This page intentionally left blank.

Graduate Institute of Applied Linguistics

Thesis Approval Sheet

This thesis, entitled

DESIGN OF AN ELECTRONIC METHOD FOR DESCRIBING WRITING SYSTEMS

written by

Eric Scott Albright

and submitted in partial fulfillment of the requirements for the degree of

Master of Arts in Applied Linguistics

has been read and approved

by the undersigned members of the faculty

of the Graduate Institute of Applied Linguistics

Gary F. Simons (Mentor)

Stephen L. Walter

Robert B. Reed

May 11, 2001

DESIGN OF AN ELECTRONIC METHOD FOR DESCRIBING WRITING SYSTEMS

By

Eric Scott Albright

Presented to the Faculty of
the Graduate Institute of Applied Linguistics
in partial fulfillment of the requirements
for the degree of

Master of Arts in Applied Linguistics

Graduate Institute of Applied Linguistics
June, 2001
Dallas, TX

THESIS DUPLICATION RELEASE

I hereby authorize the Graduate Institute of Applied Linguistics Library to duplicate this thesis when needed for research and/or scholarship.

Agreed: _____
(student signature)

Refused: _____
(student signature)

ABSTRACT

DESIGN OF AN ELECTRONIC METHOD FOR DESCRIBING WRITING SYSTEMS

Eric Scott Albright, M.A.

The Graduate Institute of Applied Linguistics, 2001

Supervising Professor: Gary F. Simons, Ph.D.

Although descriptions of writing systems abound, the literature conspicuously lacks a reference work that describes the information that should be present for a complete account of the workings of a writing system. As a result, the quality of writing system descriptions suffers, as key information is either lacking or inaccessible.

Writing system descriptions must describe the characteristics of each unit of writing together with its relationships to other pertinent units of writing and linguistics. Formal descriptions allow both machines and users to share the same source of information.

This thesis seeks to determine the slots that should be filled by information obtained from the systematic analysis of writing systems. These slots are organized into a formal model of writing system descriptions that captures the semantics of writing systems.

DEDICATION

In memory of Kenneth L. Pike who sat on the front porch of a teenager and convinced him that there is nothing in the world like the study of linguistics—especially when your wife is also a linguist.

To my linguist wife, who inspired, encouraged, and supported me during this study.

ACKNOWLEDGMENTS

This thesis would not be what it is today without the help of my mentor, Dr. Gary F. Simons. His guidance, suggestions, and critique have been invaluable.

The other members of my committee, Dr. Stephen L. Walter and Dr. Robert B. Reed, provided valuable editorial comments, which have greatly improved the quality of the final product.

Martin Hosken was also a great help from the beginning, providing a sounding board for my ideas and pouring over drafts with numerous comments.

May 11, 2001

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AFIL.....Association for Font Information Interchange
CETH.....Center for Electronic Texts in the Humanities
Co.....Company
CSLI.....Center for the Study of Language and Information
DTD.....Document Type Definition
HMM.....Hidden Markov Model
HTML.....Hyper Text Markup Language
IPA.....International Phonetic Alphabet
ISO.....International Organization for Standardization
LACUS.....The Linguistic Association of Canada and the United States
ORL.....Orthographically Relevant Level
PTGC.....Phoneme-To-Grapheme Conversion
SGML.....Standard Generalized Markup Language
SIL.....Summer Institute of Linguistics
SIL-AAB.....Summer Institute of Linguistics Australian Aborigines Branch
TEI.....Text Encoding Initiative
UCS.....Universal Character Set
WSD.....Writing System Declaration
XML.....Extensible Markup Language
XSLT.....XML Stylesheet Language for Transformations

Chapter 1: Introduction

Despite the numerous works actually describing writing systems, the literature conspicuously lacks discussion about the process of describing writing systems. Consequently, many of the writing system descriptions available today lack key information, leaving many questions unanswered.

The descriptions of writing systems have simply not received the attention that other areas of linguistics have enjoyed. For example, Pike (1947) in his book *Phonemics* included two chapters dedicated to the procedure involved in describing sound systems. Nevertheless, the reasons he gave for including this discussion about descriptions apply to writing systems just as they do to sound systems:

- The analysis "will be available to other persons" (page 174)
- "A person can come into a study of a strange language and learn it more rapidly" (page 174)
- "Students of linguistics in general also need to have precise, accurate, and detailed descriptions of the sound systems of all languages. They wish to make general studies of the types of structural relationships which are found around the world, in the hope of discovering various language characteristics of a universal nature which will increase their understanding of basic language types. For this purpose they need

statements about the large number of languages of which there are as yet no technical materials available" (page 174).

- "For the investigator himself it is important that he prepare a written statement of the phonemic data. This helps (1) to clarify his ideas, (2) to check the completeness of his materials, [and] (3) to assemble evidences which help him reach …[his] conclusions…" (page 174)

This lack of a framework and the corresponding lack of complete writing system descriptions leads to a further problem, the problem of multilingual computing where a wide variety of writing systems must be supported on an electronic system.

The advent of the computer, ushering in the current information age, has brought the ability to create, store, and manipulate large amounts of information. Much of this information is textual in nature, representing writing from numerous languages. But each language is different, and each language is written differently. The Internet has made it easy to share information with a global audience. However, with this global audience comes the global diversity of languages and writing systems.

As texts in a diversity of languages are created, shared, and archived electronically, the need to describe the writing systems used by these texts becomes readily apparent. How can one know what is represented by the graphic symbols used in a text? Or how can various computer processes "know" how to treat a string of characters in a little known language?

There is much information that one needs to know to be able to understand a writing system. What are the properties and interrelationships of the characters (the atomic units of writing systems)? What linguistic elements are expressed in graphic form? How are the sounds (or other linguistic elements) expressed in the writing system? What are their names?

Are there special contexts that determine variant graphic forms of the same character? What is the order into which the characters are conventionally arranged?

In addition, the computer must be able to recognize the characters from electronic storage and take in their properties. It must be able to sort character strings. If it is to insert hyphenation points or check the spelling or grammar of a text, it must be able to identify syllables, words, and sentences. If it is to perform searches for abstract patterns, it must be able to determine whether a given character is a member of a given class of characters. In order for a computerized process to intelligently interact with someone concerning a writing system, the machine must share the same knowledge as the user.

For most computerized processes that rely on writing system information, that information is “hard coded” into the application code. This means that the writing system information cannot be changed or extended to add support for a new writing system without changing the code and creating a new version of the program. An example of this is found readily when a language uses the apostrophe character /'/ to represent the glottal stop. In such languages, the apostrophe should be treated as a letter of the same standing as /a/ or /b/ or any other letter. However, when performing a pattern based search in Microsoft Word, /'/ is not included among the letters.

When writing system information is treated as input data, processes that involve writing systems information can be generalized to work with any language. If Microsoft Word retrieved writing system information from a source which the user could modify, the user could add /'/ as a letter and the aforementioned search would behave as expected.

This thesis presents a model for writing system descriptions that captures this type of information and can be used by both humans and computers. It is motivated by theory from

the study of writing systems and informed by what is actually attested in existing writing system descriptions.

This model is designed to be functional, that is, it is designed to be useful. Some linguists have recently begun to question the utility of the products of linguistic research: “To what extent do linguists’ descriptions serve ‘consumers’ in domains beyond the discipline of linguistics?” (Butt 1996:xv–xvi) The intent of this research is to create a model that will serve not only linguists, but extend into a number of complementary domains.

This thesis has a dual audience and a dual approach. First, this work is aimed toward researchers who want to describe the writing systems of the world, providing the groundwork for a repository of electronic writing system descriptions such as would be needed for the proper documentation of archived electronic texts. Second, it is aimed at computer programmers who want to write generalized software that is truly multilingual and extensible.

Chapter 2: Review of the literature

2.1 Introduction

As previously mentioned, the literature lacks any thorough work on the process of describing writing systems. However, literature does exist which provides insights into the problems faced in writing system analysis and description as well as into the requirements of such a system. This chapter surveys the pertinent literature on the study of writing systems, making special note of the insights that may affect the description of writing systems.

People have long been fascinated by the diversity and beauty inherent in the scripts used in writing. Campbell, in the introduction to his *Handbook of Scripts & Alphabets* as a companion volume to his *Compendium of the World's Languages* wrote that "the intrinsic interest of the scripts themselves seems to justify their separate publication" (1997:vii).

Daniels (1996) provides a thorough history of the study of writing systems, beginning with the earliest civilizations when "writing systems themselves were overlooked, or looked right through" (page 5) and tracing the development of the interest in writing system study. In the seventeenth century, journals began to carry reports from "explorers and missionaries about the languages and scripts they were encountering" (page 6). However, it was not until 1883 that Isaac Taylor would publish the "first book on writing from a scientific perspective" (Daniels 1996:6).

The modern study of writing has primarily attempted to address two questions: (1) how did writing and the many scripts of the world develop? and (2) how does writing represent speech?

The first question involves the historical development of writing systems. Most works of this nature provide many specimens of various types of writing and attempt to trace the spread of writing or the lineage of particular systems of writing. Diringer's *The Alphabet* (1968) is perhaps the best-known work of this type.

The second question involves the linguistic study of writing systems. Historically, the primary focus of linguistics has been devoted to spoken language to the neglect of writing and written language. "Writing systems per se … have absorbed the attention of only a very few linguists" (Daniels and Bright 1996:1). However, Sampson rightly observed that the system of writing needs to be described just as any other system of language (1985:13). Despite the dearth of literature on the subject, a number of definitive works on writing systems have been produced by linguists.

Not only has the field of linguistics begun to study writing systems, but other disciplines such as literacy and reading, computer science, and psycholinguistics have also produced works, which include substantial discussion of writing systems. Thus, the study of writing systems has been quite eclectic, receiving attention from numerous perspectives. The following pages provide an overview of the major contribution of that literature.

2.2 Theory of writing systems

The linguistic theory of writing systems began with an attempt to apply emic[1] principles to writing systems and to analyze writing in a similar fashion to phonology or grammar.

More recently, several unified theories of writing systems have been proposed. Informal approaches to explain the general methods by which language is represented in writing have been written by Gelb (1963), Sampson (1985), Coulmas (1989), and DeFrancis (1989). Sproat (2000), on the other hand, offers the only general formal approach to the study of writing systems. These and other contributions to the theoretical literature are introduced in the following sections.

2.2.1 Applying emic principles to writing systems

Much of the early work concerning writing systems attempts to answer the question of what are the significant units of writing. Significant units of writing must be described, while insignificant aspects may be ignored.

Pike (1947) approached the problem of writing systems from the perspective of the formation of new orthographies and thus established the goal of a "one-to-one

---

1 The terms *etic* and *emic* were coined by Kenneth Pike (by removing *phon* from the terms *phonetic* and *phonemic*) to indicate the perceptual distinctions between observed data. Thus, *emic* entities are "seen as 'same' from the perspective of the internal logic of the containing system, as if it were unchanging even when the outside analyst easily perceives that change" (Pike 1982:xii), and *etic* entities are seen as "different" from the perspective of an outside analyst. When these two perspectives are correlated, the etic units can be seen to correspond to the emic units such that the etic units are simply conditioned variants of the emic unit. The etic units are the ones that initially matter to the outside analyst and are generally imperceptible to the users. The emic units matter to the insiders. They may or may not be perceptible to the outside analyst.

correspondence between each phoneme and the symbolization of that phoneme" (1947:208) for the development of practical orthographies.[2] His chapter on that topic is naturally oriented from the perspective of the sound system rather than the writing system. This work and its contribution to phonemic theory has been foundational to the design of many new orthographies[3] since its publication.

For Pike there is one significant unit of writing for each phoneme.[4] Such treatment tends to be appropriate for the design of writing systems and for the description of recently designed writing systems. However, it is not able to deal with the actual state of many older writing systems. It is well known that spoken language tends to change much faster than its written counterpart, and thus, writing systems with long histories tend to require a more sophisticated analysis of the relationship between the sound system and the writing system.

Pulgram and Bazell were some of the first to apply emic principles to the study of writing systems. They attempted to find parallels with other emic units: phonemes and morphemes, respectively. Pulgram (1951) thought that the variations present in writing a single letter were similar to variations present in the speech sounds of a language. He posited that "graphemes," the underlying form of these variants, were similar to phonemes.

Pulgram lists the following characteristics of "graphemes" (1951:15–16):

- The smallest distinctive visual units of an alphabet are its graphemes.

---

2 He did not preclude the use of syllabaries which "have a one-to-one correspondence between each syllable and the symbol representing it" (Pike 1947:208).

3 These are orthographies which are phoneme based, that is, which seek to make as simple a correspondence between the graphs and the phonemes as possible.

4 When the system of writing is based on the syllable, there is one significant unit of writing for each syllable.

- A grapheme is a class of written characters pertaining to one alphabet.
- The *hic et nunc* written realization of a grapheme is a written alphabetic character or graph.
- The number of graphemes in each alphabet must be limited, the number of graphs cannot be.
- By definition, all graphs identifiable as members of one grapheme are its allographs.
- The graphic shape of an allograph is dependent on its producer and on its graphic surroundings.
- Graphs which are not immediately and correctly identifiable as belonging to a certain grapheme when occurring in isolation, may be identified through their meaningful position in a context.
- Alphabets are subject to graphemic change and substitution.
- The number, kind, and distribution of graphemes varies from alphabet to alphabet.

Pulgram was concerned with accounting for variations present in the form of graphic shapes. In doing so, he accounts for forms that vary given a particular context (such as word initial, medial, or final forms). However, he also accounts for variants that are caused by the production process, forms that are analogous with the variation of vocal quality in humans due to differing physiology:

> No matter how a person's handwriting realizes the grapheme of, say, the Latin alphabet, no matter what style or font a printer employs, each *hic et nunc* realization of a grapheme, which may be called a graph, can be recognized as

> belonging to a certain class and therefore deciphered by the reader. All graphs so identifiable are allographs of a given grapheme. (1951:15)

Pulgram's analysis centered on the human ability to recognize graphic forms with numerous variations as alike because of the distinct features they share. This led him to a perspective that is structural, that is, based on the form rather than the meaning that is represented:

> It appears that just as a phoneme may be interpreted as a bundle of simultaneous distinctive features that are obligatory in every realization of the phoneme (which is the reason why there is no freedom of choice for the speaker as to what distinctive features to use, and why a single distinctive feature is not a minimal emic unit), so the grapheme is composed either of the various strokes and loops of letters, or of the patterns that make up the designs of syllabic and ideographic symbols (also with an absence of choice for the writer because of the necessary simultaneity and inclusiveness of all the features, none of which by itself is an emic unit). (1965:212)

Although Pulgram divorces the units of writing from the language they represent, he does specify that the significant units of writing are not the strokes or patterns that make up a symbol. Beyond that, he does not specify how the significant unit of writing is identified.

Bazell (1956) in a similar structural vein likened the "grapheme" to the morpheme, the minimal meaningful unit of grammar. He suggested that letters could be split into the strokes by which they are formed and these strokes should be likened to phonemes:

> By definition the phoneme cannot contain smaller distinctive features unless these are simultaneous. The corresponding graphic unit should equally have no smaller features except such as are spatially superimposed. But letters are normally distinguished from each other by features (dots, curves etc.) located in different positions, these positions themselves being relevant (e.g. b/d). Hence it is, for instance, the bar and loop of *b* and *d*, not these whole letters, that answer to phonemes. (pages 44–45)

Thus, Bazell seems to indicate that the individual strokes involved in writing are significant units of writing. This naturally raises the question of what is meant by a significant unit of writing; Bazell and Pulgram seem to have very different ideas.

The structural view of writing, which Pulgram and Bazell expose, highlights the shapes of graphic forms, independent of the linguistic elements they represent. This viewpoint has come under harsh criticism by Daniels in his paper entitled "Is a Structural Graphemics Possible?" (1992). This eventually sparked debate between Daniels and Herrick (Daniels 1995, Herrick 1995a, 1995b).

Daniels claimed that "the graphemics[5] of a language should be described in terms of the phonology of its language" (1995:426), while Herrick claimed that "the graphemics of a language should be described on its own terms" (1995a:413). Daniels was concerned primarily with how writing represents language while Herrick was concerned primarily with how units of writing can be recognized by a reader.

Ultimately, Daniels disregards a structural analysis as not being useful (1995:430). For him, the fact that writing represents language takes priority over the fact that a person can recognize writing units. Therefore, he favors an analysis that highlights the linguistic relationship:

> The many scholars who accept as a given for some language some inventory of characters of a script—for English, say, the letters of the alphabet—and describe the relations between its sounds and their spellings … are, in my opinion, doing what is necessary in the study of writing systems. (1995:426)

---

5 *Graphemics* is one attempt at a name for the study of writing systems. Another, proposed by Gelb (1963), is *grammatology*, but "no name for this field of study has even become widely accepted" (Daniels and Bright 1996:1).

Although there is still room for debate, the majority of writing system researchers have chosen to describe writing in terms of its relationship to linguistic notions.

Gleason acknowledged that each of these perspectives could operate: "Not only does a writing system have its own structure which can be studied, but there is also a set of conventions of relationship between the writing system and certain structures (commonly phonologic) in an associated spoken language" (1961:409).

While recognizing structural analyses, Hall also favored linguistic ones:

> It is, in theory, possible to analyze graphemes purely in terms of the structure of their visible shapes, identifying their graphic components such as vertical strokes, horizontal strokes, curves, etc.… For our purposes, however, the only useful analysis of a graphemic system is in terms of the symbolization it affords to features of linguistic structure, i.e., in terms of its linguistic meaning. (1964:266)

Thus, the significant units of writing when describing writing systems are those units that correspond to some linguistic structure such as the phoneme. The strokes that make up these significant units are themselves only important in terms of the production of the significant units. This issue of what make up the significant units of writing is discussed further in Section 6.2 and Section 6.3.

2.2.2 Informal contributions

Many of the theoretical works on writing systems are concerned with how writing came about. This history of writing is of no use to the topic at hand. Some of these works did deal with the relationship between writing and the linguistic structures of spoken language and this is very much of use to us. Unfortunately, the contributions in this area are severely

limited. This section introduces the theoretical works on writing systems that are not of a formal nature while highlighting the requirements that they impose on a framework for the description of writing systems.

Gelb (1963) desired that his book, *A Study of Writing*, would "lay a foundation for a full science of writing, yet to be written" (page 23). Daniels, one of Gelb's students, calls this book "the first linguistically sound theoretical study of writing systems" (1996:3). However, in laying the foundation for a science of writing, Gelb failed to chart the course for such study. "Unfortunately, Gelb's attempt to lay the foundation for his 'new science' was a failure, as is now generally recognized" (Harris 1995:1).

However, Gelb did provide a linguistically motivated definition for writing, namely that writing is a "device for expressing linguistic elements by means of visible marks" (page 13). With this claim, that there exists a relationship between linguistic information and graphic information, the stage was set for linguistic description of writing systems; a description whereby these "visible marks" and their relationship to corresponding linguistic elements, such as "phrases, words, syllables, single sounds, and prosodic features" (page 14), could be made plain. However, Gelb was primarily concerned with tracing the development of writing; [6] and he does not provide any insight into the nature of the relationship between writing and language.

Hall, in a short article, attempted to "formulate and codify a comprehensive, unified theory" (1960:13) of writing systems. He believed that the analysis of writing systems "rests, basically, upon the recognition and establishment of significant units of visually perceived

---

6 This probably stems from the fact that he was "a specialist in the earliest stages of the Semitic language Akkadian" (Daniels 1996:6).

form" (1960:13). He demonstrated that distinct types of linguistic information (phonemic, morphophonemic, morphemic, semantic, and etymological) are represented by written symbols and that these various types may be simultaneously represented within the same writing system (1960:14). He also accounted for the fact that written symbols sometimes represent arbitrary information, such as the case where the spelling of a word attempts to convey etymological information that, in fact, is fallacious, such as "the French spelling of *legs* /lɛ/ 'legacy' with *g*, as if derived from *léguer* 'bequeath' instead of its actual source *laisser* 'leave' " (Hall 1960:16). Hall showed that the inventory of the linguistic elements that can be represented by written units is quite diverse. A framework for describing writing systems must be able to accommodate this simultaneous diversity.

Although Hill did not attempt a unified theory of writing systems, he proposed that "writing systems can all be classified in terms of which units are recorded, and how" (1967:93). He also makes an important claim that "all systems leave some of the linguistic structure out of the record, since not all parts of linguistic structure are required for unique identification" (1967:93). This indicates that a complete description of a writing system may not be all that is required for some types of automated process, such as automated text-to-speech processing. Grammatical, lexical, and phonological information may additionally need to be described. A framework for writing system description must be able to handle under-representation.

Herrick provides valuable information about the characteristics of the units of alphabets (1974:10–11):

- Each letter has a name.

- A letter has a pronunciation.
- A letter has a place in the alphabetical order of its language.
- A letter, and every other grapheme, has a certain role in the graphotactics[7] of its language.
- A letter is embodied by marks which have a certain basic shape or a certain few basic shapes.

These properties of letters should be included within the writing system description framework.

Haas suggested that writing relates to language at many different levels but that each writing system has one linguistic element that is the predominant unit represented by graphic forms. "The unlimited number and variety of utterances… [are analyzed] in terms of a limited inventory of recurrent units" (1983:16). These units are then assigned written symbols. "The level of the underlying analysis determines the level of the script: the written symbols represent words or morphemes or syllables or phonemes. It is the choice of one of these that characterizes a script as being derived on a certain level" (1983:16). Thus, although writing may represent a diversity of linguistic elements, one set tends to be represented more thoroughly in a particular writing system. This set should be identifiable in a framework for the description of writing systems.

Lamenting that "books on writing still tend to concentrate more on the physical appearance of scripts than on analysis of the formal relationships between graphic elements" (1985:12), Sampson was the first since Gelb to attempt a major work on writing

---

7 The *graphotactics* are "formulas which state the combinations of graphemes which may occur in that language" (Herrick 1974:10)

systems. Sampson divided the study of writing systems into three fields: typology, history, and psychology (page 15). He, like Gelb, extensively treated the history of the development of writing. Additionally, he sought to determine a taxonomy of writing systems by determining the principles used to reduce a given language to writing (page 15). Although he described aspects of four writing systems in depth, he did not provide any theoretic principles for how writing systems should be described.

Booij demonstrated from examples in Dutch that writing does not simply represent the sounds of speech, but instead, "an orthographical system represents different levels of language structure" (1987:215). He included linguistic units at intermediate derivational levels as those that can correspond to written units. This adds all the intermediate levels into the set of linguistic units that may need to be referenced in a writing system description.

Sgall attempted to "formulate the basic definitions that seem to be necessary for a theoretical description of graphemic systems based on the phonemic principle" (1987:1). He defines terms such as *grapheme*, *protographeme*, *subgrapheme*, *epigrapheme*, and *alphabet* based on the alphabetic systems of Europe. However, his system does not work for the many non-Roman writing systems and thus is not of use to a general framework designed for any language and writing system.

Coulmas (1989) provided a major theoretical work on writing systems. He supplied an important discussion about the relationship between the structures of language and the structures of writing, claiming that writing systems are not only "based on analytical perceptions", but that they help to form these analytical perceptions and the "resulting conceptualizations of the structural units of language" (page 40). He also claimed that terms such as *sentence*, *word*, and possibly even *phoneme*, although used to describe spoken

language, find their origin not in the structure of spoken language but of written language (pages 39–40). Thus, writing system descriptions should describe these larger units that have their origin in the structure of written language. Coulmas supplies limited descriptions of writing systems while tracing the development of writing. Like the others, he provides no directions about how to describe a writing system other than to acknowledge that "writing systems operate on different levels and emphasize different units of language" (1989:270).

DeFrancis (1989) argued convincingly against the prevailing view that many different linguistic levels may be predominantly represented by a writing system. Instead, he claimed that "the only writing systems that have ever been created, and … ever can be created, are those that represent language on the syllabic or phonemic levels" (page 229). While acknowledging that writing may represent other linguistic features, he claimed that other linguistic levels would never constitute the predominant system of representation. DeFrancis suggested the "Duality Principle," which claims that writing systems may convey meaning by either using "symbols which represent sounds and function as surrogates of speech," or by using "symbols that add nonphonetic information" (page 49). These two systems are applied in proportion to each other such that "the poorer a writing system is in phonetic representation, the more it compensates… by greater use of nonphonetic devices" (page 51).

DeFrancis suggested that "the key question to ask about writing systems is not the ambiguous one of what they 'represent' but the more precise query as to what are the basic units…that make the systems work" (page 56). Once these basic units are known, the corresponding question of what they represent can be addressed and explained in terms of the basic units:

> In an alphabetic system of writing, since what is represented is phonemes, we should start by asking how many phonemes there are, and what symbols are used to represent them. In the case of English, we know that there are about 40 phonemes of various kinds (consonants, vowels, stress, pitch, and others). We also know that there are 26 letters and a few other symbols available to represent those 40 phonemes…in our writing system. (pages 230–231)

> In a syllabic system of writing, since what is represented is syllables, we should start by asking how many syllables there are, and what symbols are used to represent them. In the case of Japanese, the answer is fairly clear, despite some differences of opinion among specialists. There are 105 or 113 syllables and 46 syllabic symbols to represent them. (page 231)

DeFrancis lamented "the dearth of truly illuminating discussion of the relationship between writing systems and the languages they represent" but cites Huttar's description of the Njuka language (Huttar 1987) as a "satisfying bit of scholarship" (page 237). Indeed, in this short article, Huttar has concisely described the relationships between the Njuka phonology and its representation in the Afaka syllabary, while raising linguistic issues of the adequacy of the under-representation of the script.

Harris (1995) attempts to provide the theoretical framework that was lacking in Gelb's *A Study of Writing*. He introduces questions about many aspects of writing, written communication, and writing systems, that such a framework should answer. However many of these questions are of a higher-level theoretical perspective than will be dealt with in this framework, e.g. the arrangement of text on signs.

The study of writing systems has also been addressed from the perspective of creating new orthographies for languages that have no written form. These works attempt to determine or suggest how to best represent a spoken language in written form. As previously mentioned, Pike (1947) addresses how spoken language can be reduced to writing. Smalley edited a useful volume dedicated to this perspective, which presents insights into many of the

problems encountered (1964). These are not problems of description, but rather problems of analysis and design, and thus, have not been found to be useful to this study.

These contributions to writing system theory aid the study of writing system descriptions by alerting us to the vast diversity of the linguistic units that may correspond to some unit of writing. This correspondence may be only partial. These linguistic units are discussed further in Section 6.1.

2.2.3 Formal contributions

The theoretical works on writing systems which are of a formal nature tend to be much more concerned with a means of indicating the relationships between the graphic forms and the underlying linguistic units they represent.

Nunberg (1990), in the only work of its kind, analyzed English punctuation and provided rules to account for his observations. He suggested that punctuation (as well as other graphical features) is a text-category indicator:

> The term *punctuation* is generally used to refer to a category defined in partially graphic terms: a set of non-alphanumeric characters that are used to provide information about structural relations among elements of a text, including commas, semicolons, colons, periods, parenthesis, quotation marks and so forth. From the point of view of function, however, punctuation must be considered together with a variety of other graphical features of the text, including font- and face-alternations, capitalization, indentation and spacing, all of which can be used to the same sorts of purposes. … I will talk about all of these graphical devices as instances of *text-category indicators* of written language. (page 17)

Thus, for Nunberg, text category indicators delimit pieces of text as being distinct from the surrounding text in some way. Quotation marks delimit text that is spoken, thought,

or in some way referred to. Colons, semicolons, and commas delimit clauses at different levels.

He also distinguishes between two grammars of writing: the lexical grammar, which corresponds to the spoken grammar, and the text grammar, which indicates how the text is combined to form larger categories such as paragraphs. He suggests certain text categories on which the text grammar can operate as well as the functions of these text categories. Since the design of the framework of writing system descriptions approaches the description from a general perspective, these categories will not be enforced but would provide a useful starting point for many descriptions.

Sproat (2000) offers a formal computational theory of writing systems. He attempts to answer the following questions (page xvii):

- What linguistic elements do written symbols encode?
- Do writing systems differ in the abstractness of the linguistic representation encoded by orthography, and if so, how?
- What are the formal constraints on the mapping between linguistic representation and writing?

Although some of these issues had been previously addressed, this had usually been "in an informal fashion" (page xvii).

Sproat models the relationship between written and linguistic forms (page 6) by viewing this relationship as one where "particular (sets of) linguistic elements *license* the occurrence of (sets of) orthographic elements" (page 9). He creates the term *Orthographically Relevant Level* (ORL), defining it as "that linguistic level of representation

at which [regular correspondences between linguistic elements and their orthographic expression] are most succinctly stated" (page 10). Both DeFrancis (1989) and Haas (1983) would probably have used this term had it been available to them.

He then makes two central claims: (1) a regular relation maps the ORL to the spelling, and (2) the ORL is "consistent across the entire vocabulary of the language" (page 19). Following Nunn (1998), he assumes that the regular mapping from the ORL to the spelling consists of "a set of graphic encoding rules and a set of autonomous spelling rules" (page 14). Thus, rules indicate the relationships between the linguistic units and the writing units. Later chapters, will draw extensively from Sproat's work in the formal descriptions.

2.3 Descriptions of writing systems

Numerous writing systems have been described to varying levels of completeness. Some of these are published; others remain unpublished. SIL International has writing system descriptions for many of the languages that have been studied by SIL linguists. However, these usually remain in branch archives around the world and are not publicly available. This thesis attempts to provide a format that could help to make these more widely available. Some published descriptions feature a particular language, while others are collections. These descriptions do not typically describe the method or process that was used for description. When they do, the statement is usually terse.

The following sections briefly describe the actual instances of writing system descriptions that were used as exemplars for the design of the framework for writing system

description. These represent the actual information about writing systems which scholars have thought useful to make available.

2.3.1 Collections of writing system descriptions

Tucker (1971) lists symbols that are used by various systems for writing African languages, but he does not provide specific information about how a set of symbols is used in a particular language.

Nakanishi wrote his book, *Writing Systems of the World: Alphabets, Syllabaries, Pictograms (Sekai no Moji)*, to provide a "relatively simple survey of scripts for collectors and general readers rather than specialists" (1980:7). He includes 29 languages, the basis for selection being the existence of a daily newspaper. He provides the following statement about his methodology:

> The content of each section varies according to the script discussed, but generally it includes: a chart of the symbols or letters used, with pronunciation readings; a list of the most important signs and diacritical marks; a discussion of the mechanics of script writing—style of letter, shape transformation, tone and vowel indication, joining of letters, etc.; notes on reading and script direction; and the figures used for numerals. In addition, a recent newspaper sample is given as an example of the script in daily use. (page 9)

Thus, Nakanishi found an inventory of the symbols, their pronunciations (or mappings to linguistic units), and a discussion of the mechanics of the script as well as notes on reading to be important in writing system descriptions. This is exactly the type of information that writing system descriptions seek to capture.

The *Alphabets of Africa*, edited by Hartell, endeavors to "make accessible to a wider public a sample of some 200 alphabets of the languages of Africa, especially those alphabets

developed in the last 20 years" (1993:v). Hartell's summary of the situation of the writing systems of African languages pertains to many of the world's languages, "The documentation of those systems… is largely unknown and hard to obtain outside of their immediate area of use" (1993:v). Besides limiting itself to only 200 of the more than 500 African languages that have writing systems, this collection is limited in two notable ways: "it does not deal with all the countries, nor does it treat each writing system fully" (1993:vii). However, the work is easy to use due to the consistency of the samples. It is a perfect example of the utility of even incomplete writing system descriptions, for "the exact purpose of the book is to present readers with the barest minimum of information on the speech sounds of these language samples and the graphemes used for representing them" (1993:viii). As for the process that was used, the following statements are given:

> As much as possible, we have tried to present the alphabet systems in this volume in a consistent format. Each alphabet system presented consists of four parts: (1) a phonemic inventory with its orthographic representations, (2) examples using the alphabet characters in words from the language, (3) a linear list of the alphabet in alphabetical order (characters are called graphemes and include simple characters as well as digraphs), and at the end of each country chapter (4) selected bibliographic information. (page 21)

> The phonemic inventory of each language is presented in three columns, *roughly* dividing the consonants into plosives, nasals, and continuants, and dividing the vowels into front, central, and back. Below the vowels in the chart, any features such as length, nasalization, labialization, etc. are listed. Finally, tone was indicated where it occurs, though sometimes it is not included in the actual orthography in use. (page 21)

The purpose of the *Alphabets of Africa*, namely comparison of many African writing systems, severely limited the detail present in these writing system descriptions. Yet, it also

contains some important information to be included in a writing system description: bibliographic information, examples of words, and an alphabetical order.

The landmark book edited by Daniels and Bright, *The World's Writing Systems*, describes many of the major languages of the world. They claim that their work is unique in that it goes beyond the typical information included in surveys of the world's languages or scripts to "include information about how the scripts represent the languages" (1996:xxxv). The theoretical basis for their description is limited to the following statement:

> Each contributor was asked to provide a historical sketch and the table of signforms in their standard order and their variations, but the bulk of their work was to be a description of *how the script actually works*— how the sounds of a language are represented in writing, along with a brief text in the language(s) the script is used for. (1996:xxxv)

The text is presented not only in the script but also in the standard transliteration, and in transcription using the International Phonetic Alphabet. In addition, literal and free translations for the text are given.

Daniels and Bright do not consider a writing system to be unique to a particular language. Thus, they make no distinction between the terms *writing system* and *script*. A close look at *The World's Writing Systems* indicates that it is a treatment of the world's scripts with practical application made to a single language that uses the given script. They do mention what they believe is required in order to describe a writing system linguistically, namely, "the characters of each writing system must be inventoried, and their use and interpretation ascertained" (1996:1). Although this is quite a broad set of requirements for a writing system description, in practice, the writing system descriptions present in this volume

include many important elements including an example text in the orthography and the description of how the characters correspond to linguistic elements.

Kannaiyan presents, in chart form, a comparison of Roman, Brahmi, Tamil Grantha, Tamil, Telugu, Canarese, Malayalam, Hindi, Kashmiri, Punjabi, Gujarathi, Assamese, Bengali, Oriya, Urdu, Sinhalese, and Burmese scripts (1960). For purposes of comparison, this format is interesting but it lacks the precision necessary for a scholarly comparative work.

Campbell's *Handbook of Scripts & Alphabets* (1997) contains a set of short but helpful descriptions for 43 languages. These are primarily concerned with the inventories of units with some prose to explain some interesting features of the particular writing system.

Several compilations of descriptions of the writing systems of Nigerian languages have been produced (Williamson 1983, Banjo 1985, Armstrong 1986)—a testimony to the "greater interest in Nigerian languages currently shown by Government, educators, and the general public" (Williamson 1983:v). These are some of the best exemplars of editorial consistency in the writing system descriptions.

2.3.2 Writing system descriptions for individual languages

Weir provided an account of ongoing research, which attempted to analyze the spelling-to-sound relationships for English "in order to discover not only the basic patterns involved, but also, the levels on which these patterns operate" (1967:173). The model that she and her colleagues created drew on three linguistic levels: "grapheme, morphophoneme, and phoneme" (1967:176). Because they believed that "English orthography is subject to morphophonemic rules" (1967:177), simple associations between phonemes and graphemes were rejected in favor of "hierarchically ordered rules" that would "map the single morpheme

graphemic word onto the morphophonemic level, and then, through the application of further appropriate rules, onto the phonemic level" (page 173). Rules are used quite extensively within the literature to describe the correspondences between writing units and linguistic units.

Richard Venezky worked with Ruth Weir until her death in 1965. He wrote a computer program to determine spelling-to-sound correspondences for his master's thesis (1962) and wrote his Ph.D. thesis (1965) on the spelling and sound correspondences in English. He provides summaries of his work in two articles (1967a, 1967b). His book, *The Structure of English Orthography* (1970), not only provides a detailed discussion of English orthography, but also provides a helpful discussion of orthographic analysis that includes determining the spelling units (1970:34), the graphic alternations (1970:37), and the relationships of graphic units to sound (1970:39).

Derwing, Priestly and Rochet have also attempted to formulate rules to describe the relationships between spelling and sound for English, French, and Russian. Indeed, they claim to have "achieved a level of *systematicity and homogeneity of description* … [as well as] a level of *comprehensiveness of coverage* and *attention to details* which were lacking in many previous works of this nature" (1987:32). They propose three different types of rules: spelling, pronunciation, and phonetic detail rules.

Wijk presents rules for pronouncing English, which he identifies as the most puzzling piece of the English writing system for "it is a generally recognized fact that the English language presents far greater difficulties with regard to its pronunciation than any other European language" (1966:7).

Some descriptions of writing systems are really teaching methods with exercises in learning the writing systems, e.g. for Burmese (Roop 1997). In addition to helpful descriptions of the writing system, these provide information about how to "draw" the characters correctly.

A number of other writing system descriptions were used as exemplars but had no special components in their presentations that would be useful to mention and are included here as documentation of the process. Sandefur (1984) presents an extensive description of Australian Kriol from the theoretical perspective of the development of new writing systems. Bender, Head and Cowley provide a description of the Ethiopian writing system (1976). Ronald Schaefer describes the writing system of Emai, a language of Nigeria (1987). Bruce Grant describes the Korean writing system (1982). Tadadjeu and Sadembouo edited a description of general principles to be used by all Cameroonian writing systems (1984).

2.4 Requirements for descriptions of writing systems

A number of SIL International's branches have requirements for writing system descriptions. However, these have not been published in any form. For this thesis, the branches of Burkina Faso, Cameroon, Cote d'Ivoire/Mali, Eastern Congo, Ghana, Indonesia, Kenya, Mainland Southeast Asia, Mexico, and the Philippines provided copies of their orthography manuals which state their requirements and which I referenced as part of my research.

Most of these provided a worksheet for listing the phonemes (along with the allophonic variations) and the symbol used to represent that phoneme. The punctuation marks and their uses may be discussed. Usually, an example of a word that uses each symbol

may be indicated. In addition, they may have a place to discuss any issues, which have arisen, or motivation for the particular selection of a given symbol. They may provide a place to make comparisons to other languages in the region that may have sociolinguistic influence. Some included a place for discussion about how loan words are handled in the orthography, how words are broken, and how morphophonemic issues are handled. Some included a section on alphabetical order. Many asked for a sample text in the orthography.

Most devoted the majority of their attention to the process for devising and standardizing an orthography rather than how to go about describing the orthography.

2.5 Computational models

The need for computerized models of text, which include information about the writing system, has been known for some time.

Becker (1984) challenged the field of computer science to extend the model of text to encompass any language: "the computer should deal with a universal notion of 'text' broad enough to include any of the world's living languages in any combination" (page 96). He split this problem into three subtasks:

> There must be a way for text to be represented in the memory of a computer; there must be a way for text to be typed at the keyboard of a computer; there must be a way to present text to the typist. I shall refer to these realms as encoding, typing, and rendering. By rendering, I mean both the display of text on the screen of a computer and the printing of text on paper. (1984:96)

Simons (1989) repeated Becker's challenge, "we need computers, operating systems, and programs that can potentially work in any language and can simultaneously work with many languages at the same time" (page 538). In addition, he added the requirement that

users should be able to extend the computer's "knowledge" of scripts, faulting Apple's Script Manager for not providing "a general mechanism for defining new scripts" (page 538).

Simons then argues that a general implementation of writing systems is computationally feasible. Simons' model follows in distinguishing between the form (graph) and function (character) of writing system units.

> The proper modeling of writing systems thus requires a two-level system. At the functional level are information units called *characters*. At the formal level are elements called *graphs*. The higher-level units called characters are realized by the lower-level units called graphs. (1989:541)

His model allows "users to describe new writing systems by expressing the mapping from characters to graphs as rewrite rules" (1989:545). Simons then shows that even complex rendering problems can be solved by these rewrite rules and converted into instructions that are quite simple for a computer to handle.

Hosken et al. (2000) defines a language for writing rewrite rules to express the relationship between characters and glyphs as suggested by Simons. This is used by the Graphite system under development by SIL International.

Simons and Thomson (1998) also dealt with the issue of multilingual computing, and described problems associated with multilingual computing along with the solutions they have implemented in a system called CELLAR. In addition, they provided a model for defining characters and implementing sorting for a variety of languages.

The Text Encoding Initiative (TEI), a scholarly initiative to provide a means to interchange documents in the humanities, provided a formal model for describing writing systems:

> The writing system declaration or WSD is an auxiliary document which provides information on the methods used to transcribe portions of text in a particular language and script. We use the term *writing system* to mean a given method of representing a particular language, in a particular script or alphabet; the WSD specifies one method of representing a given writing system in electronic form. A single WSD thus links three distinct objects:
>
> - the language in question
> - the writing system (script, alphabet, syllabary) used to write the language
> - the coded character set, entity names, or transliteration scheme used to represent the graphic characters of the writing system
>
> Different natural languages thus have different writing system declarations, even if they use the same script. Different methods used to write the same language (e.g. Cyrillic or Latin encoding of Serbo-Croatian), and different methods of representing the same script in electronic form (e.g. different coded character sets such as ASCII or EBCDIC, or different transliteration schemes) similarly must use different writing system declarations. (Sperberg-McQueen and Burnard 1999:563)

Birnbaum, Cournane and Flynn suggest two uses of the WSD (1999:27):

> - to document transcription methods in a human-readable form, so that those who need to render, transform, or otherwise process the document will understand the meaning of the encoded information, and
> - to support automated access to such representational information as the character codes and glyph identifiers associated with the characters … [in textual] content.

Not only are these laudable goals, but the authors argue that they are demonstrable as well. However, WSDs do not seem to have been embraced by many users of TEI. “Despite the indisputable documentation and processing value of a formal writing system description, the WSD appears to be one of the least-used features of the TEI guidelines” (Birnbaum, Cournane and Flynn 1999:50).

In addition, TEI's WSD does not provide a means to describe the relationships between graphic units or between linguistic elements and their corresponding graphic units. Only a limited set of classes that can be applied to any character.

The Unicode standard assigned limited semantics to character codes. Its primary scope is in the area of encoding. The Unicode standard has done much to make multilingual computing possible. The standard seeks to provide a "uniform method of character identification which would be more efficient and flexible than previous encoding systems" (Unicode Consortium 2000:3). However, Unicode does not fully capture all the information that is necessary to process text in a writing system. For example, sorting, locating text element boundaries, and even rendering are left up to implementations. In addition, Unicode provides a block of character codes called the "Private Use Area" that has been left undesignated to be used by those who need to exchange information that cannot be encoded using other definitions. Any character that is in the Private Use Area needs to be fully described.

The World Wide Web Consortium has produced a draft of a character model for the World Wide Web (Dürst and Yergeau 1999). It provides a flexible understanding of the notion *character*, which is both language and user dependent:

> Japanese Hiragana/Katakana are syllabaries, not phonemic alphabets. A character in these scripts is therefore not a phoneme, but a syllable. Korean Hangul combines symbols for phonemes into square syllabic blocks. Depending on the user and the application, both the individual symbols as well as the syllabic clusters can be called characters. Indic scripts use semi-regular or irregular ways to combine consonants and vowels into clusters. Depending on the user and the application, both individual consonants and vowels as well as consonant clusters or consonant-vowel clusters can be seen as characters. (1999:§3.1 Characters as seen by Humans)

Additionally, this document points out the fact that numerous many-to-many relationships exist in a computerized model of text.

Linguists have begun to realize the flexibility that is demanded by the study of language and to demand this flexibility of the computer software they use. Antworth and Valentine have described the requirements that a linguist brings in regards to software. Included in these requirements are the ability for a user to define alphabets and sorting sequences (1998). Hockey requires software to allow the user "to define the make-up of a word and the alphabetical sequence used for sorting words" (1998:119).

2.6 Summary

Although many descriptions of writing systems have been written, no single reference describes the components that should be included in writing system descriptions. Although the phonological literature is replete with approaches to describing the phonology of a language, the comparable privation in the writing system literature is stark. Actual descriptions, along with the requirements from SIL International's branches, form the best indications. The computational literature has dealt with subsets of the problem and acknowledged it, and even created a partial solution. The next chapters now build on these foundations as they develop a formal model for the description of writing systems.

Chapter 3: The problem

3.1 Rationale for the study

There are two primary reasons why this study is important. First, writing systems need to be described. Such descriptions enable newcomers to become acquainted with the writing system. The implicit knowledge of the inner workings, relationships and mappings must be made explicit and made available to those who are interested in understanding how writing systems work. Second, writing systems need to be formally described in such a way that computer programs can have access to their information. This allows computer programs to adapt their functionality to whatever writing system is in use.

The advent of computer technology has created an altogether new medium for writing: digital writing, where writing is retained not on paper or clay tablets but in a series of invisible electronic bits and bytes. The fact that computers themselves are able to store writing means that the relationship between graphic units and computational units must be described. There is enough overlap between the information about writing systems that is required by computers and that required by people to merit electronic writing system descriptions that can address the concerns of both parties.

When computer programs use electronic writing system descriptions as the source for information about the writing system, computational tasks which are writing system dependent can be handled in a general, efficient way for any language that has an electronic writing system description. Some of these tasks include breaking text strings into units such

as words or sentences, hyphenation, patterned searches, and sorting. Since the electronic writing system descriptions exist separate from the computer programs that use them, one description can be shared by many programs and new descriptions can be created without modifying the program's code.

This study hopes to encourage the description of writing systems and the dissemination of the knowledge contained therein. For those linguists involved in the creation of writing systems for as yet unwritten languages, the availability of descriptions may provide a resource for comparison which may serve as a guide for possible solutions to difficult problems they face while creating a writing system. This study contributes toward the larger goals of documenting and describing linguistic information for languages around the world.

In the past years, the Internet has become a source of information for people all over the world. The Internet is a natural method for widely disseminating copies of writing system descriptions. Standardized writing system descriptions can be gathered in a repository and made available for linguistic research or for people who are simply interested to see the writing system of a particular language.

Additionally, writing system descriptions aid in the documentation of archived materials. A writing system description attached to archived material may allow future generations to read documented material that they otherwise would have to decipher for themselves.

3.2 Theoretical framework

The theoretical framework for electronic writing system description that underlies this work is an eclectic one, drawing from a theory of writing systems and computational processes, as well as a broad base of linguistic theories from phonetics, to phonology, to syntax and semantics.

3.3 Statement of the problem to be investigated

The purpose of this study is to design a method for describing the writing systems of the world, which can be used as an information source for computers and people alike. This study specifically addresses the concerns of two audiences: (1) linguists who are interested in learning about a particular writing system or in describing one,[8] and (2) computer programmers who want to make general applications that can work with the writing systems of many different languages.

3.4 Elements to be investigated

This thesis seeks to address the following questions in relation to writing system descriptions:

- What is the relevant information that should be included?
- How should this information be arranged?

---

8 A larger audience, but with fewer requirements, are the citizens of the world who want to know how to interpret what they see in a text published on the Internet.

3.5 Limitations of the study

With over 2000 written languages, each with its own writing system, the attempt to provide a general framework for the description of writing systems seems lofty indeed. However, the present study of writing system descriptions has been limited in a few ways:

1. Only those writing systems that represent natural languages are supported. Thus, mathematical and musical notations fall outside this scope. It may be that the proposed framework could be extended to encompass these domains as well, but that will be left for further research.
2. The descriptions are primarily concerned with linguistic and data processing motivations. Thus, instruction concerning the proper method for writing a script (including the path of a stroke and the order in which these strokes are drawn) is outside the scope of this work. However, provision is made for indicating the prototypical form of a graphic unit.
3. Only a small cross-section of the writing systems of the world will be tested using the proposed method. Although the intent of this research is to design a method for describing writing systems that is general enough to handle the wide variation attested in the world's writing systems, to completely validate that such a method has indeed been designed and can accurately handle any of the world's writing systems would involve the task of describing every writing system in the world. However, Simons has demonstrated that the prominent diversity of the world's writing systems are governed by simple processes, considering that "while there is tremendous diversity in the graphic symbols …, there is not very much

conceptual diversity" (1989:539) in the writing systems of the world. Thus, if we determine the conceptual processes that operate and provide a means of describing writing systems in terms of these similarities, the wide variation should be able to be handled.

4. The data for the examples of electronic writing system descriptions will be limited to previously published (non-electronic) descriptions. Although the published descriptions may not be as thorough as will be possible in the electronic form, even the smallest amount of description is worthwhile and should be supported and encouraged.
5. Because these descriptions concern writing systems, this study has assumed that the other linguistic aspects which correlate with writing systems have been described elsewhere and may be referenced electronically.
6. Writing systems are, in a way, a type of grammar for writing, indicating how the graphic patterns used to represent speech do so. However, for many writing systems, a complete analysis requires a lexical component to handle exceptions to the more general rule. Writing system descriptions cover the information that can be handled by rule and is not present in a lexicon. In order to account for all the processes involved in writing or reading a text, the information present in the writing system description may need to be taken in conjunction with information that is present in a lexicon. No attempt is made to formally integrate the two.

7. This study is concerned with synchronic description and not diachronic. Thus, this system enables the description of the writing system of a particular language at a particular time and is not able to trace the historical development of writing systems.

This framework is general rather than specific. The attempt here is to make the framework broad enough to encompass all writing system descriptions. A more specific framework may be possible when there is more extensive and complete data available for study. Hopefully, this project will encourage such work.

3.6 Definition of terms

The term *writing* has been most aptly defined as "a system of more or less permanent marks used to represent an utterance in such a way that it can be recovered more or less exactly without the intervention of the utterer" (Daniels 1996:3).

Due to the general nature of this work, it has been necessary to create the cover terms *linguistic element*, *graphic unit*, and *computational unit* to refer to entities which would usually be referred to by a more specific term.

A *linguistic element* is "any unit of language." Linguistic elements include linguistically etic and emic units. Thus, a phone, a phoneme, a morpheme, a syllable, a word, a sentence or even a discourse are considered linguistic elements. Also included are the intermediate units that may exist in an analysis of language as in Booij's treatment of Dutch spelling where "intermediate levels [between underlying forms and phonetic information] can be represented in Dutch spelling" (1987:216).

A *graphic unit* is simply "any unit of writing." Thus, strokes, radicals, letters, characters, graphic syllables, and graphic words can be graphic units. Typically, the lowest level of a hierarchy of written structure that corresponds to a linguistic element is referred to by the terms *character* or *graph* (Sampson 1985:22). Daniels and Bright defined the term *character* to be a "general term for any self-contained element of a writing system" (1996:xl). Yet, a *graph* is a unit of writing irrespective to its function in the system.

The term *grapheme* will be used "in the manner of many commentators on writing systems, to denote the minimal functional distinctive unit of any writing system" (Henderson 1984:15 (note)), despite Daniels' insistence that the term *grapheme* "ought to be jettisoned" since "the emic system should not be expected to apply to writing" (Daniels 1992:534).

For the purpose of this thesis, a *computational unit* is "a number used by a computer to identify a graphic unit." A *code point* is "a computational unit which identifies a character." An *encoding* is "a computerized representation of writing" and defines a set of characters that can be represented as well as their associated code points. Thus, in the ASCII encoding, the letter ⟨a⟩ is represented by the code point *97*. A *glyph* is "a particular rendering pattern used by the computer to render a graphic unit."

The terms *writing system*, *script* and *orthography* can be easily misunderstood as they are often used interchangeably in ordinary conversation. However, these terms do have technical definitions that allow us to distinguish between them. Sproat aptly defined a *script* as "a set of distinct marks conventionally used to represent the written form of one or more languages" (2000:25). Thus, a script is not necessarily tied to any one particular language. For example, the Roman or Latin script is used by English, French, and many others, but the Hangul script is used only by Korean.

A *writing system* is a "set of conventions used to represent a language in writing". Writing systems tie linguistic expressions to written forms. The writing system chooses from the set of distinct marks available in the particular script and thus may not incorporate every symbol that the script makes available. According to Mountford (1996), writing systems can be classified into five "functional kinds" (page 627): orthographies, stenographies, cryptographies, pedographies, and technographies. Orthographies are the "ordinary systems of writing" (page 629), and thus often the term *orthography* can be legitimately substituted by *writing system* without any change in meaning. Orthographies include systems for spelling as well as many other resources which the other kinds of writing systems may not require, for example, systems of punctuation and systems of numeric symbols (page 630). Orthographies always relate to one particular language. Every written language has either a standard or a conventionalized orthography. The International Phonetic Alphabet is an example of a writing system that cannot be called an orthography since it does not relate to only one particular language. Instead, it is a technography, a tool "designed and used by linguists engaged in linguistic activity" (page 628).[9]

Since this method of description can be used by any of the five types of writing systems, the term *writing system* is favored as the object of these descriptions.

---

9 Stenographies are systems of shorthand for quickly taking dictation. Cryptographies are systematic manipulations of the orthography to create a coded message which makes reading the message difficult unless one knows the "key". Pedographies are simplified systems that are used for instruction until the students can graduate to the complete system.

3.7 Summary

This thesis provides a framework for describing writing systems. It is a formal framework so a computer can gain access to the information. It is a complete framework, so that every writing system of the world can potentially make use of it. This framework should be useful to linguists and to computer programmers alike.

Chapter 4: Background information

In order to understand some of the material in this thesis better, a limited understanding of how computers have historically treated text is needed. This chapter also introduces the Extensible Markup Language (XML), the formal notation system of the framework that will be used for describing writing systems.

4.1 How a computer treats text

Becker (1984) summarizes the tasks that computers are called upon to perform for text processing:

> There must be a way for text to be represented in the memory of a computer; there must be a way for text to be typed at the keyboard of a computer; there must be a way to present text to the typist. I shall refer to these realms as encoding, typing, and rendering. By rendering, I mean both the display of text on the screen of a computer and the printing of text on paper. (1984:96)

Computers cannot act on text directly, for computers are designed as number crunchers, not text crunchers. However, when text is represented as numbers, computers can become text crunchers. Thus, for each of these tasks, the units that the computer operates on are "governed by a single, basic fact: the computer can store only numbers" (Becker 1984:96).

Computers owe much of their modeling of text to the typewriter. With the typewriter, a person pressed a key and the form identified by that key appeared on the paper. When a different form was needed, a different key needed to be typed. This system was then modeled by computers. "In the early days of computing, … character generation was hard-wired into

video terminals and molded directly into print chains and type fingers" (Simons and Thomson 1998:203). Thus, in order for a character to be displayed to the user, a number would be sent to the video terminal, and the video terminal would then display the particular character represented by that number. The same process was used for printing. When a user would press a key on the keyboard, a number would be sent to the computer to represent which character had been typed, and the computer would use a number to store the character. For simplicity, the numbers used to represent a particular character were the same for each process whether encoding, typing, or rendering. Thus, when the letter ⟨a⟩ was typed on the keyboard, the number *97* was sent to the computer. When the computer stored the letter ⟨a⟩, it stored it as the number *97*, and when the letter ⟨a⟩ needed to be rendered, the number *97* was sent to the video terminal or the printer. These mappings are represented formally in Figure 1.

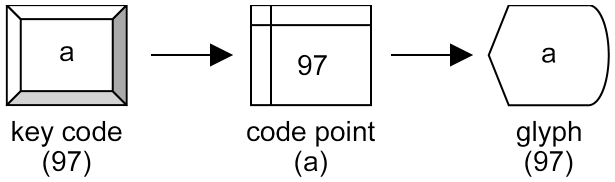

Figure 1 Simple computational unit mappings

It is for this reason that computers have treated characters primarily as unique graphic forms.

> Because most computer display terminals have a built-in one-to-one mapping from character codes to displayed forms, computer users have been forced to encode information at the level of form. That is, to get the correct graphic forms in the correct contexts it has been necessary to store distinct character codes for the distinctive forms. (Simons 1989:541)

> The special-character approach encodes information in terms of its visual form. It says that if two characters look the same, they should be represented by the same character code, and conversely, if they look different, they should have different codes. (Simons 1998:12)

However, as computers became more sophisticated and the need to represent text in languages which use non-Latin-based scripts became more acute, the computer's treatment of text has changed. The constraint of a one-to-one correspondence has been removed. A computer still treats a typed character as a number and stored characters as numbers and even the instructions to render characters as numbers. However, there has been a decoupling of these such that the relationship between typing, encoding, and rendering is more complex and more powerful. For instance, Figure 2 illustrates a mapping where the letter ⟨á⟩ is not typed with a single key or even stored as a single code point. Instead, one types the ⟨a⟩ key followed by the ⟨/⟩ key to represent the letter ⟨á⟩. This is not stored as ⟨a/⟩, or ⟨á⟩ but rather it is stored as the code point for the letter ⟨a⟩ followed by the code point for the acute accent (⟨´⟩). This in turn, maps to a single glyph which positions the acute accent above the letter ⟨a⟩ to form ⟨á⟩.

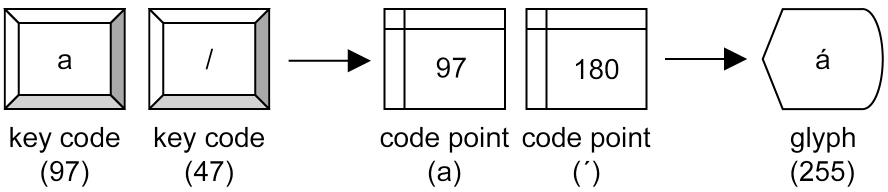

Figure 2 Complex computational unit mapping

This example was somewhat arbitrary and could just as easily have had more than two glyphs, a single code point, and a single key code.

4.2 XML in a nutshell

In order to formalize the description of writing systems, we need to have a formal syntax for representing this information. The Extensible Markup Language (XML) is a popular representation language which is both declarative and formal. It is easily read and understood, being a text-based format. XML provides features that allow documents to be validated against a standard. By modeling writing system descriptions using XML, we gain a formal model that assigns meaning to components that can be read by a computer, ensures consistent presentation for all descriptions, and guides researchers so that they are able to produce complete descriptions.

XML provides a way to segment a text into parts and to identify these parts by name. For example, the title of this chapter is *Background Information*. Because this stretch of text can be considered a single unit that designates the name of the chapter, we may enclose this text with the label *chapter-title* as shown in Figure 3.

```
<chapter-title>Background Information</chapter-title>
```

Figure 3 An XML element delimiting a span of text

A span of text so designated is called an element in XML. It has a name (chapter-title) and contents (the text "Background Information"). An element is delimited by two tags, an open element tag <*chapter-title*> and an end element tag <*/chapter-title*>. Each tag contains the name of the element, delimited by the less-than symbol (<) and the greater-than symbol (>). A slash (/) indicates that the tag is an end tag.

Elements can occur within other elements. Thus, Figure 3 could be treated as in Figure 4, where *title* is contained within *chapter*.[10]

```
<chapter>
<title>Background Information</title>
…
</chapter>
```

Figure 4 An XML element hierarchy

When elements nest, a hierarchy of elements is created. When elements of the same type nest, the tags are properly nested so that the first end tag terminates the most recently introduced start tag. In Figure 5 the first occurrence of <*/section*> closes the second occurrence of <*section*> and the second occurrence of <*/section*> closes the first occurrence of <*section*>. The outermost *section* includes the innermost section.

```
<section>
  <section>This is the innermost section.</section>
</section>
```

Figure 5 Nested XML elements of the same type

Elements may contain any combination of text and elements as illustrated in Figure 6.

```
<example>
  This is a more
  <change>
    <from>complicated</from>
    <to>sophisticated</to>
  </change>
  example.
</example>
```

Figure 6 An XML element with mixed content

10 Other elements such as sections or paragraphs would also normally be included in the content of a chapter but have here been omitted for the sake of simplicity.

The *example* element contains both text and another element called *change*. The *change* element contains two elements, *from* and *to*, each with its own text content.

Since the name of an element indicates its type, elements are usually named according to the function of their contents. This is not a requirement, however. As far as XML is concerned, an element could be named *title* or *blort*. XML simply provides a general mechanism for marking up text. The semantics of an element's name or content must be agreed upon by a user community.

Suppose that we want to include the name of the person who made the change in Figure 6, we could add an element named *by* as in Figure 7.

```
<change>
  <by>Allison</by>
  <from>complicated</from>
  <to>sophisticated</to>
</change>
```

Figure 7 Secondary information as an XML element

However, the *by* element seems to indicate secondary information that pertains to the content of the element, whereas *from* and *to* indicate primary content. XML allows elements to have both primary and secondary content. The secondary content is indicated by an *attribute* of the element. Attributes are included within the start tag and have names and values as in Figure 8, where the element *change* now has an attribute *by* with value *Allison*.

```
<example>
  This is a more
  <change by="Allison">
    <from>complicated</from>
    <to>sophisticated</to>
  </change>
  example.
</example>
```

Figure 8 Secondary information as an XML attribute

Elements with no content or only secondary content are called *empty elements* and may have either a form such as <*page-break*><*/page-break*> or <*page-break/*>.

XML uses the Unicode character set. Characters may be inserted in text by reference using the form *{* where *123* is the number of the code point of the Unicode character to be inserted or *ካ* where *12AB* is the hexadecimal value of the code point of the character. This is most useful when the character does not fall into the ASCII set or is not easily entered directly from the keyboard.

Chapter 5: Design requirements

Before designing a system for the description of writing systems, it is helpful to have a clear statement of the requirements of such a system. Some requirements could be applied to systems of description in general, while others are specific to writing system descriptions, in particular.

5.1 General requirements

A system of description should accommodate different theoretical perspectives and allow for different practical analyses. This requirement acknowledges the transitory nature of theoretical analysis. The theory that is in vogue today will likely be replaced by another tomorrow. It also acknowledges that descriptions are not made without some theory. It is the theory that indicates the types of questions that we should be asking and thus the answers that we describe. Descriptions may span theories when those theories share similar elements and perspectives. However, it is understood that a system of description cannot truly be theory neutral—it can only avoid directly specifying the theoretical perspective in which it has been framed. As a corollary to this requirement, any system should be able to be described in its own terms. This comes from a presupposition that the categories developed in a language to refer to its writing system are more accurate descriptions than a category named artificially.[11] Thus, there is a theory behind every method whether implicit or explicit.

11 This idea has been defended in Ellis (1993).

Pike (1982) as well as Matthiessen and Nesbitt (1996) have called on linguists to realize the role that theory (both implicit and explicit) plays in their observations and thus their descriptions. “The theory is part of the observer; a different theory makes a different observer; a different observer sees different things, or sees the same things as structured differently” (Pike 1982:3). Summing up Pike’s (1982) statements on theory, Matthiessen and Nesbitt say “In Pike’s view, a linguistic theory is … a tool or resource for construing what we observe. It determines what we see and it allows us to construe patterns” (1996:51).

Since there are potentially many theoretical perspectives on the study of writing systems, a general system for the description of writing systems must accommodate the distinct perspectives that arise in the description while unifying these descriptions to a degree that allows systematic comparison.

Thus, in the description framework, it is taken for granted that a writing system description involves a description of one analysis of a writing system, which includes analysis of the graphic forms and the linguistic elements they represent. The writing system can be viewed as a grammar for writing. Halliday has claimed that “the grammar’s function is to construe: the grammar transforms experience into meaning, imposing order in the form of categories and their interrelations” (1996:7). Thus, the writing system description must indicate the categories and the interrelations that impose order on the writing system.

These categories and interrelations exist because of analysis. Halliday mentions Ellis (1993) as showing that “there are no natural classes: the categories of experience have to be created by the grammar itself. Or, we might say, there are indefinitely many natural classes: indefinitely many ways in which the phenomena of our experience may be perceived as being

alike" (Halliday 1996:8). There may be analyses that fit more closely with a native speaker's own sense as to what happens, which may be considered better analyses.

A system of description should be treated as a wholly contained unit, containing all the information that would be necessary for publication. This acknowledges that the information contained within descriptions is useful to the academic community at large and should be made available. Additionally this acknowledges that there are components which are characteristic of all publications that will be required as slots within this system.

The system should allow for partial descriptions as well as complete descriptions. This presumes that while more description is better than less, some description is better than none. Analysts may not have the time or resources to engage in the complete description of a writing system. Therefore, the analyst should be able to use the system to describe the extent of his research. In fact, sometimes, analysts desire "to present readers with the barest minimum of information" (Hartell 1993:viii) for the purpose of comparisons between languages. The system should make the analyst aware of the slots that he is omitting so that it is a conscious choice to omit information rather than oversight.

The system should allow elements to be declared and to be related to all the other elements that have been declared. Halliday suggests that there should be no distinction between "describing some feature and relating it to other features: describing anything consists precisely of relating it to everything else" (1996:21). Thus, analysis of a writing system consists of determining the elements of writing systems to be described and then relating them to "everything else".

5.2 Specific requirements for writing systems

In addition to the preceding general requirements for systems of description, the following requirements, specific to the description of writing systems, apply.

- The system must allow a machine to access and understand the information contained in the description.
- The system may additionally provide explanations of the formalism intended for humans, beyond what a computer can process.
- The system must allow linguistic analysis of the writing system.
- The system should allow for enough information to be provided about the rendering of the writing system such that a Graphite[12] implementation can be created by a human.
- The system should allow conversion between different writing systems that may exist for the same language.
- The system must be able to handle writing systems with any number of units from tens to thousands of writing system units.

The writing system framework that will be introduced, must handle all the above-mentioned issues to be considered complete and adequate.

12 Graphite is a rendering system currently being developed by SIL International which enables contextual glyph selection and positioning (Correll 2000).

Chapter 6: Elements of writing systems

The first step toward creating a model of writing system descriptions is to determine the components which need to be described.

This chapter explores all the elements that we may presuppose to exist as part of a writing system, namely, linguistic elements, graphs, graphemes, higher-level writing system units, classes of graphs, and computational units. These element types are presented in turn in the following sections. In each section, I first discuss how to identify these elements and then describe their properties. Following that, I provide examples and the formalism for those examples. In the next chapter, I will demonstrate how these elements can be related to each other.

6.1 Linguistic elements

Linguistic elements have a particular type, a name by which they can be referenced, and possibly a reference to a descriptive resource where one can find further information. These characteristics of linguistic elements are discussed below.

Herrick states that characters have "certain correspondences with linguistic units in the spoken form of its language" (1974:10). This view, that characters correspond to certain linguistic elements, has been the basis for the linguistic study of writing systems. Given that "the written notation symbolizes one aspect or another of linguistic structure" (Hall 1964:264), the questions then become, *what linguistic structures are symbolized and how*?

As defined previously, a *linguistic element* is "any unit of language". It is beyond the scope of this thesis to attempt to provide guidance in the identification of these linguistic elements. Gelb included " phrases, words, syllables, single sounds, and prosodic features" (1963:13–14) as linguistic elements. Because "writing systems operate on different levels and emphasize different units of language" (Coulmas 1989:270), the corresponding linguistic elements may be from any linguistic level, including "the phonemes, the morphophonemic alternations, or the morphemes of a language, or more than one of these at the same time" (Hall 1964:266).

Linguistic elements may include linguistically etic or emic units or even units representing some intermediate stage between the etic or emic units. We need to explicitly specify the type of linguistic unit, that is whether it is a phoneme, a phone, a morpheme, or something else. This is an open-ended list and can even include intermediate derivational forms.

Booij argued that intermediate linguistic elements such as those that exist in some analyses of the correspondences between phonetic information and the underlying phonemes "can be represented in Dutch spelling" (1987:216). "The spelling of the past tense singular of verbs with stems ending in underlying /v/ or /z/" (1987:219) illustrates this graphic representation of intermediate forms. The underlying form of "to rob" in the past tense singular is /rov+də/. After undergoing syllabification and syllable final devoicing, we are left with the intermediate form [[rofdə]]. A voicing assimilation rule restores the voicing to leave us with the final phonetic form [rovdə]. "The orthographical form ⟨roofde⟩ represents the intermediate level of derivation [[rofdə]]" (1987:219). The whole process can be seen in Figure 9 taken from (Booij 1987:219, no. 8).

underlying form	/rov+də/
syllabification	$(rov)_{\sigma}$ $(də)_{\sigma}$
syllable final devoicing	f
voice assimilation	v
phonetic form	[rovdə]

Figure 9 Dutch phonological derivation levels for ⟨roofde⟩

This illustrates clearly the need to include intermediate levels as a type of linguistic element in its own right for the correspondence between the linguistic unit and the graph is most succinctly stated in terms of the form which is only present in an intermediate level.

The linguistic elements of interest to writing system descriptions will ultimately be shown to correspond to graphic forms. Thus, part of the analysis of a writing system is to identify these linguistic elements.

It is not the place of writing system descriptions to describe the various linguistic elements that it will reference. The linguistic elements will simply be referenced and the interested reader may explore them in depth in publications which provide a detailed account of the particular type of unit. Thus, a writing system description assumes that linguistic elements have been adequately described elsewhere.

Because I am assuming that linguistic elements are fully described elsewhere and it is not the place of a writing description to do otherwise, we simply need to be able to make the link to the definition of the linguistic element in another publication. In everyday language, a name is probably sufficient for this purpose. Names are one way of establishing a link to a referent. If the definition of the linguistic unit exists electronically, we can formally reference this definition by means of a Uniform Resource Identifier (URI).

The properties of a linguistic element then are its type, its name, and optionally a link to another descriptive resource.

Orthographies most commonly correspond to the linguistic elements of phones, phonemes, syllables, and morphemes. Figure 10 demonstrates how the phone [ʃ] is represented formally as an instance of a linguistic unit.

```
<linguistic-unit type="phone"
                 id="phone-esh">
  <name>ʃ</name>
</linguistic-unit>
```

Figure 10 Simple formal instance of linguistic unit

Figure 11 demonstrates the formal representation of the phoneme /p/ as well as an example link to an electronic resource that describes the sound system.

```
<linguistic-unit
  link="www.sil.org/descriptions/phonology/English#phoneme-p"
  type="phoneme"
  id="phoneme-p">
  <name>p</name>
</linguistic-unit>
```

Figure 11 Complex formal instance of linguistic unit

When the syllable /kɑ/ is regarded as a basic linguistic element, it is represented formally as in Figure 12.

```
<linguistic-unit type="syllable" id="syllable-ka">
  <name>ka</name>
</linguistic-unit>
```

Figure 12 Formal instance of syllable as linguistic unit

This same syllable can be treated as though it were simply a sequence of phonemes, /k/ + /ɑ/, in which case the phonemes are listed one after another as in Figure 13. This treatment depends on the analysis. If the syllable as a basic unit were what corresponds

to the unit of writing, Figure 12 would be preferred. However, if the sequence of phonemes were considered to be what corresponds to the writing, Figure 13 would be preferred.

```
<sequence>
  <linguistic-unitRef target="phoneme-k"/>
  <linguistic-unitRef target="phoneme-a"/>
</sequence>
```

Figure 13 Formal instance of syllable treated as a sequence of phonemes

When a morpheme is regarded as the basic linguistic element, it is represented formally as in Figure 14.

```
<linguistic-unit type="morpheme" id="niàn">
  <name>niàn</name>
</linguistic-unit>
```

Figure 14 Formal instance of morpheme as linguistic unit

Just as syllables can be decomposed into a sequence of phonemes, so morphemes can be likewise decomposed as in Figure 15. Analysts must make their own decisions as to which linguistic unit should be treated as primary.

```
<sequence>
  <linguistic-unitRef target="phoneme-n"/>
  <linguistic-unitRef target="phoneme-i"/>
  <linguistic-unitRef target="phoneme-a"/>
  <linguistic-unitRef target="phoneme-n"/>
  <linguistic-unitRef target="phoneme-tone-4"/>
</sequence>
```

Figure 15 Formal instance of morpheme treated as a sequence of phonemes

The identifying names found as the content of the linguistic-unit element are arbitrary as far as the system is concerned but should be the actual name or other intelligible designation of the particular linguistic element.

When an intermediate level unit must be referenced, as in the case of Dutch mentioned above, the type should be *intermediate* and a link should be made to the level of rule at which the correspondence is made as in Figure 16.

```
<linguistic-unit
link="www.sil.org/desc/phonology/Dutch.xml#syl-final-devoicing"
type="intermediate"
id="inter-f">
  <name>f</name>
</linguistic-unit>
```

Figure 16 Formal instance of intermediate level as linguistic unit

6.2 Graphs

The most basic information associated with any description of a writing system, is the fundamental assumption that writing is built up from discrete segments, that is, "significant units of visually perceived form" (Hall 1960:13). Some of these graphic units directly correspond to linguistic elements while others, although identifiable, have the sole function of forming graphic units. The dot of the letter ⟨i⟩ forms a part of the letter and in no way can be said to correspond to some linguistic element. The letter as a whole does have a correspondence, however.

A graph is a unit of writing, recognized as a whole, which corresponds to a linguistic element and cannot be further decomposed into units of writing which correspond to linguistic elements. The graph is then the basic symbol of the writing system. The letter ⟨i⟩, then, is a graph but the dot is not.

Sproat points out the difficulty that may be encountered in determining which unit should be designated as the graph:

> It is convenient to refer to a single Chinese character as being a [graph]….[13] … However, there is clearly important internal structure in Chinese characters … and from the point of view of a finer-grained analysis of Chinese writing, these smaller units would certainly be called … [graphs]. (Sproat 2000:28)

This finer-grained analysis structurally decomposes many Chinese characters into corresponding characters which provide clues as to the composite's referent. One such way is to combine two characters such that each provides a semantic component. For example, 〈好〉 [χǎu] "good" can be seen as a combination of the character 〈女〉 "female" and the character 〈子〉 "child". Another system of structural compounding combines one character which provides the semantic component with another which provides the phonetic component. For example, 〈蝉〉 [tʂʰán] "cicada" can be seen as a combination of the character 〈虫〉 "insect" and the character 〈单〉 [tʂʰán] (Sproat 2000:144). Since each of these parts can be corresponded to a linguistic unit, Chinese characters would need to be decomposed into their respective parts and each of these parts treated as a distinct graph. Thus, 〈好〉 [hǎo] "good" is composed of two graphs: 〈女〉 "female" and 〈子〉 "child".

Sproat has compiled evidence to support this decomposition:

> While Chinese characters certainly contain nonphonological information, it is nonetheless the case that skilled Chinese readers have learned an association between characters and their corresponding syllables that allows for very rapid access to the phonological form, in effect bypassing the rest of lexical access. (Sproat 2000:172)

Other evidence demonstrates that Chinese readers use the phonetic component when present to read unfamiliar words:

---

13 Sproat uses the term *grapheme* here "as a convenient short way of saying *basic symbol of a writing system*" (Sproat 2000:28). I use the term *graph* to refer to the same concept.

> Chinese readers, when encountering an unfamiliar character, will attempt to guess its pronunciation from the phonetic component. Indeed, with a completely unfamiliar character, they have no choice but to adopt this strategy. An instance of this is the character 鳕 <FISH+XUE> *xuě* "cod". Apparently, this character was a Japanese invention… where the second element 雪 was used not for its pronunciation *xuě* but for its meaning "snow" (the flesh of the cooked cod being snowy white). Thus, the correct analysis for Japanese would be <FISH+SNOW>…. When this character was borrowed back into Chinese, Chinese readers interpreted the 雪 component as a phonetic component, thus assigning the character the pronunciation *xuě*. (Sproat 2000:146)

Therefore, in order to fully account for the workings of the Chinese writing system, including the knowledge that a literate person has, the decompositions must be accounted for.

A similar problem arises with English and other languages which use digraphs to represent a single linguistic element. When ⟨ch⟩ occurs in the word ⟨church⟩, is it a single graph, since it represents a single phoneme /ʧ/? No, because ⟨ch⟩ can be subdivided into two graphs which can be shown to have a linguistic correspondence. It is a digraph, a sequence of two graphs with a single function. The same is true of the use of ⟨ll⟩ in Spanish. The fact that a Spanish literate would probably want to give equal status to ⟨ll⟩ and ⟨l⟩ is taken up in Section 6.3.[14]

The basic description of a graph begins with the name of the character to enable basic communication concerning the writing system. Herrick believes that the name is a universal property of every graphic unit in any system of writing: "whenever a written mark is an

---

14 It should be evident to the astute reader that a certain amount of theory has been presented here in the form of determining what a graph is and how to identify one. This may seem to contradict the earlier stated goal of not introducing a bias toward a particular theory within the description methodology. It should be obvious that even the designation of the units as graphs involves analysis. The system allows analysts to make their own decisions as to what constitutes a graph which may differ from my discussion here.

embodiment of a letter of a certain language, any normal literate of that language can name the letter which it embodies" (1974:10). Because writing is taught, it is reasonable to assume that there is language for referring to the parts of writing, a "writing system metalanguage," which is used in teaching and in communication about the process of writing.

Thus, there are two ways to designate the names of graphs from the perspective of the users of the writing system. The first is to use a phonetic transcription of the name. The second is to use a "native" spelling. Other user communities may have designated other names. For example, the study of ancient writing forms often leads researchers to give names to the observed forms despite the lack of any native speakers.

Since there are potentially many different names for the same graph, the name's type can be indicated. If the type designation is omitted, it is assumed that the name represents the native community's name for the graph in the native orthography. A type of *pronunciation* indicates that the name represents the native community's pronunciation of the name. Types of *translation* or a particular transcription may also be useful. The first of the names is considered the most common.

In addition to the property of *name*, every graph has a prototypical *form*. This can be demonstrated using an image. This image may in addition, diagram the method for writing the particular graph by means of a set of stroke patterns and directions, as in Figure 17, which illustrates the stroke patterns used to produce the Burmese character top indented ba.

Figure 17 Graph stroke patterns for Burmese top indented ba

A *unique identifier* is also assigned to each graph so that it can be referenced by other parts of the system.

Other characteristics that may be specific to a particular writing system will be accounted for in the section on classes below.

For American English, the letter ⟨a⟩ has the name [ʔɛⁱ], the letter ⟨b⟩ has the name [bi], ⟨c⟩ has the name [si], and so on as shown in Figure 18.

```
<graph id="a">
  <name xml:lang="x-ENG-IPA"
        type="pronunciation">ʔɛⁱ</name>
</graph>
<graph id="b">
  <name xml:lang="x-ENG-IPA"
        type="pronunciation">bi</name>
</graph>
<graph id="c">
  <name xml:lang="x-ENG-IPA"
        type="pronunciation">si</name>
</graph>
```

Figure 18 English names of graphs

Brazilian Portuguese has other names for the same graphic forms: the letter ⟨a⟩ has the name [ʔɑ], the letter ⟨b⟩ has the name [bɛ], ⟨c⟩ has the name [sɛ], and so on as shown in Figure 19.

```
<graph id="a">
  <name xml:lang="x-POR-IPA"
        type="pronunciation">ʔɑ</name>
</graph>
<graph id="b">
  <name xml:lang="x-POR-IPA"
        type="pronunciation">bɛ</name>
</graph>
<graph id="c">
  <name xml:lang="x-POR-IPA"
        type="pronunciation">sɛ</name>
</graph>
```

Figure 19 Portuguese names of graphs

The markup for the Burmese character *great ka* is given in Figure 20. The identifier of the character is only used internally so here I have simply used the designator *ka*. It could have just as easily been *bur-1* or even *d1fuo4s*, or anything else you may choose. The English translation of the name is given as well as the name in Burmese. Another transliterated form of the name as established by William S. Cornyn in the teaching of Burmese is also presented.

```
<graph id="ka" image="ka.gif">
  <name xml:lang="x-Burmese">ကိ:</name>
  <name type="translation" xml:lang="EN-us">great ka</name>
  <name type="Cornyn transcription">ka.ji:</name>
</graph>
```

Figure 20 Formal instance of Burmese graph declaration

6.3 Graphemes

It is obvious that the surface units of writing, the graphs, exist. Nevertheless, do emic and etic principles apply to these graphs in the same way that they do to linguistic elements? If so, what are the underlying forms of graphs? These questions have been debated extensively.

Many writing system researchers "assume the existence of a unit called *grapheme* and build from there" (Daniels 1992:528). Thus, for many researchers of writing systems, the answer to these questions is yes, emic and etic principles apply to graphs in the same way that they do for linguistic elements and that the underlying forms of graphs are graphemes.

However, Daniels has seriously questioned the validity of such assumptions. His argument can be summarized as follows: "No linguist should use a term *grapheme* without a detailed, explicit theory of graphemics, graphetics, allographs, maybe archigraphemes and other accoutrements of emic theory" (1992:528). No such theory exists. In addition, linguists

have used the term *grapheme* with different meanings, each of which identifies disparate segments that are identified as graphemes. Daniels claims that "there is no coherent definition of *grapheme* if the morpheme *eme* is to have any consistent meaning within linguistics" (1992:531). He then presents ways in which writing systems are dissimilar to phonemic systems. Ultimately, he disregards the application of emic theory to writing:

> Emic terminology relates to a property of the human mind. It applies to language, and to culture, and many aspects of human behavior because human brains have evolved over the myriadennia in a way that produces such behavior.… Therefore, the emic system should not be expected to apply to writing because writing is a conscious invention.… The notion of *graphemics* ought to be jettisoned, and with it the elusive—the impossible—concept of *grapheme*. (1992:534)

To my knowledge, Herrick has been the only one to challenge Daniels' claim in print. He challenges the idea that emic theory cannot be applied to conscious inventions:

> The fact is that people can learn not only the phonologies but also the writing systems of languages, and there are linguistic theories which can handle this fact. Our brains are equipped to receive sensations from both our eyes and our ears. And, if our brains have evolved a general reasoning power which can handle all those kinds of relationships that occur throughout language, and if this reasoning power includes, among other things, the power to distinguish between the emic and the etic—between differences that matter and differences that do not matter, then any human can learn both a writing system and a phonology of any language. (1995a:423)

Thus, the issue of emic theory is essentially the capability which all humans possess to classify items as similar although these appear to be distinct from the outsider's perspective:

> The basic phenomenon in phonemics and graphemics, as well as in semantics and in non-linguistic human activities, seems to be the faculty of the brain to classify numerous single items as members of a much smaller number of species. Ultimately this labor of sorting and classing is a device of economy,

> designed to crystallize the relevant features and abstract them from a mass of non-distinctive individual details. Epistemologically and linguistically … this may well be the most humanly intelligent performance of the human intellect. (Pulgram 1951:20)

In his reply to Herrick, Daniels conceded the possibility of forms that vary given a particular context:

> For him, round and lunate Greek *sigma* or long and curly *s* in earlier Roman type are examples of allographs; for me, those are one possible parallel of conditioned allophonic variation—but so is the Etruscan use of ⟨K⟩ before ⟨A⟩ and ⟨O⟩, ⟨C⟩ before ⟨E⟩ or ⟨I⟩, and ⟨Q⟩ before ⟨U⟩ for /k/. (1995:427)

For many this would seem to prove the case that the grapheme is an element in its own right. Sgall sees the example of Greek sigma as a classic case requiring a grapheme:

> Different characters are variants of a single *grapheme* if they do not differ in their systematic function … and … each of them occurs only in such contextual positions (determined by graphemic context) in which the other cannot occur (combinatory variants, such as the two forms of Greek sigma). (1987:8)

Because the graphs ⟨ ⟩ and ⟨ ⟩ can be shown to exist in complementary distribution, we could posit the existence of an underlying form that is a grapheme representing the two varieties of Greek sigma. We can then write rules to explain the correspondences between the grapheme and its surface forms: ⟨ ⟩ is only found word finally while ⟨ ⟩ is found in every other environment. The phoneme /s/ then corresponds to this grapheme sigma and we are left with the rules in Figure 21.

/s/ → sigma
sigma → ς / __ #
sigma → σ / elsewhere

Figure 21 Rules for Greek sigma including grapheme

Although this demonstrates that a grapheme can be postulated, it still does not demonstrate the need for such an element. After all, there is only a simple one-to-one correspondence between the phoneme /s/ and the grapheme sigma. If we omit the grapheme, we can still account for the data using rules such as those in Figure 22. Since this is the simpler explanation, it is also the better explanation in the absence of any other role for the grapheme sigma in the description.

/s/ → ς / __ #
/s/ → σ / elsewhere

Figure 22 Rules for Greek sigma omitting grapheme

Therefore, simply assuming a unit called a *grapheme* due to evidence of complementary distribution does not buy us anything. Unfortunately, Daniels does not make this point explicitly, or it would have made his case even stronger. At any rate, the burden of proof is placed on someone to show that either (1) there exists a relationship between the sound system (or another linguistic system) and the writing system, which can be accounted for more simply by the use of the grapheme than without it, or (2) that the grapheme is a unit which is psychologically real in that it exists in the mind of the literate user of the writing system. The fallacy of many who seek to demonstrate the psychological reality of the grapheme is that they unintentionally equate the phoneme with the grapheme, thereby demonstrating the psychological reality of the phoneme rather than the grapheme.

Perhaps the best evidence of the psychological reality of the grapheme is when the "letters of the alphabet" are itemized by a speaker of the language. This in itself is problematic in that many aspects of the writing system are omitted in such a listing. For example, punctuation characters and the numbers are omitted in the English alphabet,

although they form a very real part of the writing system. However, the fact that Greek sigma is not listed twice seems to provide at least some psychological evidence for the reality of a single grapheme.

As mentioned in the section on graphs, a Spanish literate would probably want to give equal status to ⟨ll⟩ and ⟨l⟩. This is because ⟨ll⟩ would be listed as a letter of the alphabet by a Spanish literate. Thus, when putting words into alphabetic order, ⟨ll⟩ is treated as a single unit that falls between ⟨l⟩ and ⟨m⟩. Thus there seems to be evidence that ⟨ll⟩ is considered a single unit by a Spanish literate. Such psychological evidence seems to be enough to merit the introduction of the grapheme as the unit to which ⟨ll⟩ would be assigned.

Since the purpose of this paper, is to support description of a wide variety of writing systems using many theoretical perspectives, the issue of what designates a grapheme is not as important as the fact that the possibility for such a unit of description exists. Different analyses may designate different graphemes. The grapheme is an emic unit, which represents the form that a literate user deems important. However, this does not mean that every writing system analysis would find the concept of grapheme useful or necessary.

A grapheme has the same properties as a graph: namely, a *unique identifier* so that the grapheme can be referenced by other parts of the system, a set of *names* by which the grapheme is referenced in speech and a prototypical *form*.

The declaration for the Spanish grapheme ⟨ll⟩ is represented in Figure 23.

```
<grapheme id="ll" image="ll.gif">
  <name xml:lang="x-ES-IPA" type="pronunciation">ɛʎɛ</name>
</grapheme>
```

Figure 23 Formal instance of grapheme declaration

6.4 Writing system units

Although graphs (or possibly graphemes) are the building blocks of writing, there are higher-level units that are created when these building blocks are arranged. Orthographic syllables, words, phrases, and sentences are all made from arrangements of these building blocks. Because these units are dependent on the internal structure of the writing system, a classification system is writing system dependent. Thus, in the description system, I provide the mechanism for declaring these higher-level units and relating them to each other.

The writing system unit has a name that corresponds to the level of the unit. It also has a unique identifier so that it might be referenced within the system. In addition to these features, the writing system unit has a composition declaration that defines the unit in terms of lower-level writing system units or graphs (or graphemes). The composition declaration uses rewrite rules to define these writing system units.

In English, words are composed of the letters of the alphabet. They may also have a hyphen or an apostrophe as part of the word. The formal declaration for English words is given in Figure 24.

```
<writing-unit id="word">
  <name>orthographic word</name>
  <is-made-of>
    <sequence>
      <repeatable>
        <classRef target="alphabet"/>
      </repeatable>
      <optional>
        <choice>
          <sequence>
            <graphRef target="apostrophe"/>
            <optional>
              <repeatable>
                <classRef target="alphabet"/>
              </repeatable>
            </optional>
          </sequence>
          <sequence>
          <graphRef target="hyphen"/>
            <repeatable>
              <classRef target="alphabet"/>
            </repeatable>
          </sequence>
        </choice>
      </optional>
    </sequence>
  </is-made-of>
</writing-unit>
```

Figure 24 Formal instance of word writing unit declaration

Sproat discusses the need for "a more powerful notion than simple concatenation" (Sproat 2000:34) and introduces five catenation operators to handle some of the more complex script phenomena such as that exhibited by Chinese characters: leftwards catenation, right catenation, downwards catenation, upwards catenation, and surrounding catenation. He believes that inside catenation is not necessary because "the only cases where such placement occurs is in fossilized forms" (Sproat 2000:50), but I include it as a possibility since it may prove necessary for some analyses other than Chinese. While the notion of catenation is valuable to East Asian languages, its benefit for the other writing systems of the world remains to be shown.

For instance, Sproat gives the following illustration of catenation:

The Chinese 鱗 *lín* “fish scale” is composed of the components, 魚, 米, 夕, and 牛, arranged as follows: 魚→[米↓[夕　牛]]. (Sproat 2000:37)

Figure 25 illustrates the formal declaration for the catenation involved in this example:

```
<catenate type="right" minOccurs="1" maxOccurs="1">
  <graphRef target="魚"/>
  <catenate type="down" minOccurs="1" maxOccurs="1">
    <graphRef target="米"/>
    <catenate type="right" minOccurs="1" maxOccurs="1">
      <graphRef target="夕"/>
      <graphRef target="牛"/>
    </catenate>
  </catenate>
</catenate>
```

Figure 25 Formal instance of catenation declarations

A general declaration for a Chinese character is shown in Figure 26.

```
<writing-unit id="character">
 <name>character</name>
 <is-made-of>
  <choice>
   <repeatable>
     <catenate type="down">
       <classRef target="d-component"/>
       <choice>
         <classRef target="component"/>
         <writing-unitRef target="character"/>
       </choice>
     </catenate>
   </repeatable>
   <repeatable>
     <catenate type="left">
      <classRef target="l-component"/>
      <choice>
       <classRef target="component"/>
       <writing-unitRef target="character"/>
      </choice>
     </catenate>
   </repeatable>
   <repeatable>
     <catenate type="up">
       <classRef target="u-component"/>
       <choice>
         <classRef target="component"/>
         <writing-unitRef target="character"/>
       </choice>
     </catenate>
   </repeatable>
   <repeatable>
     <catenate type="surround">
       <classRef target="s-component"/>
       <choice>
         <classRef target="component"/>
         <writing-unitRef target="character"/>
       </choice>
     </catenate>
   </repeatable>
   <repeatable>
     <catenate type="right">
       <classRef target="component"/>
       <choice>
         <classRef target="component"/>
         <writing-unitRef target="character"/>
       </choice>
     </catenate>
   </repeatable>
  </choice>
 </is-made-of>
</writing-unit>
```

Figure 26 Formal declarations of Chinese writing units

Korean also has a syllabic unit which requires catenation, however it is not nearly as complicated as Chinese. Korean always has a orthographic syllable structure of CV or CVC, or CVCC. The final consonant cluster is always placed under the catenation of the onset and the nucleus. However, the onset and nucleus may catenate horizontally or vertically and are determined by the class of the vowel. A horizontally oriented vowel causes downward catenation, while a vertically oriented vowel causes leftward catenation.

For example, the syllable 〈못〉 is constructed from the combination of 〈ㅁ〉, 〈ㅗ〉, and 〈ㅅ〉; while 〈잘〉 is constructed from the combination of 〈ㅈ〉, 〈ㅏ〉, and 〈ㄹ〉.

Korean also has a special situation in which a vowel may be both horizontally oriented as well as vertically oriented and cause the consonant to occur in the upper left hand corner of the syllable block. For example, the syllable 〈쉼〉 is constructed from the combination of 〈ㅅ〉, 〈ㅟ〉, and 〈ㅁ〉. At first glance, this seems like a case that Sproat's catenation operators cannot handle. However, the surround operator can be used here to accomplish the desired effect, despite the fact that it is only partially surrounding the consonant.

The formal rules for Korean catenation can be found in Figure 27.

```
<writing-unit id="syl">
  <name>syllable</name>
  <is-made-of>
    <catenate type="down">
      <writing-unitRef target="onset-nucl"/>
      <writing-unitRef target="coda"/>
    </catenate>
  </is-made-of>
</writing-unit>

<writing-unit id="coda">
  <name>coda</name>
  <is-made-of>
    <catenate type="left">
      <classRef target="C"/>
      <optional>
        <classRef target="C"/>
      </optional>
    </catenate>
  </is-made-of>
</writing-unit>

<writing-unit id="onset-nucl">
  <name>onset nucleus</name>
  <is-made-of>
    <choice>
      <catenate type="left">
        <classRef target="C"/>
        <classRef target="vertical-V"/>
      </catenate>
      <catenate catenate="down">
        <classRef target="C"/>
        <classRef target="horizontal-V"/>
      </catenate>
      <catenate catenate="surround">
        <classRef target="C"/>
        <classRef target="surround-V"/>
      </catenate>
    </choice>
  </is-made-of>
</writing-unit>
```

Figure 27 Formal instance of Korean writing unit declarations

One may also desire to indicate "the direction of writing …[as] a property of the writing system" (Simons 1989:549).

> What we are referring to here … are the features of *orientation*. These are (freak systems apart): *axis*, horizontal or vertical; *direction*, left-to-right/right-to-left or down/up; and "*lining*" (often overlooked), i.e. whether the lines, in a horizontal script, succeed one another downwards or upwards, or, in a vertical

> script, succeed one another from right to left (as they do in fact in traditional Chinese and Japanese writing) or from left to right. (Mountford 1990:706)

Figure 28 contains the catenation declarations for volumes, pages, and lines for Korean. Notice that each level references the lower-level writing unit that composes it.[15]

```
<writing-unit id="volume">
  <name>volume</name>
  <is-made-of>
    <repeatable>
      <catenate type="left">
        <writing-unitRef target="page"/>
      </catenate>
    </repeatable>
  </is-made-of>
</writing-unit>

<writing-unit id="page">
  <name>page</name>
  <is-made-of>
    <optional>
      <repeatable>
        <catenate type="down">
          <writing-unitRef target="line"/>
        </catenate>
      </repeatable>
    </optional>
  </is-made-of>
</writing-unit>

<writing-unit id="line">
  <name>line</name>
  <is-made-of>
    <repeatable>
      <catenate type="left">
        <writing-unitRef target="syl"/>
      </catenate>
    </repeatable>
  </is-made-of>
</writing-unit>
```

Figure 28 Formal instance of Korean writing unit declarations

15 Line breaking and page breaking mechanisms are outside the scope of this work.

6.5 Classes

Classes of graphs (or graphemes) provide support for abstract pattern searches and are used internally within the system. In the preceding section, classes were used to build the structural patterns. When the sonorants are designated by creating a class of sonorants, we are able to support the complex search requirements imposed by Antworth and Valentine:

> Often one has a particular hypothesis in mind and wants to search a data set for specific data items that will confirm or disconfirm the hypothesis. Searching software must permit abstract pattern matching, not just finding literal forms. For instance, most word processors permit you to search for a literal string such as *phoneme*; but very few word processors permit you to search for a pattern such as *any word containing only sonorants*. (1998:174)

Any set of graphs that share a feature form a class. Thus, all the members of a class could be said to have the feature with the same name as the class's name. Classes and features can be interpolated from each other, thus I have chosen to only allow the description of features in terms of class designations. This is for the purpose of storage so that there is a single representation for the information, but it in no way limits the potential for another system to overlay the same information to the user in terms of features.

In addition to having a set of graphs, classes have a unique identifier (for internal use) and a set of names.

To create a definitive list of features or classes for all writing systems would be a noble task but it is not necessary for our purposes. In order to support any writing system, we must simply provide a mechanism for assigning graphs to classes and leave the choice of class designations up to the analyst. I follow Pike's advice that "the *features themselves* must be treated as emic and as *etically variable*" (1982:88).

In one analysis of the English writing system the letters of the alphabet have two forms, one uppercase (e.g. ⟨A⟩, ⟨B⟩, ⟨C⟩, …) and the other lowercase (e.g. ⟨a⟩, ⟨b⟩, ⟨c⟩, …). Thus, we can differentiate these graphs by creating two classes: *lowercase* and *uppercase* as in Figure 29.

```
<class id="lower">
  <name>lowercase</name>
  <graphRef target="a"/>
  <graphRef target="b"/>
  <graphRef target="c"/>
  …
  <graphRef target="z"/>
</class>
<class id="upper">
  <name>uppercase</name>
  <name>capitalized</name>
  <graphRef target="A"/>
  <graphRef target="B"/>
  <graphRef target="C"/>
  …
  <graphRef target="Z"/>
</class>
```

Figure 29 Formal instance of class declarations for case

Classes can be defined to include other classes as in Figure 30.

```
<class id="letter">
  <name>letter</name>
  <classRef target="upper"/>
  <classRef target="lower"/>
</class>
```

Figure 30 Formal instance of class declarations including other classes

As a shortcut, classes can also be defined to include another class and then to exclude particular members as in Figure 31 where the class designating the consonants includes all the letters excluding the class of vowels. There is a need to avoid circular declarations, and thus all class declarations must ultimately be reducible to a set of graphs (or graphemes).

```
<class id="C">
  <name>consonant</name>
  <classRef target="letter"/>
  <excluding>
    <classRef target="V"/>
  </excluding>
</class>
```

Figure 31 Formal instance of class declarations excluding other classes

One of the issues that arises with many Indic writing systems, is that a particular graph may only exhibit a particular feature if it is in a particular position. Thus, the interpretation of which class a particular graph is in may be context sensitive. In Hindi, for example, each graph denotes a particular consonant and a default vowel. The default vowel may be changed by the addition of a vowel diacritic sign. "A following short vowel *a* is considered inherent in each consonant symbol; thus, unless these letters are modified by other attached symbols, प is *pa*, and र is *ra*.… Vowels other than *a*, when they follow a consonant, are written as obligatory diacritics … as in पु *pu*" (Bright 1996:387). Thus, the definition of a class must include the ability to specify the context in which a particular graph is a member of the class. Some example classes to handle Hindi are demonstrated in Figure 32 where the base consonant is a consonant followed by a vowel diacritic (thus not including the inherent vowel) and all other consonants include an inherent vowel.

```
<class id="base-consonant">
  <name>base consonant</name>
  <classRef target="consonant"/>
  <when>
    <followed-by>
      <classRef target="vowel-diacritic"/>
    </followed-by>
  </when>
</class>

<class id="consonant-a">
  <name>consonant with an inherent vowel</name>
  <classRef target="consonant"/>
</class>
```

Figure 32 Formal instance of contextually determined class membership

6.6 Computational units

From the computational perspective, the computational task of handling writing is split into three areas: input, storage, and rendering. Each of these has its own units: key codes, code points, and glyphs, respectively. We will explore each of these. A computer stores writing by encoding characters. “Encoding is governed by a single, basic fact: the computer can store only numbers” (Becker 1984:96). Thus, each of the computational units will center around a number which the computer uses to represent the particular aspect of digital writing.

6.6.1 Key codes

A key code is a number that is generated by the keyboard when a particular key or combination of keys are pressed. When the *A* key on the keyboard is pressed, the number 65 is sent as the key code to the operating system and made available to the current application. When the *A* key and the *Shift* key on the keyboard are pressed at the same time, the number

81 is sent as the key code to the operating system and made available to the current application.

Although this is technically what is happening from the computer software's perspective, this does not reflect the perspective of a user. From the perspective of the user, he is pressing a labeled key or combination of keys. Thus, rather than require a user to determine the number that the system generates when a key or combination of keys is generated, a better solution is to allow the user to define the keys that are significant using labels.

There is a problem with this though, in that different keyboards have different sets of keys. For example, a Brazilian keyboard has a key dedicated to the *ç* letter. This key is lacking on all American keyboards. Therefore, this framework will provide a set of standard names for keys but this set is not considered all-inclusive. An implementation which manages the keyboard from the information in these writing system descriptions must provide a way for the user to add named keys and to associate them with their appropriate value. Thus, the task of associating a key label with a key code has been left up to an implementation.

The keystroke is a set of keys, indicated by their labels, that must be pressed simultaneously. The keystroke that occurs when the *Control* key, the *Shift* key and the *A* key are pressed simultaneously is represented in Figure 33.

```
<key-stroke value="CONTROL SHIFT A"/>
```

Figure 33 Formal instance of key code declaration

6.6.2 Coded units

Coded units are numbers which identify a particular character within a character set. These may be standardized character sets, such as ASCII or Unicode or may be specialized character sets. Coded units are the representation of writing that is stored by the computer.

Because different character sets have a different number of characters within their inventories, some character sets require more space per character than others do. Therefore, we need to determine the significant units. The character representing the letter ⟨A⟩ in English is given the coded unit with a value of *65*. In the ASCII character set, this value is stored within a context of 8 bits (enough to handle 256 characters or 2^8). In the Unicode character set, this value is stored within a context of 16 bits (enough to handle 65536 or 2^{16}). Currently character sets use either 8, 16 or 32 bits of significance.

The coded unit for the letter ⟨A⟩ in the Unicode character set is shown in Figure 34

```
<coded-unit bits="16" value="65"/>
```

Figure 34 Formal instance of coded unit declaration

6.6.3 Glyphs

A glyph is a reference point into a particular font or set of fonts. Hosken et al. in their *Graphite Description Language* identify four ways to identify a glyph (2000:8):

- by using the actual internal glyph number in the font.
- by Unicode value via the internal character map (cmap) in the font, which takes a Unicode codepoint number and returns a glyph number.
- by Postscript name
- by 8-bit character code according to a codepage and then via the font character map.

The Unicode value and the character code are mappings themselves. They will not be used to identify a glyph lest there be circular references. Thus, the glyph potentially has a number which corresponds to the actual glyph number in the font, or a name. The number is a decimal number. The font must be named when the glyph number is designated. The font may be a list of comma-separated font names. The glyph can also have an image. Figure 35 provides an example of the glyph for the ⟨Ç⟩ character.

```
<glyph number="100"
       name="Ccedilla"
       font="Arial, Times New Roman"
       img="Ccedilla.gif"/>
```

Figure 35 Formal instance of glyph declaration

Chapter 7: Relationships between writing system elements

The major load of information in writing system descriptions involves relationships. There are relationships between graphs themselves, relationships between graphs and linguistic elements, between graphs and glyphs, graphs and their computerized encodings. There may even be a relationship between the graphs of one writing system and those of another as is commonly found in transliterated material. In addition, there are relationships between graphs that must also be expressed. For our purposes, the relationships between different types of elements of writing systems are mappings between these types. The relationships between elements of the same type are relations.

7.1 Potential relationships

The question of how the elements of writing systems relate to each other is not as simple as it may seem. From the linguistic perspective, the situation is straightforward. The linguistic units correlate to graphs as in Figure 36 such that one may relate a linguistic unit to a graph or a graph to a linguistic unit.

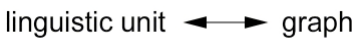

Figure 36 Linguistic relationships

All the relationships present from a linguistic perspective are bi-directional. That is, the mapping from linguistic units to graphs (writing) and the mapping from graphs to linguistic units (reading) are both significant. In the diagrams in this section, arrows indicate

the significant mapping relationships. A bi-directional mapping relationship is indicated by a double arrow. Each arrow represents a set of rules which can be written to account for the mapping relationships between the members.

If one includes graphemes in one's analysis, the linguistic units correlate to graphemes and the graphemes correlate to graphs as in Figure 37. The relationship between the linguistic units and the graphs is indirectly specified by means of the graphemes and can be inferred. Thus, the linguistic units may be mapped to the relevant graphemes or graphs (by inference). The graphemes may be mapped to the relevant linguistic units or graphs; and the graphs may be mapped to the relevant graphemes or linguistic units (by inference).

Figure 37 Linguistic relationships including graphemes

Figure 37 represents the conceptual processes involved in writing systems as language is given written expression and this written expression is in turn interpreted back into language.

The perspective from a computational vantage is very different from the linguistic relationships since computational units do not deal with conceptual processes but rather physically implement writing for a computer. The computational units are related as in Figure 38 where key codes produce code points which in turn produce glyphs.

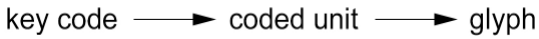

Figure 38 Computational relationships

Whereas, the linguistic relationships were bi-directional, the computational relationships are unidirectional. Although key codes must be transformed into the appropriate coded unit, the converse is not true, that is, coded units do not need to be transformed into key codes.

These two perspectives, the linguistic and the computational, have largely existed apart from each other. This thesis seeks to provide a mechanism to formally integrate the two. In order to do so, we must now introduce a third perspective, that of the computational implementer. The computational implementer seeks to understand how a writing system works as understood from the linguistic perspective and then to design the implementation of the writing system in the computational system in as straightforward a manner as possible. The computational implementer must relate the linguistic perspective to the computational perspective and vice versa.

Typically, a computational implementer will seek to provide simple relationships between the linguistic perspective and the computational perspective. For example, Figure 39 illustrates a situation where a computational implementer has represented graphemes by means of coded units (ideally with a one-to-one relationship) and has represented graphs by means of glyphs.[16]

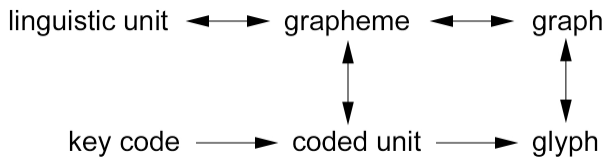

Figure 39 Computational implementation of writing

16 In the event that graphemes are not necessary to the analysis, the graphs are generally represented by coded units and by glyphs.

Though it may be most common for coded units to represent graphemes, this is not the only way to design an implementation. It is possible that the coded units could represent graphs, graphemes, or even linguistic units at different levels. Multiple kinds of relationships can be mixed within the same system, though such a system would be considered far from ideal. The relationship that can be expressed most concisely is the better choice.

The mappings discussed above may be classified in terms of the perspective that they describe: linguistic, computational, or implementational. Figure 39 represents mappings in each of these groupings. The mapping describing the relationships between linguistic units and graphemes is a linguistic mapping. The mapping describing the relationships between coded units and glyphs is a computational mapping and the mapping describing the relationships between graphemes and coded units is an implementational mapping.

All the relationships that are represented in Figure 39 need not be enumerated for a complete description. Given the relationships in Figure 39, it is possible to determine the glyphs for a particular coded unit by following one of two paths: from coded unit to glyph, or from coded unit to grapheme to graph to glyph. Thus, either the relationship from graph to glyph or the relationship from coded unit to glyph is superfluous since either could be inferred given the other.

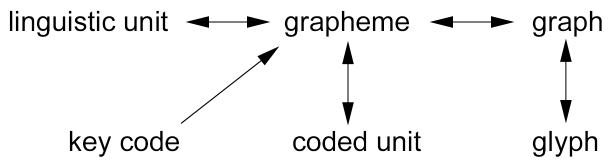

Figure 40 Minimal set of linguistic and computational relationships

Figure 40 illustrates a minimal number of mappings needed to accurately describe all the relationships. These mappings have been chosen to represent the conceptual relationships from the perspective of a user familiar with the writing system. Thus, the key codes have been related to the graphemes rather than the coded units since a user is indicating the particular grapheme by his key press and does not even need to know of the existence of coded units. Likewise, the glyphs have been related to the graphs since they most closely represent the actual forms of the graphs. It is interesting to note that no computational mappings are made explicit in Figure 40. They may all be inferred given the other relationships which are explicitly specified.

When graphemes are not present in the analysis, the graphs become a natural constellation point through which all other elements relate. Thus, a key code would indicate a particular graph. A graph would be expressed by a glyph and a graph would be stored by a coded unit. Historically, the computational designer's choice has been to equate glyphs and graphs (thus we see a tendency toward precomposed glyphs as part of fonts rather than decomposed ones) and, in addition, to equate code points and graphs. This simplified the process by exclusively using one-to-one relationships. Unfortunately, many processes now assume these one-to-one correspondences, even when many of the world's writing systems require relationships that are more complex. Forced oversimplification of one relationship leads to another overly complicated relationship. A good designer must balance each relationship in order to determine the best way to achieve overall simplicity in the implementation.

Because this approach to writing system description does not seek to bind one's hands, but rather to simply provide the tools to make relationships explicit, it is able to

express the relationships between any elements of writing systems. Of course, as with any analysis, the more concise and simple the analysis, while still accounting for all the known data, the better.

It should be noted that a writing system description that includes only the linguistic relationships provide an adequate description of an orthography. From such a description, a computational designer could determine the key codes, coded units and glyphs that would be necessary, and could determine the simplest way to implement them in terms of the linguistic relationships. The other types of relationships (computational and implementational) are necessary only to describe how the computer implementation works in terms of the writing system.

7.2 Mappings

The correspondences between linguistic elements and graphs are rarely one-to-one correspondences, for Gelb has aptly concluded that "writing can never be considered an *exact* counterpart of the spoken language. Such an ideal state of point-by-point equivalence in which one speech unit is expressed by one sign, and one sign expresses only one speech unit, has never been attained in writing" (1963:15).

Haas concludes that the correspondences are "determined by rules and tendencies of *glotto-graphic translation*: from speech into writing, and from writing into speech" (1983:19). These rules must "identify the items that are matched" (page 19), and "state the contextual conditions of that match" (page 19). These contextual conditions may be simple, corresponding to a single linguistic level, or they may "refer to co-occurring elements of *other* levels as contextual conditions of the match" (page 20). Sproat joins Haas

in affirming that "the mapping between linguistic representation and orthography can be handled by context-sensitive rewrite rules" (Sproat 2000:16).

The linguistic correspondence to the graphs is not the only relationship that requires more power than simple one-to-one correspondences. Simons warns that "in many writing systems, a given letter of the alphabet is realized by different graphs in different contexts" (1989:540). Becker (1984) described the need to break the singleness of correspondence between character codes used in storage and the glyph identifiers in a font and instead to define regular rules to define these correspondences. Simons demonstrated that these rules can handle a wide variety of rendering problems and proposed that "a generalized implementation of writing systems would allow users to describe new writing systems by expressing the mapping from characters to graphs as rewrite rules. The computer could take over from there to compile that description into a very efficient implementation as a finite state transducer" (1989:545).[17]

Standards for representing units of writing in a computer do not always match the linguistic analysis of the abstract characters. An encoding may split a single character into two code points or there may be more than one way to represent the same character. In Unicode, the LATIN SMALL LETTER E WITH ACUTE can be stored at code point U-00E9 or by a sequence of code points U-0065 U-0301 representing LATIN SMALL LETTER E and COMBINING ACUTE ACCENT, respectively. Likewise, an encoding may combine two characters into a single code point. Some characters may need to rely on context to determine the proper encoding.

---

17 A finite state transducer is one of the most simple and efficient computer algorithms. SIL International's work on a rendering system called *Graphite* (Correll 2000) promises to bridge the gap and allow these rules to operate in real situations.

Because the relationships are not simple one-to-one relationships (Haas 1983:19, Simons 1989:538) or even one-to-many relationships, a more complex and powerful system of representing the relationships must be devised.

This problem of representing relationships is not new to linguistic study. Phonologies and grammars make extensive use of rules to describe the relationships inherent within their systems. Rules have been used extensively in the study of writing systems as well to represent the relationships between characters and linguistic elements (Haas 1983, Derwing, Priestly and Rochet 1987, Sproat 2000, Simons 1989).

In the descriptive framework developed here, the mapping from one set of elements to another is defined by a set of correspondence rules. Correspondence rules differ from traditional generative phonological rules in two notable ways. First, traditional generative phonological rules are unidirectional (i.e., the rule specifies the relationship from one element to another but not vice-versa), but correspondence rules are bi-directional (i.e., the rule specifies the relationship from one element to another and vice-versa). Second, traditional generative phonological rules are sequentially ordered rules (i.e., each rule applies in sequence and the result of the operation becomes the input to the subsequent rule), but correspondence rules operate in parallel and there are no intermediate stages.

Correspondence rules are two-level rules. The benefits of two-level rules are summarized in Antworth (1990):

> Because two-level rules express a correspondence rather than rewrite symbols, they apply in parallel rather than sequentially. Thus, no intermediate levels of representation are created as artifacts of a rewriting process. Only the lexical and surface levels are allowed. It is this aspect of their nature that is emphasized by the name “two-level” rules. Furthermore, because the two-

> level model is defined as a set of correspondences between lexical and surface representation, two-level rules are bi-directional. (pages 28–29)

Thus, by using correspondence rules which function as two-level rules, we are able to express a single correspondence rule between the linguistic units and graphs. Thus, given the graphs we can determine the corresponding linguistic units or given the linguistic units we can determine the corresponding graphs. This is possible because two-level rules can describe both levels in the environments.

Correspondence rules apply whenever their structural descriptions are met. Those rules with more specific structural descriptions apply with greater precedence. A longer match sequence is considered to be a more specific structural description. Thus, a match sequence of ⟨aa⟩ is more specific than ⟨a⟩. If this were not the case, the match sequence of ⟨aa⟩ would never be handled since a string with ⟨aa⟩ also matches the match sequence ⟨a⟩.

⟨d⟩ ≅ /^{n}d/
⟨dr⟩ ≅ /^{n}r/

Figure 41 Precedence of correspondence rules

In Figure 41, the second rule has a more specific structural description than the first rule. The operator ≅ means “corresponds to” and indicates a correspondence relation between the items that precede it and those that follow. The first rule is essentially shorthand for stating that ⟨d⟩ when followed by anything other than ⟨r⟩ corresponds to the phoneme /^{n}d/.

⟨b⟩ ≅ /b^{h}/ / __ ⟨a⟩
⟨b⟩ ≅ /b/ / __ ⟨aa⟩

Figure 42 Precedence of correspondence rules with environments

When the match sequences are identical, a sequence with a longer environment is considered to be more specific. In Figure 42, the second rule, which matches ⟨b⟩ when it is in the environment where it is followed by ⟨aa⟩ is more specific than the first rule which matches ⟨b⟩ when it is in the environment where it is followed by ⟨a⟩. Again, this is necessary for the longer environments to be matched at all. When two rules have the same structural descriptions, they may be collapsed into a single rule and may be an instance of free variation.

Antworth defines four distinct operators on two-level rules which are necessary for the low level hand compiling that must be done for PC-KIMMO (1990:32–34). However, all these operators may be inferred from the correspondence rule notation we describe without the added complexity of determining which of the operators to use and how to use them in conjunction with each other.[18]

Derwing, Priestly and Rochet (1987) indicate that rules should be "reasonably concise", "non-redundant", and achieve "a level of *systematicity and homogeneity of description*" (page 32). They also should "achieve … a level of *comprehensiveness of coverage* and *attention to details*" (page 32). These are laudable goals but ones which must

---

18 The operator ⇒ "only but not always" is used when the correspondence (regardless of the environment) exists only once in the entire rule-set. This can be determined by a search of the entire rule-set.

The operator ⇐ "always but not only" is used when the underlying form in the given environment exists only once in the entire rule-set. This too can be determined by a search of the entire rule-set.

The operator ⇔ "always and only" is the combination of the operators ⇐ and ⇒. This is true when the conditions for both ⇐ and ⇒ are met.

The operator /⇐ "never" is used to cover exceptions to a more general rule. I have an exception clause within the rule to cover exceptions so this can also be inferred.

be self imposed by the analyst and not by the descriptive framework. However, the descriptive framework should not limit the ability to produce such rules.

Haas requires his "glotto-graphic translation" rules to "(i) identify the items that are matched and (ii) state the contextual conditions of that match" (1983:19). His statement can be generalized to apply to any correspondence rule. Thus, a rule has three components: (1) the initial state, and (2) the final state, corresponding to Haas' "items that are matched" (1983:19), and (3) the context or environment in which the rule applies. For two-level rules, there is no initial or final state *per se*, since the rules are bi-directional. Instead, the initial state is one member of the correspondence and the final state is the other. This correspondence has an environment which in turn is a set of correspondences.

In a correspondence rule, the elements that correspond are indicated by a *correspondence* element and the environment is designated by the *when* element. When there are multiple environments, the correspondence is considered to apply if any of those environments apply.

The correspondence between Greek /s/ and *sigma* mentioned previously in the section on graphemes is as in Figure 43, where word final /s/ corresponds to ⟨ ⟩ and /s/ corresponds to ⟨ ⟩ in all other environments. This is formally represented in XML notation in Figure 44.

s ≙ ς / __ #
s ≙ σ

Figure 43 Sigma correspondence rule

```
<rule>
  <correspondence>
    <linguistic-unitRef target="phoneme-s"/>
    <graphRef target="final-sigma"/>
  </correspondence>
  <when>
    <followed-by>
      <classRef target="word-end"/>
    </followed-by>
  </when>
</rule>

<rule>
  <correspondence>
    <linguistic-unitRef target="phoneme-s"/>
    <graphRef target="sigma"/>
  </correspondence>
</rule>
```

Figure 44 Formal sigma correspondence rule

Notice that this rule also sets up a constraint for the occurrence of the final-sigma character form. It can only occur before a character in the class word-end. It does not constrain the occurrence of the phoneme /s/ since it occurs without constraint. When there is no environment specified in a correspondence rule, the environment does not affect the correspondence.

The rules in Figure 44 are not ordered, but the first rule has precedence over the second since it has a more specific structural description.

The formal declaration for writing system units does not seek to establish any constraints on the occurrence of certain sequences of characters. The set of these constraints are called the graphotactics of the language. “Each language has graphotactic formulas which state the combinations of graphemes which may occur in that language” (Herrick 1974:10). The graphotactics are specified as part of the correspondence rules.

Sometimes we want to allow a choice between different units, or to define optional or repeatable units. This is accomplished with the grouping elements: *sequence* and *choice* and

with the elements *optional* and *repeatable*. *Sequence* defines a sequence of units or groups, while *choice* defines a choice between units or groups. The content of *optional* is optional and there is no limit to the number of occurrences of the content of *repeatable*. Groups can also exclude units. This is done by using the optional *excluding* element.

Figure 45 demonstrates an example sequence that is optional and contains the graphs ⟨m⟩ and ⟨b⟩.

```
<optional>
  <sequence>
    <graphRef target="m"/>
    <graphRef target="b"/>
  </sequence>
</optional>
```

Figure 45 Formal instance of an optional sequence

Figure 46 demonstrates an example choice that is required but may occur as many times as desired. The choice is between the graphs ⟨p⟩, ⟨b⟩, and ⟨m⟩.

```
<repeatable>
  <choice>
    <graphRef target="p"/>
    <graphRef target="b"/>
    <graphRef target="m"/>
  </choice>
</repeatable>
```

Figure 46 Formal instance of a repeatable choice

An example rule from (Carney 1994:300) is presented in Figure 47. This is an English spelling-to-sound rule that specifies that the letter ⟨b⟩ corresponds with no sound when it comes after the letter ⟨m⟩ and before a word boundary. The letter ⟨e⟩ may optionally intervene between the ⟨b⟩ and the word boundary. The corresponding formalism is presented in Figure 48.

B.2 b ≡ ∅ | m — (e) #

Figure 47 English graph to sound correspondence rule

```
<mapping-set cor1="graph" cor2="linguistic-unit>
  ...
  <rule>
    <correspondence>
      <graphRef target="b"/>
      <null/>
    </correspondence>
    <when>
      <preceded-by>
        <graphRef target="m"/>
      </preceded-by>
      <followed-by>
        <sequence>
          <optional>
            <graphRef target="e"/>
          </optional>
          <classRef target="word-end"/>
        </sequence>
      </followed-by>
    </when>
  </rule>
  ...
```

Figure 48 Formal equivalent of English graph to sound correspondence rule

The *null* in the formalism is no linguistic unit as this rule is a member of the *mapping-set* which define the correspondences between graphs and linguistic units.

Here are more cases of rules and how they would be represented in the formalism. Derwing, Priestly and Rochet (1987) have successfully created a set of rules for Russian, American English, and French which can produce phonetic forms from the orthographic forms. It should be noted, though, that many of these rules "depend on specific boundary-symbols, which in turn are defined (in part) with reference to grammatical entities (namely, prefixes and prepositions)" (1987:46). Thus, the rules that describe the relationships of graphic elements to linguistic elements may depend on other linguistic information.

> For French, a number of rules are sensitive to grammatical information. Thus, the sequence *-ent* in word final position represents schwa if it is the 3rd

person plural verb marker, or the nasal vowel [ã] in all other cases, as indicated in the following ordered rules: (Derwing, Priestly and Rochet 1987:47)

R 22 ent → ə / 3rd person verb marker
R 23 en → ã / —C#

Figure 49 French correspondence rule referencing grammatical information

This can be formally represented as shown in Figure 50.

```
<rule>
  <correspondence>
    <sequence>
      <graphRef target="e"/>
      <graphRef target="n"/>
      <graphRef target="t"/>
    </sequence>
    <linguistic-unitRef type="phone-schwa"/>
  </correspondence>
  <when>
    <in>
      <linguistic-unitRef target="morpheme-3p"/>
    </in>
  </when>
</rule>
<rule>
  <correspondence>
    <sequence>
      <graphRef target="e"/>
      <graphRef target="n"/>
    </sequence>
    <linguistic-unitRef type="phone-atilde"/>
  </correspondence>
  <when>
    <followed-by>
      <sequence>
        <classRef target="C"/>
        <classRef target="word-end"/>
      </sequence>
    </followed-by>
  </when>
</rule>
```

Figure 50 Formal equivalent of French correspondence rule

Figure 51 contains the default rule for mapping the Burmese character *great ka* to a phoneme by means of a simple one-to-one mapping.

```
<rule>
  <correspondence>
    <graphRef target="ka"/>
    <linguistic-unitRef type="phone-k">
  </correspondence>
</rule>
```

Figure 51 Formal instance of simple correspondence rule

Great ka in conjunction with a following *supporting ya* or *encircling ra* (medial forms of *supine ya* and *crooked ra*, respectively), correspond to a palatal stop. This rule is shown in Figure 52

```
<rule>
  <correspondence>
    <sequence>
      <graphRef target="ka"/>
      <choice>
        <graphRef target="m-ya"/>
        <graphRef target="m-ra"/>
      </choice>
    </sequence>
    <linguistic-unitRef type="phone-c">
  </correspondence>
</rule>
```

Figure 52 Formal instance of correspondence rule with context

In Burmese, certain characters take on different forms depending on their context. In Figure 53, the grapheme *(line) drawn down (yei: hca.)* has a default rendering, but when preceded by a consonant that makes this rendering ambiguous, a longer form of *(line) drawn down (long yei: hca.)* is used instead.

```
<rule>
  <correspondence>
    <graphemeRef target="aa"/>
    <graphRef target="yei:-hca."/>
  </correspondence>
</rule>

<rule>
  <correspondence>
    <graphemeRef target="aa"/>
    <graphRef target="long-yei:-hca."/>
  </correspondence>
  <when>
    <preceded-by>
      <classRef target="long-aa-C"/>
    </preceded-by>
  </when>
</rule>
```

Figure 53 Formal instance of correspondence rule

7.3 Relations

A description may need to indicate how one set of writing system elements are related to another set of the same type of writing system elements. For example, many Roman based orthographies need to indicate the capitalized forms of the lower case letters and vice versa. These are two sets of graphs which are in a specified relation. A similar correspondence rule is used to accomplish these relations. The distinction is that relation rules are contained within a *relation-set* element which indicates the features which are embodied in the relation.

The upper case equivalent of the Turkish lower case dotless i ⟨ı⟩ is ⟨I⟩ and ⟨i⟩ is the lowercase equivalent of ⟨İ⟩. The features involved in the relation are lower case and upper case. These correspondences are represented formally in Figure 54.

```
<relation-set feature1="lower case" feature2="upper case">
  ...
  <rule>
    <correspondence>
      <graphRef target="idotless"/>
      <graphRef target="I"/>
    </correspondence>
  </rule>
  <rule>
    <correspondence>
      <graphRef target="i"/>
      <graphRef target="Idot"/>
    </correspondence>
  </rule>
  ...
</relation-set>
```

Figure 54 Case relation rules

Given the information in the definition of the lower and upper case relation, a program could transform the lower case forms into upper case or vice versa.

7.4 Collating sequence

Herrick recognizes what he calls an “alphabetic order” whereby “normal literates who are given a group of written words (or other strings of characters that embody letters) can arrange them according to this alphabetic order” (1974:10). This ordering relationship is not confined to alphabets. Any language with a dictionary has the entries in some order defined by the writing system. A computer will need access to this information if it is to perform the task of sorting. Thus, characters may be ordered relative to each other within the writing system. In computer terminology, the order is called a *collating sequence*

Linguists have made control over the sorting sequence a requirement for software programs:

> Sorting a data set according to a particular criterion groups all the bits of data that go together, thereby allowing a pattern to emerge. As noted above in

> connection with database programs, a sorting program for linguistic use must permit the user complete control over definition of the alphabet and sorting sequence. (Antworth and Valentine 1998:174)

At first, it would seem as though we could list all the characters in the order in which they should occur. If we did this for American English, we would arrive at the list started in Figure 55.

```
a A b B c C d …
```

Figure 55 Simple collation sequence

A close analysis of this treatment of capitalization shows that ⟨brown⟩, ⟨Brown⟩, ⟨born⟩ and ⟨Born⟩ would be sorted as in Figure 56.

```
born
brown
Born
Brown
```

Figure 56 Words in alphabetical order separated by case

This may be a useful sort for some purposes, but the general practice for sorting American English is for these words to be sorted as in Figure 57, so that words which differ only in their capitalization are sorted next to each other.

```
born
Born
brown
Brown
```

Figure 57 Words in alphabetical order with case as second level

To get these results, we cannot simply use a single list. The typical solution to this problem has been to create secondary sorting keys. The first sorting key sets up the order of a series of equivalent units. When the comparison does not involve two units which are

deemed to be equivalent, only the first level is required. However, if the comparison involves two units that are equivalent at the first level, the second sorting key applies in like fashion. We can make lists of sorting orders to designate the sorting keys at each level of the sorting algorithm. Capitalization becomes a secondary sorting key. The case distinction is not factored in until the second level as in Figure 58.

```
level 1: (a=A) (b=B) (c=C) (d=D) …
level 2: a A b B c C d D …
```

Figure 58 Levels of sorting keys

This relationship can actually be captured by a collation tree as in Figure 59.

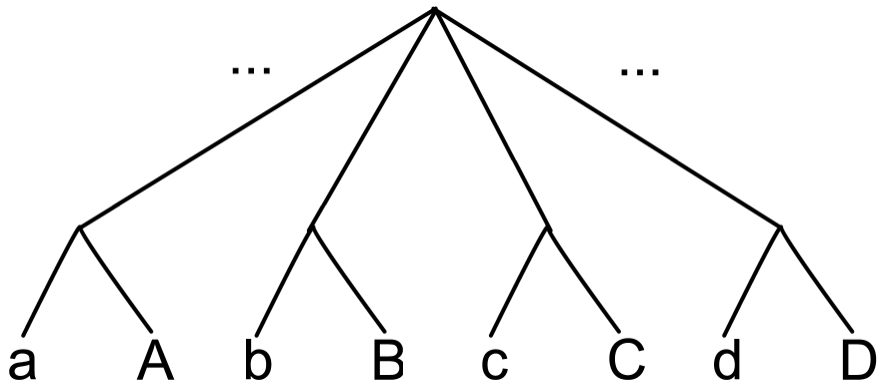

Figure 59 Collation tree

The first node in the tree contains the list of sequences at level one. The children of any node are considered to be equal to each other at the level of the common ancestor.

Tree structures can be described in terms of nested sequences. In fact, nested sequences are exactly how the tree structure inherent within every XML document is represented. Thus, by using nested collating sequences, we are really defining a collating tree. Figure 59 is represented formally in Figure 60

```
<collating-sequence>
  <collating-sequence>
    <graphRef target="a"/>
    <graphRef target="A"/>
  </collating-sequence>
  <collating-sequence>
    <graphRef target="b"/>
    <graphRef target="B"/>
  </collating-sequence>
  <collating-sequence>
    <graphRef target="c"/>
    <graphRef target="C"/>
  </collating-sequence>
  <collating-sequence>
    <graphRef target="d"/>
    <graphRef target="D"/>
  </collating-sequence>
  …
<collating-sequence>
```

Figure 60 Formal collation tree

Diacritics, hyphens, and apostrophes must usually be treated in a similar manner, though other layers may have to be defined to get the appropriate results. Each layer applies only if preceding layers have been unable to determine the sorting order. In American English, accented characters must be handled in the second layer and capitalization in the third layer to achieve appropriate results (e.g., ⟨resume⟩ precedes ⟨Resume⟩ which precedes ⟨résumé⟩ which precedes ⟨Résumé⟩ (Unicode Consortium 2000:137). The three layers are illustrated in Figure 61. The outermost layer distinguishes between the base characters, regardless of case or diacritics. The second layer distinguishes between diacritics, regardless of case, while the innermost layer makes the case distinction.

```
<collating-sequence>
  …
  <collating-sequence>
    <collating-sequence>
      <graphRef target="e"/>
      <graphRef target="E"/>
    </collating-sequence>
    <collating-sequence>
      <graphRef target="eacute"/>
      <graphRef target="Eacute"/>
    </collating-sequence>
  </collating-sequence>
  …
<collating-sequence>
```

Figure 61 Level precedence in a collation tree

In addition to the different levels of sorting keys, *The Unicode Standard* (2000) indicates other challenges for sorting. “In French sorting, the differences in accents later in the string are more important than those earlier in the string” (page 138). In French, characters with no accents sort before those with accents, and characters with the acute accent ⟨ ´ ⟩ sort before characters with the circumflex accent ⟨ ^ ⟩. Thus, ⟨ pécher ⟩ precedes ⟨ pêcher ⟩. However, the differences in accents are compared on the final characters first, so ⟨ pêche ⟩ precedes ⟨ péché ⟩. This can only be accounted for by indicating that the secondary layer has reverse precedence (from the end of the string to the start instead of from the start to the end). Each level in the tree indicates how the string should be processed to determine the correct precedence. The default is start to end processing.

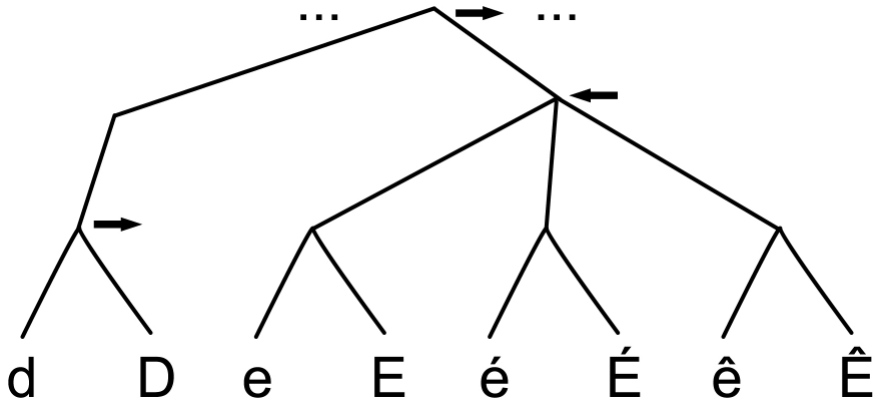

Figure 62 Collating tree for French

Figure 63 illustrates the formal notation for the orderings noted in Figure 62.

```
<collating-sequence>
  …
  <collating-sequence direction="end-to-start">
    <collating-sequence>
      <graphRef target="d"/>
      <graphRef target="D"/>
    </collating-sequence>
  </collating-sequence>
  <collating-sequence direction="end-to-start">
    <collating-sequence>
      <graphRef target="e"/>
      <graphRef target="E"/>
    </collating-sequence>
    <collating-sequence>
      <graphRef target="eacute"/>
      <graphRef target="Eacute"/>
    </collating-sequence>
    <collating-sequence>
      <graphRef target="ecirc"/>
      <graphRef target="Ecirc"/>
    </collating-sequence>
  </collating-sequence>
  …
<collating-sequence>
```

Figure 63 Formal collating tree for French

Since the direction is an attribute of each node in the tree and not a separate level indicator, the attributes of all nodes at a particular level must agree or it is an error.

Another challenge is to allow characters to be ignored at a particular level such that "the character itself is ignored unless there are no stronger differences in the string" (Unicode Consortium 2000:138) . Such a case is illustrated in Figure 64.

```
blackbird
black-bird
blackbirds
black-birds
```

Figure 64 Ignorable characters

In order to handle the special case of ignorable characters, we need to indicate that the characters are ignored at a particular level. The character is placed in the correct position within the tree and then branches are created and labeled as ignored for the number of levels that are necessary before it would be significant. When a branch is labeled as ignored, its parent branch must also be labeled as ignored. This is demonstrated in Figure 65.

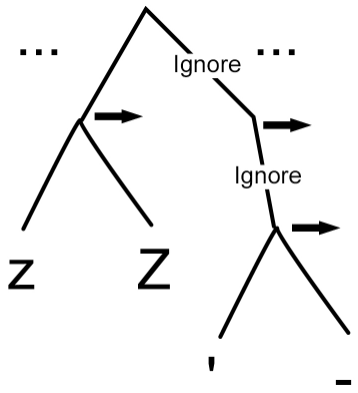

Figure 65 Collating tree with ignored characters

In Figure 65, there is one more ignored level than the final level of distinction found so that the ignored character will only be significant after all other levels have been evaluated. When there is an apostrophe, it will occur before the hyphen but they are ignored

unless there are no other distinctions found. The formal equivalent of Figure 65 is illustrated in Figure 66.

```
<collating-sequence>
  …
  <collating-sequence>
    <graphRef target="z"/>
    <graphRef target="Z"/>
  </collating-sequence>
  <collating-sequence ignore="yes">
    <collating-sequence ignore="yes">
      <graphRef target="apostrophe"/>
      <graphRef target="hyphen"/>
    </collating-sequence>
  </collating-sequence>
  …
<collating-sequence>
```

Figure 66 Formal ignorable characters

It is quite possible that I am erring on the side of general expressive power here. If all languages which use ignored characters treat them as in English where there is no other distinction that can be made before ignored characters are sorted, then there would be no need to include all the levels in an ignore branch. However, if a language exists (and I know of none) where a case distinction is made only after the ignored characters are factored in, then the level at which the ignored characters apply is important and the current expressive power is required.

"In traditional German, *ö* sorts as if it were *oe*, putting it after *od* and before *of*" (Unicode Consortium 2000:138). This ordering cannot be achieved using a simple listing of orders (e.g. ⟨o⟩ ⟨ö⟩ ⟨p⟩) to obtain the correct result. To achieve the correct ordering of ⟨ö⟩ using a list of characters would require us to treat ⟨od⟩ and ⟨of⟩ as characters. Not only does this force us into an unnatural analysis of the data, but all combinations of ⟨o⟩ and the following letter must be treated as characters in order to achieve the correct result. Taking a

cue from the verbal description, we can, instead, equate it to ⟨oe⟩ as a preprocessing step. This is the way that a German conceives of that segment as well. Figure 67 illustrates the formal notation for designating an equivalency for sorting purposes.

```
<ordering>
  <name>Alphabetic Order</name>
  <sort-equivalencies>
    <sort-equivalency>
      <graphRef target="oumlaut"/>
      <sequence>
        <graphRef target="o"/>
        <graphRef target="e"/>
      </sequence>
    </sort-equivalency>
  </sort-equivalencies>
  <collating-sequence>
    …
  <collating-sequence>
</ordering>
```

Figure 67 Formal sort equivalencies

The need for sort equivalencies extends into realms beyond the simple need to equate certain characters with others. Whole strings of characters may need to be substituted by another string. For example, when sorting English surnames, ⟨Mc⟩ as a prefix is usually sorted as though it were ⟨Mac⟩.

In Japanese Katakana, the length mark (ー), which "lengthens the vowel of the preceding character" (Unicode Consortium 2000:139) is sorted based on the previous character. "For example, after the character カ 'ka', the ー *length mark* indicates a long 'a' and comes after ア 'a'; after the character キ 'ki', the ー *length mark* indicates a long 'i', and comes after イ 'i'" (Unicode Consortium 2000:139). Thus, ⟨カア⟩ precedes ⟨カー⟩, which precedes ⟨カイ⟩, and ⟨キア⟩ precedes ⟨キイ⟩, which precedes ⟨キー⟩.

In order to handle this case, we need to extend our means of designating sort orders to include environmental conditions. Thus, a character may occur multiple times within the collating tree, but each instance must have a distinct environment in which it would occur at that position. This leads us to the analysis in Figure 68

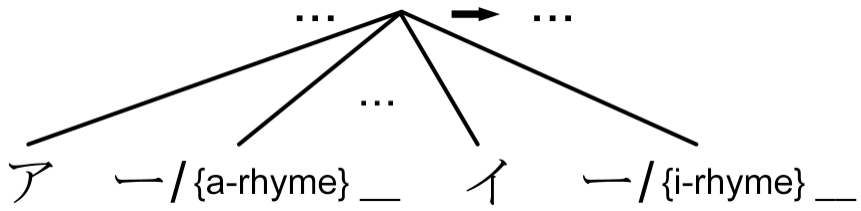

Figure 68 Katakana collation tree

The formal version of this collation tree is shown in Figure 69.

```
<collating-sequence>
  …
  <graphRef target="a"/>
  <context-graphRef target="length-mark">
    <when>
      <preceded-by>
        <classRef target="a-rhyme"/>
      </preceded-by>
    </when>
  </context-graphRef>
  …
  <graphRef target="i"/>
  <context-graphRef target="length-mark">
    <when>
      <preceded-by>
        <classRef target="i-rhyme"/>
      </preceded-by>
    </when>
  </context-graphRef>
  …
</collating-sequence>
```

Figure 69 Formal Katakana ordering rules

Chapter 8: From formalism to publication

The formalism presented in the previous chapters is the notation that would make it possible for a computer system to read and use a writing system description. But what of the second audience I am addressing, the human readers? This chapter describes how the system meets their needs as well.

8.1 Elements needed for publication

If we are to make the description available for users in a publishable form, we need to address the elements that are required beyond the strictly formal elements of writing system description. Basic information relevant to any description of a language must also be included. I address each of these in turn.

8.1.1 Language

The language element identifies the language that the writing system description is about. This is used to form the title of the work.

The language may be identified by name. It may also be identified by means of codes that uniquely identify the particular language. This may be from the SIL Ethnologue or another standard like ISO 639. For example, American English is identified in Figure 70.

```
<language>
  <code issuer="Ethnologue">ENG</code>
  <code issuer="RFC-1766">en-US</code>
  <name>American English</name>
</language>
```

Figure 70 Formal instance of language identification

8.1.2 Author

An authorship statement includes the name of the author, the author's affiliation, and contact information. There may be any number of authors associated with the work. Figure 71 shows the information associated with a typical authorship statement.

```
<author>
  <fullname>Eric Albright</fullname>
  <affiliation>SIL International</affiliation>
  <contact>eric_albright@sil.org</contact>
</author>
```

Figure 71 Formal instance of author

8.1.3 Prose sections

The XML document model contains a number of sections which structure the publication. These are

- introduction
- history
- illustrative-text
- linguistic units
- graphs
- graphemes
- classes

- writing-units
- constraints
- orderings
- relationships
  - mapping-set
  - relation-set
- issues
- conclusion
- references

The introduction introduces the writing system context. This should include information about the people who use it. The history is an optional section which introduces the history of the development of the writing system. It may also be incorporated into the introduction section. The issues section addresses any known issues that have arisen with the writing system as it stands. Problem areas should be addressed in this section. The illustrative text section provides a text written in the orthography along with a translation. The conclusion is a section for summing up and drawing any conclusions. The references are gathered into a section at the end.

Although not all of these sections are required for every publication, they have been made available to provide consistency between descriptions. The formal definitions of constituents and rules are included within the appropriate sections. Each formal declaration may contain a prose description to explain the formalism in prose. The content of the prose sections has been left up to the user so that the author is not bound to a particular set of elements. Unless specialized elements are required, however, the ones defined by HTML are

sufficient for most uses. Figure 72 shows the beginning of a description for the Fijian writing system. It includes an introduction as well as a brief history of the creation of the writing system.

```
<EWSD xml:lang="en-US" name="Fijian" date="2001-04-16">
 <language>
  <code issuer="Ethnologue">FJI</code>
  <name>Fijian</name>
 </language>

 <author>
  <fullname>Eric Albright</fullname>
  <affiliation>SIL International</affiliation>
  <contact>eric_albright@sil.org</contact>
 </author>

 <introduction>
  <p>Fijian is not a very complicated writing system, being
   simply phonemic. However, although Fijian uses letters
   which are familiar to users of Roman alphabets, some of
   these letters do not have familiar correspondences to the
   sound system and thus may surprise the uninitiated.</p>
  <p>The Fijian writing system is detailed in this
   description. Special note is made of the correspondences
   between the graphic symbols and the phonemes they
   represent.</p>
  <p>The Fijian writing system has been described as
   <q>a model of consisitency and simplicity…—in
   spite of its peculiar use of the letters
   <transcription type="orthographic">b
```

Prose discussion, which may follow any of the formal components is illustrated in Figure 73.

```
<class id="loan">
 <name>Loan</name>
 <graphRef target="f"/>
 <graphRef target="F"/>
 <graphRef target="j"/>
 <graphRef target="J"/>
 <graphRef target="p"/>
 <graphRef target="P"/>
 <discussion>
  <p>The letters
   <transcription type="orthographic">f
```

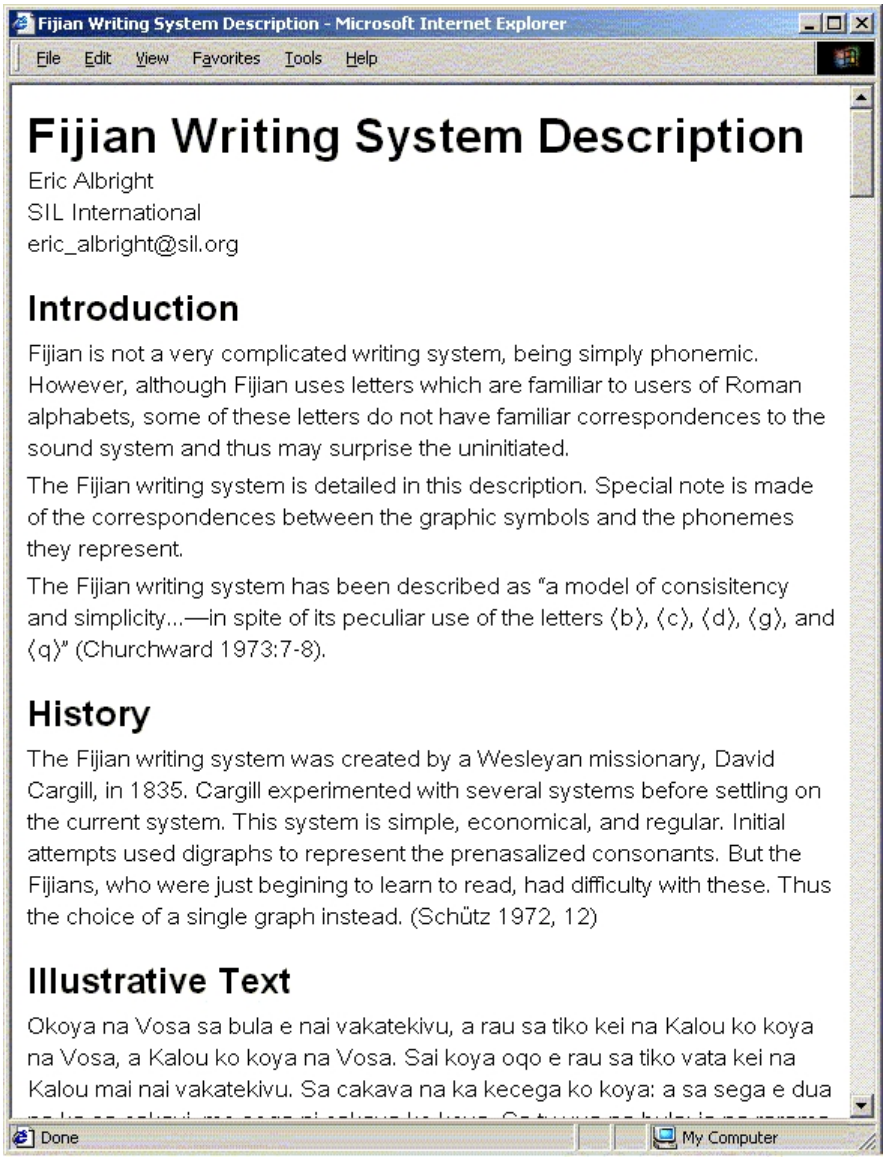

Figure 74 Rendering of Fijian writing system description introduction

Figure 75 shows a rendering of Figure 73.

Loan ∋ f, F, j, J, p, P

The letters ⟨f⟩, ⟨j⟩, and ⟨p⟩ and their capitalized equivalents represent sounds that are not Fijian and are only used in Fijianized loan words from English.

Figure 75 Rendering of prose discussion

Since the function of the formal components has been clearly delimited, changes in the format are simple using a stylesheet. Thus, many stylesheets could be created to suit the needs of different audiences.

8.3 Implementation of example

The implementation of this approach to writing system descriptions is documented in three appendixes. Chapter gives the complete DTD for the electronic writing system descriptions described in this thesis. Chapter provides excerpts from a description of the Fijian writing system. In the interest of saving space, when large segments of the example seem to repeat a similar structure, a single instance of that structure is given along with ellipses to indicate where segments have been omitted. This is the document used to produce Figure 74. Chapter contains a complete XSLT stylesheet for rendering electronic writing system descriptions that conform to the DTD in Chapter to the HTML format used by Internet browsers. This is the stylesheet used to produce Figure 74.

Chapter 9: Conclusions

9.1 The best descriptions

This thesis has demonstrated the design of a formal framework for describing any writing system by generalizing the descriptive process for writing systems. It seeks to improve the descriptions of writing systems and to increase the availability of such descriptions.

The best linguistic descriptions are meaningful. That is, they are accessible to the audience. This formal framework seeks to provide the ability to provide writing system descriptions which are accessible to computational processes as well as to people. The formal components address the concerns of computational processes, while the descriptive prose elements work well for the linguists and other interested users.

The best linguistic descriptions are also accurate and comprehensive in that they completely account for all the known data. Formal descriptions are much more difficult to write than informal descriptions, for one must think of all the possibilities and leave no ambiguity. However, they also force one to account for all the data and to think in terms that are more precise. Thus, formal descriptions tend to be more comprehensive.

9.2 Results

An XML DTD has been produced to formalize this framework and to provide validation of electronic writing system descriptions that follow the framework. Structured editors for XML documents use the DTD to aid the creator.

The XSLT stylesheet that is provided proves the concept that writing system descriptions in this formal electronic format can be made available to humans in an understandable form. However, it is rudimentary and requires feedback from potential users to make it into a true production-worthy stylesheet.

9.3 Wider implications

The framework for writing formal rules could be used in many other linguistic descriptions. Thus, this thesis breaks new ground for formal linguistic descriptions of all types which use rules.

There are many kinds of linguistic description that could benefit from having formal models for electronic forms. For instance, an electronic description of the phonology and an electronic version of the lexicon are needed. Until such descriptions are made, the true potential of a network of linguistic information cannot be reached.

9.4 Recommendations

This framework has been designed with the intention that software would be written to elicit the information concerning writing systems from an expert so that the complexity of this framework would be made transparent, perhaps through a series of questions whose

answers are put into the appropriate spots within the description. Such software should be written to provide maximum ease in the information description process.

As for the computer functionality, a library of functions that can be used by any program to gain access to this information will need to be written and made available for widespread implementation of this method in computer applications.

Appendix A: Electronic writing system description document type definition

```
<!-- Electronic Writing System Descriptions
     Document Type Definition
     Written by Eric Albright
     This DTD is XML 1.0 Compliant

     Copyright (c) 2000-2001 Eric S. Albright. Permission to copy
     and to distribute is hereby granted provided that this
     notice is included in each copy.

     Base model.
                                                                  -->

<!ENTITY % INHERITED '#IMPLIED'                                    >
<!ENTITY % HREF      'CDATA'                                       >

<!-- ISO dates have the format YYYY-MM-DD                        -->
<!ENTITY % ISO-date  'CDATA'                                       >
<!ENTITY % Number    'CDATA'                                       >
<!ENTITY % Language  'CDATA'                                       >

<!ENTITY % group 'choice | sequence | catenate'                    >

<!ENTITY % units ' %group; |
           graphemeRef | graphRef |
           writing-unitRef | classRef |
           linguistic-unitRef |
           glyph | coded-unit | key-stroke |
           null | any'                                            >


<!-- Electronic Writing System Description (EWSD) -->
<!ELEMENT EWSD               (head?,
                              language,
                              author*,
                              introduction?,
                              history?,
                              illustrative-text?,
                                graphs,
                                graphemes?,
                                classes?,
                                writing-units?,
                                linguistic-units?,
                                relationships?,
                                orderings?,
                              issues?,
                              conclusion?,
                              references?)                         >
<!ATTLIST EWSD
```

```
          xml:lang          %Language;          #REQUIRED
          name              CDATA               #REQUIRED
          date              %ISO-date;          #REQUIRED      >
<!--
          name
                the name which will be used as the identifier.
                Conforming applications may associate the
                value of a lang or xml:lang attribute in the
                document with this name to determine the
                electronic writing system description to be
                referenced.
          date
                date last modified, may serve to determine
                the version of this electronic writing system
                description                                    -->

<!-- The m-block entity should be redefined to include block
      level elements to be used in description.
     The m-reference entity should be redefined to include those
     elements that need to be identified in references.
-->
<!ENTITY % m-block 'ANY' >
<!ENTITY % m-inline 'ANY' >
<!ENTITY % m-reference 'ANY'>

<!ELEMENT language          (code*, name+)                     >
<!--            multiple codes allow for iso639, IANA,
                SIL Ethnologue, or other standard language
                codes.
                multiple names allow names to be written in
                different languages or with different types. -->
<!ATTLIST language
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT code               (#PCDATA)                         >
<!ATTLIST code
          issuer             CDATA              #REQUIRED      >
<!--
          issuer
                the name of the official registration body   -->

<!ELEMENT name               (#PCDATA)                         >
<!ATTLIST name
          xml:lang          %Language;          %INHERITED;
          type              CDATA               #IMPLIED       >

<!ELEMENT author            (fullname, affiliation?, contact?)  >
<!ATTLIST author
          xml:lang          %Language;          %INHERITED;    >
```

```
<!ELEMENT fullname          (#PCDATA)                          >
<!ATTLIST fullname
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT affiliation       (#PCDATA)                          >
<!ATTLIST affiliation
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT contact           (#PCDATA)                          >
<!ATTLIST contact
          xml:lang          %Language;          %INHERITED;    >


<!ELEMENT introduction      (head?, (%m-block;)*, section*)    >
<!ATTLIST introduction
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT head              (#PCDATA | %m-inline;)*            >
<!ATTLIST head
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT section           (head, (%m-block;)*, section*)     >
<!ATTLIST section
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT history           (head?, (%m-block;)*, section*)    >
<!ATTLIST history
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT issues            (head?, (%m-block;)*, section*)    >
<!ATTLIST issues
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT illustrative-text (head?, (%m-block;)*, section*)    >
<!ATTLIST illustrative-text
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT conclusion        (head?, (%m-block;)*, section*)    >
<!ATTLIST conclusion
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT references        (head?, reference+)                >
<!ATTLIST references
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT reference         %m-reference;                      >
<!ATTLIST reference
          xml:lang          %Language;          %INHERITED;    >

<!ELEMENT discussion        (%m-block;)*                       >
```

```
<!ATTLIST discussion
          xml:lang          %Language;           %INHERITED;    >


<!ELEMENT graphs            (head?, introduction?, graph+)      >
<!ATTLIST graphs
          xml:lang          %Language;           %INHERITED;    >

<!ELEMENT graph             (name+, discussion?)                >
<!ATTLIST graph
          id                ID                   #REQUIRED
          image             %HREF;               #IMPLIED       >

<!ELEMENT graphRef          EMPTY                               >
<!ATTLIST graphRef
          target            IDREF                #REQUIRED      >

<!ELEMENT glyph             EMPTY                               >
<!ATTLIST glyph
          name              CDATA                #IMPLIED
          number            %Number;             #IMPLIED
          font              CDATA                #REQUIRED
          img               %HREF;               #IMPLIED       >

<!ELEMENT graphemes         (head?, introduction?, grapheme+)   >
<!ATTLIST graphemes
          xml:lang          %Language;           %INHERITED;    >

<!ELEMENT grapheme          (name+, discussion?)                >
<!ATTLIST grapheme
          id                ID                   #REQUIRED
          image             %HREF;               #IMPLIED       >

<!ELEMENT graphemeRef       EMPTY                               >
<!ATTLIST graphemeRef
          target            IDREF                #REQUIRED      >

<!ELEMENT classes           (head?, introduction?, class+)      >
<!ATTLIST classes
          xml:lang          %Language;           %INHERITED;    >

<!ELEMENT class             (name+,
                             (graphRef | context-graphRef |
                              graphemeRef | context-graphemeRef |
                              classRef | context-classRef)+,
                              discussion?)                       >
<!ATTLIST class
          id                ID                   #REQUIRED      >
```

```
<!ELEMENT context-graphRef  (when)                              >
<!ATTLIST context-graphRef
          target            IDREF                #REQUIRED      >

<!ELEMENT context-graphemeRef (when)                            >
<!ATTLIST context-graphemeRef
          target            IDREF                #REQUIRED      >

<!ELEMENT context-classRef  (excluding?, when)                  >
<!ATTLIST context-classRef
          target            IDREF               #REQUIRED       >

<!ELEMENT when              (preceded-by?, followed-by?, in?)   >

<!ELEMENT classRef          (excluding?)                        >
<!ATTLIST classRef
          target            IDREF               #REQUIRED       >

<!ELEMENT writing-units     (head?, introduction?,
                             writing-unit+)                     >

<!ELEMENT writing-unit      (name+, is-made-of, discussion?)    >
<!ATTLIST writing-unit
          id                ID                  #REQUIRED       >

<!ELEMENT writing-unitRef   EMPTY                               >
<!ATTLIST writing-unitRef
          target            IDREF               #REQUIRED       >

<!ELEMENT is-made-of        (%units;)                           >

<!ELEMENT linguistic-units (head?, introduction?,
                            linguistic-unit+)                   >
<!ATTLIST linguistic-units
          xml:lang          %Language;          %INHERITED;     >

<!ELEMENT linguistic-unit    (name+, discussion?)               >
<!ATTLIST linguistic-unit
          type              CDATA               #REQUIRED
          link              %HREF;              #IMPLIED
          id                ID                  #REQUIRED       >

<!ELEMENT linguistic-unitRef EMPTY                              >
<!ATTLIST linguistic-unitRef
          target            IDREF               #REQUIRED       >

<!ELEMENT coded-unit        EMPTY                               >
<!ATTLIST coded-unit
          bits              (8|16|32)           #IMPLIED
          value             %Number;            #REQUIRED       >
```

```
<!ELEMENT key-stroke        EMPTY                                 >
<!ATTLIST key-stroke
          value             CDATA               #REQUIRED       >
<!-- value is a space delimited set of key labels:
These are implementation defined: e.g.
ESCAPE
F1 F2 ... F24
PRINTSCREEN
CONTROL
SHIFT
ALT
A B C ... Z
0 1 ... 9
NUMPAD0 NUMPAD1 ... NUMPAD9
PLUS
MINUS
EQUALS
PERIOD
COMMA
SEMICOLON
APOS
BACKSPACE
INSERT
DELETE
HOME
END
UP
DOWN
LEFT
RIGHT
PAGEDOWN
PAGEUP
SPACE
TAB
-->

<!ELEMENT sequence          ((%units;|optional|repeatable)+)    >

<!ELEMENT context-sequence  ((%units;|optional|repeatable)+,
                             when)                              >

<!ELEMENT catenate          ((%units;|optional|repeatable)+)    >
<!ATTLIST catenate
          type              (left | right |
                             down | up |
                             surround | in)     #REQUIRED       >

<!ELEMENT choice            ((%units;|optional|repeatable)+)    >
```

```
<!ELEMENT optional          (%units; | repeatable)              >
<!ELEMENT repeatable        (%units;)                           >

<!ELEMENT excluding         (%units;)                           >

<!ELEMENT relationships     (head?, introduction?,
                             (mapping-set | relation-set)+)     >

<!ELEMENT mapping-set       (head?, introduction?, name+, rule+)>
<!ATTLIST mapping-set
          cor1              (linguistic-unit |
                             grapheme |
                             graph |
                             glyph |
                             coded-unit |
                             key-code |
                             writing-unit)      #REQUIRED

          cor2              (linguistic-unit |
                             grapheme |
                             graph |
                             glyph |
                             coded-unit |
                             key-code |
                             writing-unit)      #REQUIRED       >


<!ELEMENT rule              (correspondence, when*,
                              discussion?)                      >

<!ELEMENT relation-set      (head?, introduction?, name+, rule+)>
<!ATTLIST relation-set
          feature1          CDATA      #REQUIRED
          feature2          CDATA      #REQUIRED               >

<!ELEMENT correspondence     ((%units;), (%units;))             >

<!ELEMENT preceded-by        (%units; | correspondence)         >

<!ELEMENT followed-by        (%units; | correspondence)         >

<!ELEMENT in                 (%units; | correspondence)         >


<!ELEMENT orderings          (head?, introduction?, ordering+)  >

<!ELEMENT ordering           (head?, introduction?, name+,
                              sort-equivalencies?,
                              collating-sequence)               >
```

```
<!ELEMENT sort-equivalencies (sort-equivalency+)                >
<!ELEMENT sort-equivalency   ((graphRef | graphemeRef |
                               sequence),
                              (graphRef | graphemeRef | sequence
                               | null))                         >
<!ELEMENT collating-sequence (((graphRef | context-graphRef |
                                graphemeRef |
                                context-graphemeRef |
                                sequence | context-sequence |
                                collating-sequence),
                               discussion?)+)                   >
<!ATTLIST collating-sequence
      direction            ( start-to-end | end-to-start )
                                                 'start-to-end'
        ignore               (yes | no)          'no'          >

<!ELEMENT null               EMPTY                              >
<!ELEMENT any                EMPTY                              >

<!-- End of Electronic Writing System Description DTD         -->
```

Appendix B: Electronic writing system description example

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="ewsd.xsl"?>
<!DOCTYPE EWSD SYSTEM "ewsd.dtd" [
  <!ENTITY font-set "serif, sans-serif">
  <!ENTITY eng "&#x014b;">
  <!ENTITY eth "&#x00f0;">
  <!ENTITY opene "&#x025b;">
  <!ENTITY openo "&#x0254;">
  <!ENTITY long "&#x02d0;">
]>
<EWSD xml:lang="en-US" name="Fijian" date="2001-04-16">
  <language>
    <code issuer="Ethnologue">FJI</code>
    <name xml:lang="en-US">Fijian</name>
  </language>

  <author>
    <fullname>Eric Albright</fullname>
    <affiliation>SIL International</affiliation>
    <contact>eric_albright@sil.org</contact>
  </author>

  <introduction>
    <p>Fijian is not a very complicated writing system,
      being simply phonemic. However, although Fijian uses
      letters which are familiar to users of Roman alphabets,
      some of these letters do not have familiar correspondences
      to the sound system and thus may surprise the
      uninitiated.</p>
    <p>The Fijian writing system is detailed in this
      description. Special note is made of the correspondences
      between the graphic symbols and the phonemes they
      represent.</p>
    <p>The Fijian writing system has been described as <q>a
      model of consisitency and simplicity...&#8212;in spite of
      its peculiar use of the letters
      <transcription type="orthographic">b
```

```
    This system is simple, economical, and regular. Initial
    attempts used digraphs to represent the prenasalized
    consonants. But the Fijians, who were just begining to
    learn to read, had difficulty with these. Thus the choice
    of a single graph instead. (Sch&#252;tz 1972, 12)</p>
  ...
</history>

<illustrative-text>
  <p>Okoya na Vosa sa bula e nai vakatekivu, a rau sa
    tiko kei na Kalou ko koya na Vosa, a Kalou ko koya na Vosa.
    Sai koya oqo e rau sa tiko vata kei na Kalou mai nai
    vakatekivu. Sa cakava na ka kecega ko koya: a sa sega e dua
    na ka sa cakavi, me sega ni cakava ko koya. Sa tu vua na
    bula; ia na rarama ni tamata na bula. Sa cila mai na rarama
    e na butobuto; a sa sega ni kunea na butobuto.</p>
  ...
</illustrative-text>

<graphs>
  <introduction>
    <p>The Fijian writing system uses a Roman script and
      thus the graphic forms are familiar. The following are the
      graphic forms that are used to write Fijian. The
      correspondences will be accounted for in a later
      section.</p>
    ...
  </introduction>
  <graph id="a">
    <name>a</name>
  </graph>
  <graph id="A">
    <name>A</name>
  </graph>
  ...
</graphs>

<classes>
  ...
  <class id="loan">
    <name>Loan</name>
    <graphRef target="f"/>
    <graphRef target="F"/>
    <graphRef target="j"/>
    <graphRef target="J"/>
    <graphRef target="p"/>
    <graphRef target="P"/>
    <discussion>
      <p>The letters <transcription
        type="orthographic">f
```

```
        <transcription type="orthographic">j
```

```
  </writing-unit>
</writing-units>

<linguistic-units>
  <introduction>
    <p>The following are the phonemes of Fijian. The
      correspondences between these phonemes and the
      graphs of the Fijian writing system will be taken
      up in a later section.</p>
    ...
  </introduction>
  <linguistic-unit id="phoneme-a" type="phoneme">
    <name>a</name>
  </linguistic-unit>
  <linguistic-unit id="phoneme-a-long" type="phoneme">
    <name>a&long;</name>
  </linguistic-unit>
  ...
</linguistic-units>

<relationships>
  <mapping-set cor1="graph" cor2="linguistic-unit">
    <introduction>
      <p>The following rules establish the
        correspondences between the graphic symbols and the
        phonemes of Fijian. The correspondence is a simple one
        where one phoneme corresponds to a single graph and vice
        versa as the rules are bi-directional. The environments on
        the graphic side only establish the grapho-tactic
        constraints of the writing system since there are no other
        choices for the correspondences.</p>
     ...
    </introduction>
    <name>Reading and Writing</name>
    <rule>
      <correspondence>
        <choice>
          <graphRef target="a"/>
          <graphRef target="A"/>
        </choice>
        <choice>
          <linguistic-unitRef target="phoneme-a"/>
          <linguistic-unitRef target="phoneme-a-long"/>
        </choice>
      </correspondence>
    </rule>
    ...
    <rule>
      <correspondence>
        <choice>
```

```
          <graphRef target="b"/>
          <graphRef target="B"/>
        </choice>
        <linguistic-unitRef target="phoneme-mb"/>
      </correspondence>
      <when>
        <followed-by>
          <classRef target="Vowel"></classRef>
        </followed-by>
      </when>
    </rule>
    ...
  </mapping-set>
  <mapping-set cor1="graph" cor2="coded-unit">
    <name>Unicode encoding</name>
    <rule>
      <correspondence>
        <graphRef target="A"/>
        <coded-unit bits="16" value="65"/>
      </correspondence>
    </rule>
    ...
  </mapping-set>
  <mapping-set cor1="graph" cor2="glyph">
    <name>Rendering</name>
    <rule>
      <correspondence>
        <graphRef target="A"/>
        <glyph font="&font-set;" number="65"/>
      </correspondence>
    </rule>
    ...
  </mapping-set>
  <relation-set feature1="lower case" feature2="upper case">
    <name>Case relation between lower and upper case</name>
    <rule>
      <correspondence>
        <graphRef target="a"/>
        <graphRef target="A"/>
      </correspondence>
    </rule>
    ...
  </relation-set>
</relationships>

<orderings>
  <ordering>
    <name>Alphabetical Order</name>
    <collating-sequence>
      <collating-sequence>
```

```
          <graphRef target="a"/>
          <graphRef target="A"/>
        </collating-sequence>
        ...
        <discussion>
          <p>The alphabetical order of treats the distinction
            between letters as a primary difference and the
            distinction between case as a secondary difference.</p>
        </discussion>
      </collating-sequence>
    </ordering>
  </orderings>

  <references>
    <reference>Churchward, C. Maxwell. 1973. <title>A New
      Fijian Grammar</title>. Suva, Fiji: Government Press.
    </reference>
    ...
  </references>
</EWSD>
```

Appendix C: Electronic writing system description stylesheet

```
<?xml version='1.0'?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:functions="urn:EricAlbright:EWSD"
   version="1.0">

<msxsl:script language="JScript" implements-prefix="functions">
    function Hex(nodelistSource) {
      return parseInt(nodelistSource.item(0).text).toString(16);
    }
    function Char(nodelistSource) {
      return String.fromCharCode(
                  parseInt(nodelistSource.item(0).text));
    }
 </msxsl:script>

  <!-- Has not been defined in stylesheet yet! -->
  <xsl:template match="*">
    <xsl:message terminate="yes">
      <xsl:value-of select="name()"/> not handled in stylesheet yet!
    </xsl:message>
  </xsl:template>



<!-- Root ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xsl:template match="/">
    <HTML>
      <HEAD>
        <TITLE>
          <xsl:call-template name="make-title"/>
        </TITLE>
        <STYLE>
          BODY {
            font-family: Arial Unicode MS;
          }

          H1, H2, H3 {
            font-family:Arial;
            margin-bottom:0;
          }

          p {
            margin-top: .25em;
            margin-bottom: 0;
          }
```

```
          p.authorLine {
            margin-top: 0;
            margin-bottom: 0;
          }

          span.text, span.title {
            font-style: italic;
          }
          span.linguistic-unit {
            font-family: Arial Unicode MS;
          }
          a:link {
            color:blue; text-decoration:none;
          }
          a:visited {
            color:blue; text-decoration:none;
          }

        </STYLE>
      </HEAD>
      <BODY>
        <xsl:apply-templates/>
      </BODY>
    </HTML>
  </xsl:template>

  <xsl:template match="name | introduction |
                       discussion | author | head">
    <xsl:apply-templates/>
  </xsl:template>


<!-- Headings +++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xsl:template match="EWSD">
    <H1>
      <xsl:call-template name="make-title"/>
    </H1>
    <xsl:apply-templates select="child::*[not(self::language
                                              or self::head)]"/>
  </xsl:template>

  <xsl:template match="EWSD/introduction |
                       history | illustrative-text |
                       graphs | graphemes |
                       classes | writing-units |
                       linguistic-units |
                       relationships | orderings |
                       issues | conclusion |
```

```
                    references">
<H2>
 <xsl:choose>
   <xsl:when test="head">
     <xsl:apply-templates select="head"/>
   </xsl:when>
   <xsl:when test="self::introduction">
     <xsl:text>Introduction</xsl:text>
   </xsl:when>
   <xsl:when test="self::history">
     <xsl:text>History</xsl:text>
   </xsl:when>
   <xsl:when test="self::illustrative-text">
     <xsl:text>Illustrative Text</xsl:text>
   </xsl:when>
   <xsl:when test="self::graphs">
     <xsl:text>Graphs</xsl:text>
   </xsl:when>
   <xsl:when test="self::graphemes">
     <xsl:text>Graphemes</xsl:text>
   </xsl:when>
   <xsl:when test="self::classes">
     <xsl:text>Classes</xsl:text>
   </xsl:when>
   <xsl:when test="self::writing-units">
     <xsl:text>Writing Units</xsl:text>
   </xsl:when>
   <xsl:when test="self::linguistic-units">
     <xsl:text>Linguistic Units</xsl:text>
   </xsl:when>
   <xsl:when test="self::relationships">
     <xsl:text>Relationships</xsl:text>
   </xsl:when>
   <xsl:when test="self::orderings">
     <xsl:text>Orderings</xsl:text>
   </xsl:when>
   <xsl:when test="self::issues">
     <xsl:text>Issues</xsl:text>
   </xsl:when>
   <xsl:when test="self::conclusion">
     <xsl:text>Conclusion</xsl:text>
   </xsl:when>
   <xsl:when test="self::references">
     <xsl:text>References</xsl:text>
   </xsl:when>
   <xsl:otherwise>
     <xsl:message terminate="yes">Undefined
     <xsl:value-of select="name()"/> default head.</xsl:message>
   </xsl:otherwise>
 </xsl:choose>
```

```
   </H2>
   <xsl:apply-templates/>
  </xsl:template>

<!-- Author ++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xsl:template match="fullname | affiliation | contact">
    <P class="authorLine">
      <xsl:apply-templates/>
    </P>
  </xsl:template>

<!-- Reference ++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xsl:template match="reference">
    <P>
      <xsl:call-template name="make-anchor"/>
      <xsl:apply-templates/>
    </P>
  </xsl:template>

<!-- Orderings ++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xsl:template match="ordering">
    <H3>
      <xsl:call-template name="make-anchor"/>
      <xsl:for-each select="name">
        <xsl:if test="position() != 1">
          <xsl:text>, </xsl:text>
        </xsl:if>
        <xsl:apply-templates/>
      </xsl:for-each>
    </H3>
    <xsl:apply-templates select="*[not(self::name)]"/>
  </xsl:template>

  <xsl:template match="order-layer">
    <P>
      <xsl:call-template name="make-anchor"/>
      <xsl:apply-templates/>
    </P>
  </xsl:template>

  <xsl:template match="order-rule">
    <P style="margin-bottom:0; margin-top:0;">
      <xsl:attribute name="title">
        <xsl:apply-templates mode="verbose"/>
      </xsl:attribute>
      <xsl:call-template name="make-anchor"/>
      <xsl:apply-templates select="follows"/>
```

```
    <xsl:apply-templates select="graphemeRef | graphRef"/>
    <xsl:apply-templates select="precedes"/>
  </P>
</xsl:template>

<xsl:template match="sort-equivalencies">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="sort-equivalency">
  <xsl:apply-templates select="*[position()=1]"/>
  <xsl:text>=</xsl:text>
  <xsl:apply-templates select="*[position()>1]"/>
</xsl:template>

<xsl:template match="collating-sequence">
  <P>
    <xsl:attribute name="style">text-indent:<xsl:value-of
    select="count(parent::collating-sequence)"/>em</xsl:attribute>
  <xsl:text>[</xsl:text>
  <xsl:if test="position()=1 and child::collating-sequence">
    <xsl:text> </xsl:text>
  </xsl:if>
  <xsl:apply-templates select="*[not(self::discussion)]"/>
  <xsl:text>] </xsl:text>
  </P>
  <xsl:apply-templates select="discussion"/>
</xsl:template>

<xsl:template match="mapping-set | relation-set">
  <H3><xsl:value-of select="name[1]"/></H3>
  <xsl:apply-templates select="introduction | rule | discussion"/>
</xsl:template>

<xsl:template match="rule">
  <P style="margin-bottom:0; margin-top:0em;">
    <xsl:attribute name="title">
      <xsl:apply-templates mode="verbose"/>
    </xsl:attribute>

    <xsl:call-template name="make-anchor"/>
    <xsl:apply-templates/>
  </P>
</xsl:template>

<xsl:template match="correspondence">
  <xsl:apply-templates select="*[position()=1]"/>
  <SPAN title="corresponds to">
    <xsl:text> &#x2258; </xsl:text>
  </SPAN>
```

```
  <xsl:apply-templates select="*[position()>1]"/>
</xsl:template>

<xsl:template match="coded-unit">
  <xsl:value-of select="@value"/>
  <xsl:text> (x</xsl:text>
    <xsl:value-of select="functions:Hex(@value)"/>
  <xsl:text>)</xsl:text>
</xsl:template>

<xsl:template match="glyph">
  <xsl:if test="@name">
    <xsl:text>name: </xsl:text>
    <xsl:value-of select="@name"/>
  </xsl:if>
  <xsl:if test="@number">
    <xsl:text>glyph identifier: </xsl:text>
    <xsl:value-of select="@number"/>
  </xsl:if>
  <xsl:apply-templates select="@img"/>
</xsl:template>

<xsl:template match="glyph/@img">
  <xsl:text> (</xsl:text>
  <IMG style="vertical-align:base-line;" src="{.}"></IMG>
  <xsl:text>)</xsl:text>
</xsl:template>

<xsl:template match="grapheme">
  <P>
    <xsl:call-template name="make-anchor"/>
    <xsl:for-each select="name">
      <xsl:if test="position() != 1">
        <xsl:text>, </xsl:text>
      </xsl:if>
      <xsl:apply-templates/>
    </xsl:for-each>
  </P>
</xsl:template>

<xsl:template match="graph">
  <P>
    <xsl:call-template name="make-anchor"/>
    <xsl:if test="@image">
      <IMG href="{@image}"/>
      <xsl:text> &#8212;<!--em dash--> </xsl:text>
    </xsl:if>

    <xsl:for-each select="name">
      <xsl:if test="position() != 1">
```

```
        <xsl:text>, </xsl:text>
      </xsl:if>
      <xsl:apply-templates/>
    </xsl:for-each>
  </P>
</xsl:template>

<xsl:template match="graphRef | context-graphRef">
  <xsl:for-each select="//graph[@id=current()/@target]">
    <A href="#{generate-id()}">
      <xsl:attribute name="title">the graph named
      <xsl:value-of select="name"/></xsl:attribute>
      <xsl:value-of select="name"/>
    </A>
  </xsl:for-each>

  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="class">
  <P class="class">
    <xsl:call-template name="make-anchor"/>

    <xsl:for-each select="name">
      <xsl:if test="position() > 1">
        <xsl:text> (</xsl:text>
      </xsl:if>
      <SPAN>
        <xsl:attribute name="title">
          <xsl:choose>
            <xsl:when test="position() = 1">
              <xsl:text>the class named </xsl:text>
            </xsl:when>
            <xsl:otherwise>
              <xsl:text>also known as </xsl:text>
            </xsl:otherwise>
          </xsl:choose>
          <xsl:apply-templates select="."/>
        </xsl:attribute>
        <xsl:apply-templates select="."/>
      </SPAN>
      <xsl:if test="position() > 1">
        <xsl:text>)</xsl:text>
      </xsl:if>
    </xsl:for-each>

    <SPAN title="contains as members">
      <xsl:text> &#x220b; </xsl:text>
    </SPAN>
```

```
    <xsl:for-each select="*[not(self::name or self::discussion)]">
      <xsl:if test="position() != 1">
        <xsl:text>, </xsl:text>
      </xsl:if>
      <xsl:apply-templates select="."/>
    </xsl:for-each>
  </P>
  <xsl:apply-templates select="discussion"/>
</xsl:template>

<xsl:template match="classRef | context-classRef">
  <xsl:if test="excluding | when">
    <xsl:text>(</xsl:text>
  </xsl:if>
  <xsl:for-each select="//class[@id=current()/@target]">
    <A href="#{generate-id()}">
      <xsl:attribute name="title">the class named
      <xsl:value-of select="name"/></xsl:attribute>
      <xsl:value-of select="name"/>
    </A>
  </xsl:for-each>
  <xsl:apply-templates/>
  <xsl:if test="excluding | when">
    <xsl:text>)</xsl:text>
  </xsl:if>
</xsl:template>

<xsl:template match="writing-unit">
  <P class="writing-unit">
    <xsl:call-template name="make-anchor"/>

    <xsl:for-each select="name">
      <xsl:if test="position() > 1">
        <xsl:text>(</xsl:text>
      </xsl:if>
      <SPAN>
        <xsl:attribute name="title">
          <xsl:choose>
            <xsl:when test="position() = 1">
              <xsl:text>the writing unit named </xsl:text>
            </xsl:when>
            <xsl:otherwise>
              <xsl:text>also known as </xsl:text>
            </xsl:otherwise>
          </xsl:choose>
          <xsl:apply-templates select="."/>
        </xsl:attribute>
        <xsl:apply-templates select="."/>
      </SPAN>
      <xsl:if test="position() > 1">
```

```
        <xsl:text>)</xsl:text>
      </xsl:if>
    </xsl:for-each>

    <xsl:apply-templates select="*[not(self::name or
                                       self::discussion)]"/>
  </P>
  <xsl:apply-templates select="discussion"/>
</xsl:template>

<xsl:template match="writing-unitRef">
  <xsl:for-each select="//writing-unit[@id=current()/@target]">
    <A href="#{generate-id()}">
      <xsl:attribute name="title">the writing unit named
      <xsl:value-of select="name"/></xsl:attribute>
      <xsl:value-of select="name"/>
    </A>
  </xsl:for-each>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="is-made-of">
  <SPAN title="is made of">
    <xsl:text> &#x2258; </xsl:text>
  </SPAN>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="linguistic-unit">
  <P class="writing-unit">
    <xsl:call-template name="make-anchor"/>

    <xsl:for-each select="name">
      <xsl:if test="position() > 1">
        <xsl:text>(</xsl:text>
      </xsl:if>
      <SPAN>
        <xsl:attribute name="title">
          <xsl:choose>
            <xsl:when test="position() = 1">
              <xsl:text>the </xsl:text>
              <xsl:value-of select="parent::*/@type"/>
              <xsl:text> named </xsl:text>
            </xsl:when>
            <xsl:otherwise>
              <xsl:text>also known as </xsl:text>
            </xsl:otherwise>
          </xsl:choose>
          <xsl:apply-templates select="."/>
        </xsl:attribute>
```

```
        <xsl:choose>
          <xsl:when test="parent::*/@type='phoneme'">
            <xsl:text>/</xsl:text>
          </xsl:when>
          <xsl:when test="parent::*/@type='phone'">
            <xsl:text>[</xsl:text>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="parent::*/@type"/>
            <xsl:text> (</xsl:text>
          </xsl:otherwise>
        </xsl:choose>
        <xsl:apply-templates select="."/>
        <xsl:choose>
          <xsl:when test="parent::*/@type='phoneme'">
            <xsl:text>/</xsl:text>
          </xsl:when>
          <xsl:when test="parent::*/@type='phone'">
            <xsl:text>]</xsl:text>
          </xsl:when>
          <xsl:otherwise>
            <xsl:text>)</xsl:text>
          </xsl:otherwise>
        </xsl:choose>

      </SPAN>
      <xsl:if test="position() > 1">
        <xsl:text>)</xsl:text>
      </xsl:if>
    </xsl:for-each>
    <xsl:if test="@link">
      <A href="{@link}"> more info...</A>
    </xsl:if>
  </P>
  <xsl:apply-templates select="discussion"/>
</xsl:template>

<xsl:template name="lu">
  <SPAN>
    <xsl:attribute name="title">
      <xsl:text>the </xsl:text>
      <xsl:value-of select="@type"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="."/>
    </xsl:attribute>

    <xsl:choose>
      <xsl:when test="@type='phoneme'">
        <xsl:text>/</xsl:text>
      </xsl:when>
```

```
      <xsl:when test="@type='phone'">
        <xsl:text>[</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="@type"/>
        <xsl:text> (</xsl:text>
      </xsl:otherwise>
    </xsl:choose>

    <SPAN class="linguistic-unit"><xsl:value-of select="."/></SPAN>
    <xsl:choose>
      <xsl:when test="@type='phoneme'">
        <xsl:text>/</xsl:text>
      </xsl:when>
      <xsl:when test="@type='phone'">
        <xsl:text>]</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text>)</xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </SPAN>
</xsl:template>

<xsl:template match="linguistic-unitRef">
  <xsl:for-each select="//linguistic-unit[@id=current()/@target]">
    <A href="#{generate-id()}">
      <xsl:call-template name="lu"/>
    </A>
  </xsl:for-each>
</xsl:template>


<xsl:template match="graphemeRef">
  <xsl:for-each select="//grapheme[@id=current()/@target]">
    <A href="#{generate-id()}">
      <xsl:attribute name="title">the grapheme named
      <xsl:value-of select="name"/></xsl:attribute>
      <xsl:value-of select="name"/>
    </A>
  </xsl:for-each>
</xsl:template>

<xsl:template match="excluding">
  <SPAN title="excluding">
    <xsl:text> &#x220c; </xsl:text>
  </SPAN>
  <xsl:apply-templates/>
</xsl:template>
```

```
<xsl:template match="when">
  <SPAN title="in the environment of">
    <xsl:text> / </xsl:text>
  </SPAN>
  <xsl:apply-templates select="preceded-by"/>
  <xsl:text> _ </xsl:text>
  <xsl:apply-templates select="followed-by"/>
  <xsl:apply-templates select="in"/>
</xsl:template>

<xsl:template match="preceded-by | followed-by | in">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="preceded-by" mode="verbose">
  <xsl:apply-templates mode="verbose"/>
</xsl:template>

<xsl:template match="followed-by" mode="verbose">
  <xsl:apply-templates mode="verbose"/>
</xsl:template>

<xsl:template match="in" mode="verbose">
  <xsl:apply-templates mode="verbose"/>
</xsl:template>


<xsl:template match="must-precede">
  <xsl:text> &#x227a; </xsl:text>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="must-precede" mode="verbose">
  <xsl:apply-templates mode="verbose"/>
</xsl:template>

<xsl:template match="must-follow">
  <xsl:text> &#x227b; </xsl:text>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="must-follow" mode="verbose">
  <xsl:apply-templates mode="verbose"/>
</xsl:template>

<xsl:template name="make-title">
  <xsl:choose>
    <xsl:when test="/EWSD/head">
      <xsl:apply-templates select="/EWSD/head"/>
    </xsl:when>
```

```
      <xsl:otherwise>
        <xsl:apply-templates select="/EWSD/language/name"/>
        <xsl:text> Writing System Description</xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <xsl:template name="make-anchor">
    <A>
      <xsl:attribute name="id">
        <xsl:value-of select="generate-id()"/>
      </xsl:attribute>
    </A>
  </xsl:template>

  <xsl:template match="null">
    empty-set
  </xsl:template>

  <xsl:template match="any">
    full-set
  </xsl:template>

<!-- Groups ++++++++++++++++++++++++++++++++++++++++++++++++++ -->

  <xsl:template match="sequence">
    <SPAN title="sequence of">
      <xsl:text>( </xsl:text>
    </SPAN>
      <xsl:for-each select="child::*">
        <xsl:if test="position() != 1">
          <SPAN title="followed by">
            <xsl:text>, </xsl:text>
          </SPAN>
        </xsl:if>
        <xsl:apply-templates select="."/>
      </xsl:for-each>
    <SPAN title="sequence of">
      <xsl:text> )</xsl:text>
    </SPAN>
  </xsl:template>

  <xsl:template match="choice">
    <SPAN title="choice of">
      <xsl:text>( </xsl:text>
    </SPAN>
      <xsl:for-each select="child::*">
        <xsl:if test="position() != 1">
          <SPAN title="or">
            <xsl:text> | </xsl:text>
```

```
          </SPAN>
        </xsl:if>
        <xsl:apply-templates select="."/>
      </xsl:for-each>
    <SPAN title="sequence of">
      <xsl:text> )</xsl:text>
    </SPAN>
  </xsl:template>

  <xsl:template match="optional">
    <xsl:apply-templates/>
    <xsl:if test="not(repeatable)">
      <SPAN title="optional">
        <xsl:text>?</xsl:text>
      </SPAN>
    </xsl:if>
  </xsl:template>

  <xsl:template match="repeatable">
    <xsl:apply-templates/>
      <SPAN title="repeatable">
        <xsl:text>+</xsl:text>
      </SPAN>
  </xsl:template>

  <xsl:template match="optional/repeatable">
    <xsl:apply-templates/>
      <SPAN title="optional and repeatable">
        <xsl:text>*</xsl:text>
      </SPAN>
  </xsl:template>

<!-- ++++++++++++++++++++++++ User Defined ++++++++++++ -->
  <xsl:template match="p">
    <P>
       <xsl:apply-templates/>
    </P>
  </xsl:template>

  <xsl:template match="example">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="text">
    <SPAN class="text">
      <xsl:apply-templates/>
    </SPAN><xsl:text> </xsl:text>
  </xsl:template>

  <xsl:template match="gloss | q">
```

```
  <xsl:call-template name="alternate-quote"/>
</xsl:template>

<xsl:template match="transcription">
    <xsl:choose>
      <xsl:when test="@type='phonetic'">
        <xsl:text>[</xsl:text>
      </xsl:when>
      <xsl:when test="@type='phonemic'">
        <xsl:text>/</xsl:text>
      </xsl:when>
      <xsl:when test="@type='orthographic'">
        <xsl:text>&#x2329;<!-- left pointing
                              angle bracket --></xsl:text>
      </xsl:when>
    </xsl:choose>

    <xsl:text>&#xfeff; &#xfeff;<!--
                  thin space --></xsl:text>
    <xsl:apply-templates/>
    <xsl:text>&#xfeff; &#xfeff;<!--
                  thin space --></xsl:text>

    <xsl:choose>
      <xsl:when test="@type='phonetic'">
        <xsl:text>]</xsl:text>
      </xsl:when>
      <xsl:when test="@type='phonemic'">
        <xsl:text>/</xsl:text>
      </xsl:when>
      <xsl:when test="@type='orthographic'">
        <xsl:text>&#x232A;<!-- right pointing
                              angle bracket --></xsl:text>
      </xsl:when>
    </xsl:choose>
</xsl:template>

<xsl:template name="alternate-quote">
  <xsl:param name="contents"><xsl:apply-templates/></xsl:param>

  <xsl:choose>
    <xsl:when test="0=count(ancestor::q |
                            ancestor::soCalled |
                            ancestor::gloss) mod 2">
      <xsl:text>&#x201c;<!--ldquote--></xsl:text>
      <xsl:copy-of select="$contents"/>
      <xsl:text>&#x201d;<!--ldquote--></xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>&#x2018;<!--lsquote--></xsl:text>
```

```
        <xsl:copy-of select="$contents"/>
        <xsl:text>&#x2019;<!--lsquote--></xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <xsl:template match="title">
    <SPAN class="title">
      <xsl:apply-templates/>
    </SPAN>
  </xsl:template>


</xsl:stylesheet>
```

Bibliography

Antworth, Evan L. 1990. *PC-KIMMO: A Two-level Processor for Morphological Analysis.* Dallas, TX: Summer Institute of Linguistics.

———— and J. Randolph Valentine. 1998. "Software for Doing Field Linguistics." in Lawler, John M. and Helen Aristar Dry, eds. *Using Computers in Linguistics: A Practical Guide.* London, New York: Routledge. 170–196.

Armstrong, Robert G., ed. 1986. *Orthographies of Nigerian Languages: Manual IV*. Lagos, Nigeria: National Language Centre, Federal Ministry of Education.

Banjo, Ayo, ed. 1985. *Orthographies of Nigerian Languages: Manual III.* Lagos, Nigeria: National Language Centre, Federal Ministry of Education.

Bazell, Charles Ernest. 1956. "The Grapheme." *Litera.* 3:43–46.

Becker, Joseph D. 1984. "Multilingual word processing." *Scientific American.* 251(1):96–107.

Bender, Marvin L., Sydney W. Head and Roger Cowley. 1976. "The Ethiopian Writing System." in Bender, Marvin L., J. Donald Bowen, Robert L. Cooper and Charles A. Ferguson, eds. *Language in Ethiopia.* London: Oxford University Press. 120–129.

Birnbaum, David J., Mavis Cournane and Peter Flynn. 1999. "Using the TEI Writing System Declaration (WSD)." *Computers and the Humanities.* 33(1–2):49–57.

Booij, Geert E. 1987. "The Reflection of Linguistic Structure in Dutch spelling." in Luelsdorff, Philip A., ed. *Orthography and Phonology.* Amsterdam, Philadelphia: John Benjamins Publishing. 215–224.

Bright, William. 1996. "The Devanagari Script." in Daniels, Peter T. and William Bright, eds. *The World's Writing Systems.* New York: Oxford University Press. 384–390.

Butt, David. 1996. "Theories, Maps and Descriptions: An Introduction." in Hasan, Ruqaiya, Carmel Cloran and David Butt, eds. *Functional Descriptions: Theory in Practice.* Amsterdam, Philadelphia: John Benjamins Publishing. *Current Issues in Linguistic Theory.* 121. xv–xxxv.

Campbell, George L. 1997. *Handbook of Scripts & Alphabets.* London, New York: Routledge.

Carney, Edward. 1994. *A Survey of English Spelling.* London, New York: Routledge.

Correll, Sharon. 2000. *Graphite: An Extensible Rendering Engine for Complex Writing Systems.* Paper presented at 17th International Unicode Conference: www.sil.org/computing/graphite/IUC17_paper.PDF.

Coulmas, Florian. 1989. *The Writing Systems of the World.* Oxford: Blackwell.

Daniels, Peter T. 1992. "Is a Structural Graphemics Possible?" in Brend, Ruth M., ed. *The Eighteenth LACUS Forum 1991.* Lake Bluff, IL: LACUS. 528–537.

———— 1995. "Reply to Herrick." in Powell, Mava Jo, ed. *The Twenty-first LACUS Forum 1994.* Chapel Hill, NC: LACUS. 425–431.

———— 1996. "The Study of Writing Systems." in Daniels, Peter T. and William Bright, eds. *The World's Writing Systems.* New York: Oxford University Press. 3–20.

———— and William Bright, eds. 1996. *The World's Writing Systems.* New York: Oxford University Press.

DeFrancis, John. 1989. *Visible Speech: the Diverse Oneness of Writing Systems.* Honolulu: University of Hawaii Press.

Derwing, Bruce L., Tom M. S. Priestly and Bernard L. Rochet. 1987. "The Description of Spelling-to-sound Relationships in English, French and Russian: Progress, Problems and Prospects." in Luelsdorff, Philip A., ed. *Orthography and Phonology.* Amsterdam, Philadelphia: John Benjamins Publishing. 31–52.

Diringer, David. 1968. *The Alphabet: A Key to the History of Mankind.* 3rd edition. New York: Funk & Wagnalls.

Dürst, Martin J. and François Yergeau, eds. 1999. *Character Model for the World Wide Web.* World Wide Web Consortium: www.w3.org/TR/1999/WD-charmod-19991129.

Gelb, I. J. 1963. *A Study of Writing.* Revised edition, first published 1952. Chicago: University of Chicago Press.

Gleason, H. A. 1961. *An Introduction to Descriptive Linguistics.* Revised edition. New York et alibi: Holt, Rinehart and Winston.

Grant, Bruce K. 1982. *A Guide to Korean Characters: Reading and Writing Hangŭl and Hanja.* Second Revised Edition. Elizabeth, NJ, Seoul, Korea: Hollym.

Haas, William. 1983. "Determining the Level of a Script." in Coulmas, Florian and Konrad Ehlich, eds. *Writing in Focus.* Berlin, New York, Amsterdam: Mouton. *Trends in Linguistics. Studies and Monographs.* 24. 15–29.

Hall, Robert A. 1960. "A Theory of Graphemics." *Acta Linguistica Hafniensa.* 8:13–20.

———— 1964. *Introductory Linguistics.* Philadelphia: Chilton Books.

Halliday, M. A. K. 1996. "On Grammar and Grammatics." in Hasan, Ruqaiya, Carmel Cloran and David Butt, eds. *Functional Descriptions: Theory in Practice.* Amsterdam, Philadelphia: John Benjamins Publishing. *Current Issues in Linguistic Theory.* 121. 1–38.

Harris, Roy. 1995. *Signs of Writing.* London, New York: Routledge.

Hartell, Rhonda L., ed. 1993. *Alphabets of Africa.* Dakar: UNESCO.

Henderson, Leslie. 1984. "Writing Systems and Reading Processes." in Henderson, Leslie, ed. *Orthographies and Reading: Perspectives from Cognitive Psychology, Neuropsychology, and Linguistics.* London, Hillsdale, NJ: Lawrence Erlbaum Associates. 11–24.

Herrick, Earl Myron. 1974. "A Taxonomy of Alphabets and Scripts." *Visible Language.* 8:5–32.

———— 1995a. "Of Course a Structural Graphemics is Possible!" in Powell, Mava Jo, ed. *The Twenty-first LACUS Forum 1994.* Chapel Hill, NC: LACUS. 413–424.

———— 1995b. "Reply to Daniels's Reply." in Powell, Mava Jo, ed. *The Twenty-first LACUS Forum 1994.* Chapel Hill, NC: LACUS. 432–440.

Hill, Archibald A. 1967. "The Typology of Writing Systems." in Austin, William M., ed. *Papers in Linguistics in Honor of Léon Dostert.* The Hague, Paris: Mouton. *Janua Linguarum Series Maior.* 25. 93–99.

Hockey, Susan. 1998. "Textual Databases." in Lawler, John M. and Helen Aristar Dry, eds. *Using Computers in Linguistics: A Practical Guide.* London, New York: Routledge. 101–137.

Hosken, M., B. Hallissy, W. Cleveland, S. Correll and A. Ward. 2000. *Graphite Description Language.* Version 1.900. Dallas: SIL International.

Huttar, George L. 1987. "The Afaka Script: An Indigenous Creole Syllabary." in Fleming, Ilah, ed. *The Thirteenth LACUS Forum 1986.* Lake Bluff, IL: LACUS. 167–177.

Kannaiyan, V. 1960. *Scripts In and Around India.* Madras, India: Government Museum.

Matthiessen, Christian and Christopher Nesbitt. 1996. "On the Idea of Theory-Neutral Descriptions." in Hasan, Ruqaiya, Carmel Cloran and David Butt, eds. *Functional Descriptions: Theory in Practice.* Amsterdam, Philadelphia: John Benjamins Publishing. *Current Issues in Linguistic Theory.* 121. 39–83.

Mountford, John. 1990. "Language and Writing-Systems." in Collinge, N. E., ed. *An Encyclopedia of Language.* New York: Routledge, Chapman & Hall. 701–739.

———— 1996. "A Functional Classification." in Daniels, Peter T. and William Bright, eds. *The World's Writing Systems.* New York: Oxford University Press. 627–632.

Nakanishi, Akira. 1980. *Writing Systems of the World: Alphabets, Syllabaries, Pictograms (Sekai no Moji).* Rutland, VT: Charles E. Tuttle Co.

Nunberg, Geoffrey. 1990. *The Linguistics of Punctuation.* Stanford, CA: Center for the Study of Language and Information. *CSLI Lecture Notes.* 18.

Pike, Kenneth L. 1947. *Phonemics.* Ann Arbor: University of Michigan Press.

———— 1982. *Linguistic Concepts: An Introduction to Tagmemics.* Lincoln, London: University of Nebraska Press.

Pulgram, Ernst. 1951. "Phoneme and Grapheme: A Parallel." *Word.* 7:15–20.

———— 1965. "Graphic and Phonic Systems: Figurae and Signs." *Word.* 21:208–224.

Roop, D. Haigh. 1997. *An Introduction to the Burmese Writing System.* Manoa, HI: University of Hawaii at Manoa. *Southeast Asia Paper.* (41).

Sampson, Geoffrey. 1985. *Writing Systems: a Linguistic Introduction.* Stanford, CA: Stanford University Press.

Schaefer, Ronald P. 1987. *An Initial Orthography and Lexicon for Emai: an Edoid Language of Nigeria.* Bloomington, IN: Indiana University Linguistics Club. *Studies in African Grammatical Systems.* 5.

Sgall, Petr. 1987. "Towards a Theory of Phonemic Orthography." in Luelsdorff, Philip A., ed. *Orthography and Phonology.* Amsterdam, Philadelphia: John Benjamins Publishing. 1–30.

Simons, Gary F. 1989. "The Computational Complexity of Writing Systems." in Brend, Ruth M. and David G. Lockwood, eds. *The Fifteenth LACUS Forum 1988.* Lake Bluff, IL: LACUS. 538–553.

——————— 1998. "The Nature of Linguistic Data and the Requirements of a Computing Environment for Linguistic Research." in Lawler, John M. and Helen Aristar Dry, eds. *Using Computers in Linguistics: A Practical Guide.* London, New York: Routledge. 10–25.

——————— and John V. Thomson. 1998. "Multilingual Data Processing in the CELLAR Environment." in Nerbonne, John, ed. *Linguistic databases.* Stanford, CA: Center for the Study of Language and Information. 203–234.

Smalley, William A., ed. 1964. *Orthography Studies.* London: United Bible Societies.

Sperberg-McQueen, C. M. and Lou Burnard, eds. 1999. *Guidelines for Electronic Text Encoding and Interchange.* Revised Reprint. Chicago, Oxford: Text Encoding Initiative.

Sproat, Richard. 2000. *A Computational Theory of Writing Systems.* Cambridge: Cambridge University Press. *Studies in Natural Language Processing.*

Tadadjeu, Maurice and Etienne Sadembouo, eds. 1984. translated by Emmanuel Chia. *General Alphabet of Cameroon Languages (Alphabet General Des Langues Camerounaises).* Bilingual edition. Yaoundé, Cameroon: University of Yaoundé. *PROPELCA.* 1.

Unicode Consortium. 2000. *The Unicode Standard.* Version 3.0. Reading, MA: Addison-Wesley.

Venezky, Richard L. 1962. *A Computer Program for Deriving Spelling to Sound Correlations.* Masters thesis. Cornell University.

———— 1965. *A Study of English Spelling-to-Sound Correspondences on Historical Principles.* Dissertation. Stanford: Stanford University.

———— 1967a. "English Orthography: Its Graphical Structure and its Relation to Sound." *Reading Research Quarterly.* 2.

———— 1967b. "The Basis of English Orthography." *Acta Linguistica.* 10:145–159.

———— 1970. *The Structure of English Orthography.* The Hague: Mouton. *Janua Linguarum. Series Minor.* 82.

Weir, Ruth H. 1967. "Some Thoughts on Spelling." in Austin, William M., ed. *Papers in Linguistics in Honor of Léon Dostert.* The Hague, Paris: Mouton. *Janua Linguarum Series Maior.* 25. 169–177.

Wijk, Axel. 1966. *Rules of Pronunciation for the English Language: An Account of the Relationship between English Spelling and Pronunciation.* London: Oxford University Press. *Language and Language Learning.* 12.

Williamson, Kay, ed. 1983. *Orthographies of Nigerian Languages: Manual II.* Lagos, Nigeria: National Language Centre, Federal Ministry of Education.

Eric Albright received his Bachelor of Arts degree in Communications from Bryan College in 1994. After working as a computer programmer for a year in the SIL translation department, he began his study of linguistics at the Texas SIL. After a year, he took a job as a computer programmer in the industry.

In 1998, he joined SIL International and resumed his linguistic studies. As an expert in XML and XSL technologies, he has made numerous contributions to XML related projects within SIL International.

In December 2000, he presented a paper on his preliminary thesis research at the Linguistic Exploration Workshop on Web-Based Language Documentation and Description.

He will be taking a field assignment with SIL in August 2001.

This page intentionally left blank.