a theory of distributed word-based natural language understanding. Unpublished Ph. D. dissertation: University of Maryland.

Smith, N. V. 1982. Mutual knowledge. New York: Academic Press.

Smith, Richard Dean. 1976. Una gramática tagmémica del barasano del sur. Bogotá: Ministerio de Gobierno.

-----. 1977. Southern Barasano sentence structure. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part II, ed. by Robert E. Longacre and Frances Woods, 175-206. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

-----. 1979. Algunos rasgos contrastivos del barasano del sur y el español. Estudios Tucanos III, ed. by Nancy L. Morse, 9-83. (Serie Sintáctica 7). Bogotá: Ministerio de Gobierno.

Smith, Richard Dean, and Connie Smith. 1976. Fonología del barasano del sur. Sistemas fonológicos de idiomas colombianos, III, 95-104. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Snell, Barbara M. 1979. Introduction. Translating and the computer: proceedings of a seminar, London, 14th November, 1978, ed. by Barbara M. Snell, v-vii. Amsterdam: North-Holland Publishing Company.

Sparck Jones, Karen, and Martin Kay. 1973. Linguistics and information science. New York: Academic Press.

Sparck Jones, Karen, and Yorick Wilks, eds. 1983. Automatic natural language parsing. Chichester: Ellis Horwood Limited, Publishers.

Steiner, George. 1975. After Babel: aspects of language and translation. Oxford: Oxford University Press.

Stolte, Joel, and Nancy Stolte. 1976. Fonología del barasano del norte. Sistemas fonológicos de idiomas colombianos, III, 83-94. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Strom, Clayton L. A sketch of Retuarã grammar. Forthcoming.

Tait, J. I. 1985. Generating summaries using a script-based language analyzer, Progress in artificial intelligence, ed. by Luc Steels and J. A. Campbell, 312-318. Chichester: Ellis Horwood, Limited.

Tennant, Harry. 1981. Natural language processing: an introduction to an emerging technology. New York: Petrocelli Books.

Thomas-Flinders, Tracy, ed. 1981. Inflectional morphology: introduction to the extended word-and-paradigm theory. (UCLA Occasional Papers 4: Working Papers in Morphology.)

Tosh, Wayne. 1965. Syntactic translation. The Hague: Mouton and Company.

Tovar, Antonio, and Consuelo Larrucea de Tovar. 1984. Otras lenguas del Marañon al Orinoco, con el grupo tucano. Catálogo de las lenguas de América del sur, con clasificaciones, indicaciones tipológicas, bibliografía y mapas. Nueva edición refundida, 154-167. Madrid: Editorial Gredos.

Trudgill, Peter. 1983. On dialect: social and geographical perspectives. New York: New York University Press.

Tsiapera, Maria. 1971. Generative studies in historical linguistics. (Current Inquiry into Language & Linguistics 2.) Champaign, IL: Linguistic Research, Inc.

Tucker, Allen B., Jr. 1979. Text processing: algorithms, languages, and applications. (Computer Science and Applied Mathematics). New York: Academic Press.

Varile, Giovanni B. 1983. Charts: a data structure for parsing. Parsing natural language, ed. by Margaret King, 73-88. New York: Academic Press.

Vazquez-Ayora, Gerardo. 1977. Introducción a la traductología: curso básico de traducción. Washington, D. C.: Georgetown University Press.

Waltz, Carolyn. 1976. Some observations on Guanano dialogue. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part I, ed. by Robert E. Longacre and Frances Woods, 67-110. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

Waltz, Nathan. 1976a. Discourse functions of Guanano sentence and paragraph. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part I, ed. by Robert E. Longacre and Frances Woods, 21-146. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

-----. 1976b. Hablemos el guanano: una gramática pedagógica del guanano. Bogotá: Ministerio de Gobierno.

Waltz, Nathan, and Carolyn Waltz. 1979. Fonología del guanano. Sistemas fonológicos de idiomas colombianos, I, ed. by Viola Waterhouse, 29-40. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Waltz, Nathan E., and Alva Wheeler. 1972. Proto Tucanoan. Comparative studies in Amerindian languages, ed. by Esther Matteson, 119-149. The Hague: Mouton and Company.

Weaver, Warren. 1955. Translation. Originally written in 1949, reprinted in Machine translation of languages, ed. by William N. Locke and A. Donald Booth, 15-23. New York: The Technology Press of the MIT and John Wiley and Sons, Inc.

Weber, David. 1976. Suffix-as-operator analysis and the grammar of successive encoding in Llacon (Huanuco) Quechua. (Documento de Trabajo 13). Yarinaçocha, Peru: Instituto Linguístico de Verano.

Weber, David J., and William C. Mann. 1980. Assessing the prospects for Computer-Assisted Dialect Adaptation in a particular language. Notes on Linguistics 15:29-40. July 1980.

-----. 1981. Prospects for Computer-Assisted Dialect Adaptation. American Journal of Computational Linguistics 7:165-177. July-September 1981.

Welch, Betty. 1977. Tucano discourse, paragraph, and information distribution. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part II, ed. by Robert E. Longacre and Frances Woods, 229-253. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

West, Birdie. 1977. Results of a Tucanoan syntax questionnaire pilot study. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part II, ed. by Robert E. Longacre and Frances Woods, 339-375. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

-----. 1980. Gramática popular del tucano. Bogotá: Ministerio de Gobierno.

West, Birdie, and Betty Welch. 1979. Sistema fonológico del tucano. Sistemas fonológicos de idiomas colombianos, I, ed. by Viola Waterhouse, 13-28. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Wheeler, Alva. 1970. Grammar of the Siona language: Colombia, South America. Unpublished Ph. D. dissertation: University of California, Berkeley.

Wheeler, Alva, and Margaret Wheeler. 1962. Siona phonemics. Studies in Ecuadorian Languages, I, ed. by Benjamin Elson, 96-113. Norman, OK: Summer Institute of Linguistics.

Whisler, David. 1976. Some aspects of Tatuyo discourse. Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador, Part I, ed. by Robert E. Longacre and Frances Woods, 207-252. (Summer Institute of Linguistics Publications in Linguistics and Related Fields, 52). Dallas: Summer Institute of Linguistics.

Whisler, David, and Janice Whisler. 1976. Fonología del tatuyo. Sistemas fonológicos de idiomas colombianos, III, 113-122. Translated by Jorge Arbeláez G. Bogotá: Ministerio de Gobierno.

Whorf, Benjamin Lee. 1956. Language, mind, and reality. Originally published in 1942 in The Theosophist (Madras, India), 63.1:281-291, (January 1942). Reprinted in Language, thought, and reality: selected writings of Benjamin Lee Whorf, ed. by John B. Carroll, 246-270. New York: The Technology Press of the Massachussetts Institute of Technology and John Wiley and Sons, Inc.

-----. 1973. Linguistic relativity. Readings in applied linguistics, I, ed. by J. P. B. Allen and S. Pit Corder, 103-113. (The Edinburgh Course in Applied Linguistics.) London: Oxford University Press.

Wilks, Yorick. 1979: Machine translation and artificial intelligence. Translating and the computer: proceedings of a seminar, London, 14th November, 1978, ed. by Barbara M. Snell, 27-44. Amsterdam: North-Holland Publishing Company.

-----. 1983. Machine translation and the artificial intelligence paradigm of language processes. Computers in language research, 2. Part I: Formalization in literary and discourse analysis. Part II: Notating the language of music, and the (pause) rhythms of speech, ed. by Walter A. Sedelow and Sally Yeates Sedelow, 61-111. New York: Mouton Publishers.

Wilss, Wolfram. 1982. The science of translation: problems and methods. Tubingen: Gunter Narr Verlag.

Winograd, Terry. 1983. Language as a cognitive process, I: syntax. Reading, MA: Addison-Wesley Publishing Company.

Winston, Patrick H. 1984. Artificial intelligence. Second edition. Reading, MA: Addison-Wesley Publishing Company.