Automatic Concordance Generation of Syriac Texts

George Anton Kiraz

1. Introduction

In the *Vtum Symposium Syriacum*, I gave a paper on computer applications in Syriac studies (OCA 236, p. 451-8), where I mentioned briefly the issue of generating concordances; the current paper is dedicated to this subject.

The generation of automatic indices and concordances from computer-readable texts falls under the rubric of 'literary and linguistic computing'. This paper aims at describing a database used in concordance generation of Syriac texts, sc. the SEDRA database, and the concordance-generator, SyrCONC. The database and the program were designed and implemented for the purpose of generating my *Concordance to the Syriac New Testament* (Brill 1993), hereinafter *Concordance*.

Sections 2 and 3 give a description of SEDRA and text databases; section 4 describes the concordance-generator SyrCONC; finally, section 5 gives the highlights of related current research. (Technical sections are marked with an asterisk, '*'; these sections can be skipped without loss in continuity.)

2. Description of the SEDRA Database

The *Syriac Electronic Data Retrieval Archive* (SEDRA) is a database management system for linguistic computing in Syriac. (Its name is derived from Syr. *sedrâ* 'array'; databases are arrays of information.) Such a system can be used in a wide range of linguistic applications, especially lexically oriented projects.

2.1. History

SEDRA was designed in 1989 to generate the mentioned *Concordance*. The first version of the database, SEDRA I, was a relational database and made a substantial use of the Syriac New Testament database of the Way International (New Knoxville, Ohio); the latter was used by the Way International to generate their concordance (*The Concordance to the Peshitta Version of the Aramaic New Testament*, Ohio 1985). The plans for the new

Concordance, however, required more information to be added to the database; as a result, SEDRA was modified in 1990 resulting in SEDRA II.

Early 1991 was the time to start writing the concordance-generator, i.e. a program which would generate the *Concordance* from the database; however, while implementing the program, it became clear that the network model would be more adequate than the relational model (this jargon will be discussed under 'Preliminaries' below). The database was redesigned using the network model, and additional fields were added. This version of the system, SEDRA III, was used to generate the *Concordance* and is described in this paper.

2.2. Preliminaries

Before looking at SEDRA III, introducing the subject of database systems seems in order. A database is a collection of data arranged for ease and speed of search and retrieval. The term schema is used to describe the arrangement of a database (what information the database should contain and how the information is organized). A library, for example, uses a database to maintain its catalogue which provides easy and speedy search and retrieval on books, authors, publishers, etc.

Related items of data are treated as a unit and maintained in a record; each element of a record is a field (records are indicated here between curly braces, { }, and fields between square brackets, [ ]). Each record entry in the database is an instance of the record. In the library example, the database may maintain two records: {BOOK} and {AUTHOR}; the former may contain the fields [Title] and [Author], and the latter may contain the fields [Last-Name], [First-Name] and [Key]. Figure 1 illustrates this showing two instances of the record {AUTHOR} and three instances of the record {BOOK}.

How can one find the author of a book, or the books written by an author? One can see in the above example that the values of the [Author] fields in the {BOOK} records correspond to the values of the [Key] fields in the {AUTHOR} records; in fact, this constitutes a *relation* between the two record instances. Databases whose records are related through common data fields (e.g. 'JG' and 'WC' in Figure 1) are said to be relational databases.

There are other ways of representing relations between records. A network database explicitly defines a relation between two records through a set: a set enables one record (e.g. the {AUTHOR} record) to be linked to a second record (e.g. the {BOOK} record), where each *one* instance of the

{AUTHOR}

[Last-Name]	Cureton
[First-Name]	William
[Key]	WC
[Last-Name]	Gwynn
[First-Name]	John
[Key]	JG

{BOOK}

[Title]	Apocalypse of St John
[Author]	JG
[Title]	Ancient Syriac Documents
[Author]	WC
[Title]	Remnants of the Later Syriac . . .
[Author]	JG

Figure 1. A relational database.

former can be linked to *many* instances of the latter. In this one-to-many relation, the former is called the owner of the set and the latter the member of the set. Figure 2 shows the {AUTHOR} record owning the {BOOK} record; one can see that each author can be linked to many books (the links are shown with arrows), and there is no need for common fields anymore.

From the above, we deduce that in order to define a set, one needs to specify three elements: the owner record (e.g. the record {AUTHOR}), the member record (e.g. the record {BOOK}), and the sorting order to determine which instance of the member record is 1st, 2nd, etc. (the members are organized according to the value of the field [Title] in the above example). Note that the field which determines the order is always in the member record.

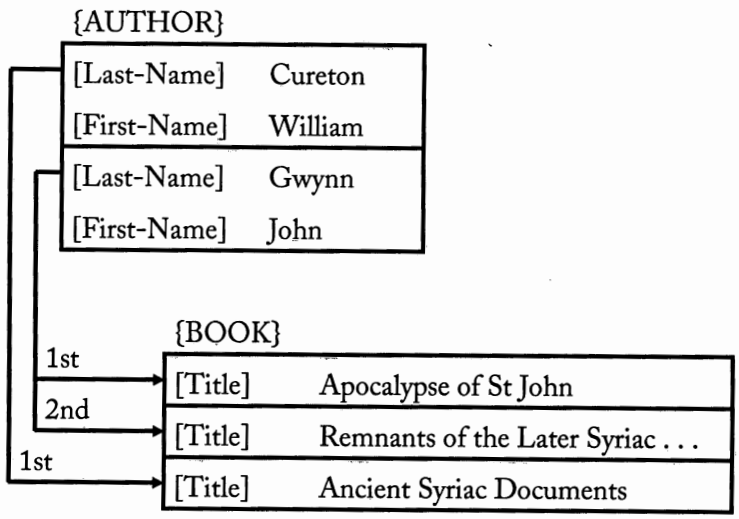

Figure 2. A network database.

2.2.1. *Technical note on set implementation**

In the network model, sets are linked lists of database addresses (§2.5). Each instance of records which are owners should contain set pointers: fields which point (using database addresses) to the first and last members of the set they own. Each instance of records which are members should contain member pointers: fields which point to the owner record, previous member record, and next member record. These pointers are in fact implicit fields in records.

2.3. The schema of SEDRA

Now we turn our attention to describe the SEDRA database. Figure 3 gives the schema of the database with some examples.

SEDRA III maintains the linguistic information of Syriac in six records:

- The record {ROOT}: each instance gives a root, e.g. */ktb/*.
- The record {LEXEME}: each instance gives a lexeme, e.g. */ktābâ/ 'book'*.
- The record {WORD}: each instance gives a word, e.g. */waktābeh/* 'and his book'.
- The record {MEANING}: gives English meanings.
- The record {ETYMOLOGY}: gives the etymology of words which derive from foreign languages.
- The record {NOTES}: allows notes to be written.

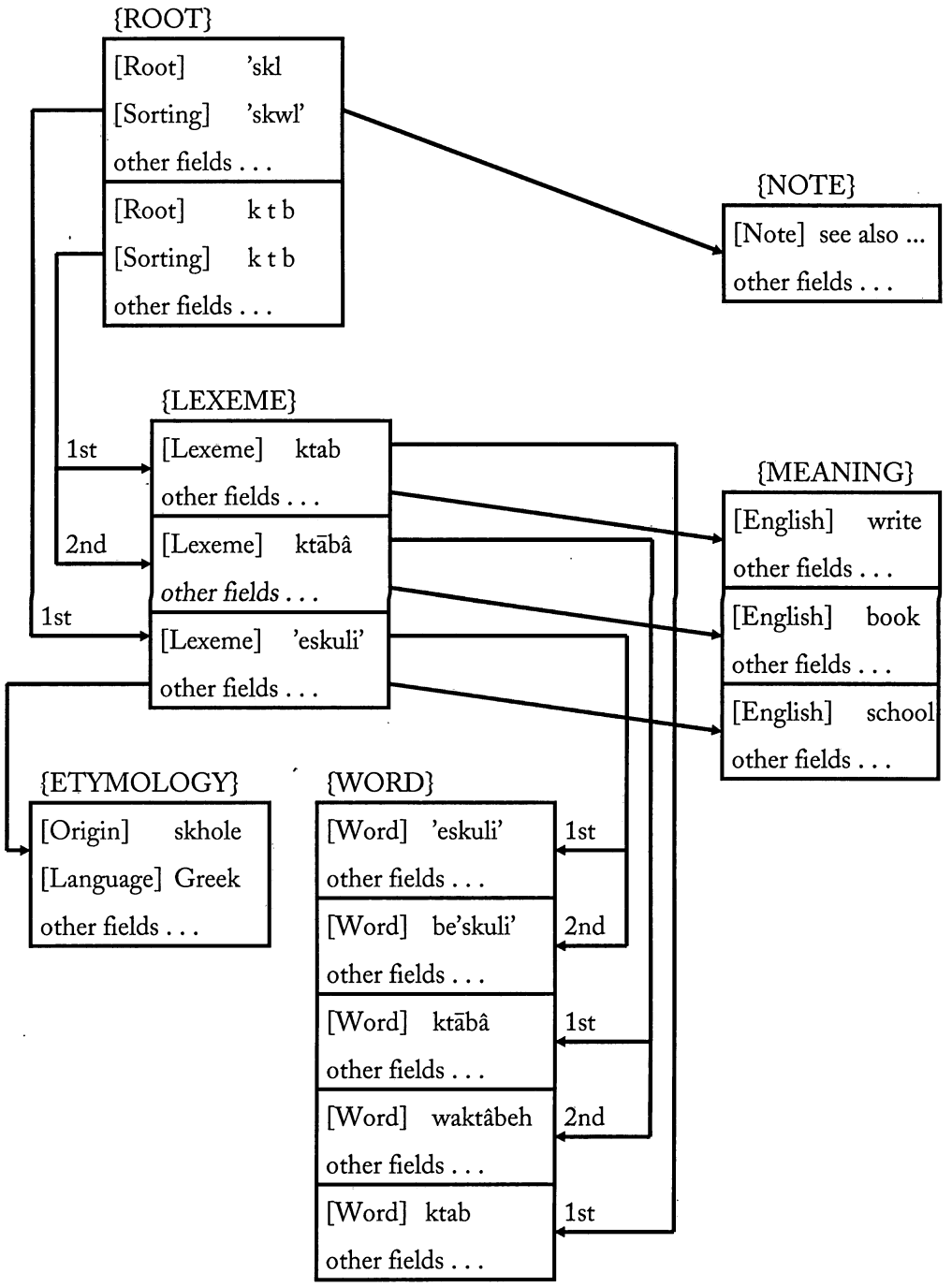

Figure 3. SEDRA III schema.

The illustration shows arrows which indicate set relations; the source of an arrow indicates an owner record and its destination point to a member record. There are six sets in SEDRA:

- The set ROOT-LEXEME allows each root instance to own many lexeme instances, e.g. the root */ktb/* owns the lexemes */ktab/* 'to write' and */ktābâ/* 'book'.
- The set ROOT-NOTES allows each root to have any number of notes associated with it, e.g. the root */mskn/* has the note "see */skn/*" associated with it.
- The set LEXEME-WORD allows each lexeme instance to own many word instances, e.g. the lexeme */ktābâ/* 'book' owns the words */ktātâ/*, */baktābâ/*, */waktābeh/* etc.
- The set LEXEME-MEANING allows each lexeme instance to own many meaning instances, e.g. the lexeme */ktābo'/* owns the meanings 'Scripture', 'book' and 'writing'.
- The set LEXEME-ETYMOLOGY allows each lexeme instance to own an etymological source, e.g. the lexeme */'eskuli'/* owns the etymological source Gr. *skhole*.
- The set LEXICAL-NOTES allows each lexeme to have any number of notes associated with it, e.g. the lexeme */baytâ/* has the note "for */beyt/* = 'between', see under */byn/*".

Following is a detailed description of the fields contained in each record.

2.3.1. *The record* {ROOT}

This record consists of four fields:

- [Root] gives the orthographic form of the root, e.g. */qm/*.
- [Sorting] gives the orthographic form for sorting purposes, e.g. */qwm/*.
- [Seyāmê] a flag which indicates the presence or absence of *seyāmê*.
- [Bracketing] indicates if the root is to be placed between parentheses, square brackets, etc.

This record owns the records {LEXEME} and {NOTES}.

2.3.2. *The record* {LEXEME}

This record consists of the following fields:

- [Lexeme] gives the orthography of a lexeme, e.g. */ktābâ/*.
- [Order] a binary field which determines the order of nominal lexemes owned by one root as follows (§2.4*):

Bits	Content	Values
0-3	suffix1	-tâ, -yâ, -nâ, -ānâ, -inâ, -unâ, -tānâ, -tunâ, -usâ, -ārâ, -qānâ, -in
4-5	suffix2	-āyâ, -itâ
6-7	suffix3	-utâ, -âit
8-9	prefix	m-, t-
10-12	1st rad. vowel	a, ā, e, i, u
13-15	2nd rad. vowel	a, ā, e, i, u
16-18	3rd/ rad. vowel	a, ā, e, i, u
19-21	4th rad. vowel	a, ā, e, i, u
22-24	total no. of vowels	
25-27	no. of radicals	bi, tri, 4, compound
28-31	derived form	p'al, ethp'el, pa"el, ethpa"el, af'el, ettaf'al, etc.

- [Seyāmê] a flag which indicates the presence or absence of *seyāmê*.
- [Bracketing] indicates if the lexeme is to be placed between parentheses.
- [Freq] a flag which indicates high frequent lexemes (such as particles and prepositions) to be listed in the appendix of the *Concordance*.

This record owns the records {WORD}, {MEANING}, {ETYMOLOGY} and {NOTES}; it is owned by the record {ROOT}.

2.3.3. *The record* {WORD}

This record consists of the following fields:

- [Word] gives the orthographic form of a non-vocalized word, e.g. */bktbḥ/*.
- [Vocalized] gives the corresponding vocalized form, e.g. */baktābeḥ/*.
- [Order] a binary field which determines the order of words under a lexeme according to morphological features as follows (§2.4*):

Bits	Content	Values
0-1	RESERVED	
2-3	suffix gender	common, masc., fem.
4-5	suffix person	third, second, first
6	suffix number	sing. pl.
7-8	suffix type	none, suffix, contraction
9-14	prefix	b, d, w, l, bb, bd, etc.
15-16	gender	common, masc., fem.,
17-18	person	third, second, first
19-20	number	sing., pl.
21-22	state	abs., constr., emph.
23-25	tense	perf., impf., impt., inf., act. part., pass. part.
26-31	form	p'al, ethp'el, pa"el, ethpa"el, af'el, ettaf'al, etc.

- [Seyāmê] a flag which indicates the presence or absence of *seyāmê*.
- [Lexical] a flag which indicates that this word instance represents the vocalization of its lexeme owner; in nominal forms, the emph. masc. sing. is marked, and in verbal forms the p'al 3rd masc. sing. is marked.

This record is a member of the record {LEXEME}.

2.3.4. *The record* {MEANING}

This record contains the following fields:

- [English] gives the key English meaning, e.g. if the meaning is 'to swallow up', this field gives the key word 'swallow'.
- [Previous] gives the words before the key meaning, e.g. 'to'.
- [Next], give the words after the meaning, e.g. 'up'.
- [Comment] allows a string for a comment.
- [Other-Info] a binary field which gives information regarding fonts (e.g. whether the strings of [Previous] and [Next] are in italic or not), and morphological details (e.g. the meaning is valid for *pa"el* forms).
- [Include] a binary field which provides fifteen flags to indicate whether a particular meaning applies to a particular project, e.g. include this meaning in the *Concordance*.

This record is a member of the record {LEXEME}.

2.3.5. *The record* {ETYMOLOGY}

This record contains the following fields:

- [Form] gives the etymological form for foreign words, e.g. '*kaisar*'.
- [Language] indicates the language of the form, i.e. Greek.
- [Bracketing] indicates if the form is to be placed between parentheses or not.

The record is a member of the record {LEXEME}.

2.3.6. *The record* {NOTES}

This record type contains one field, [Note], which allows a string. This record is a member of the records {ROOT} and {LEXEME}.

2.4. Using sorted sets*

Recall (§2.2) that the members of a set are arranged in a particular sorting order specified in the definition of the set. The sorting order depends on the value of a specific field in the member record, e.g. the field [Order] in the records {LEXEME} and {WORD}. This is used, for example, to sort lexical members of the ROOT-LEXEME set, and word members of the LEXEME-WORD set. For each lexical entry, for example, the database keeps track of the order of the word members according to the value of the field [Order] in the record {WORD}.

Assume that one wants to list words in the following precedence: form (*p'al*, *ethp'el*, *pa"el*, etc.), tense (perf., impf., etc.), state (abs., const., emph.), number (sing., pl.), person (3rd, 2nd, 1st), and so on. To do so, a *p'al* word entry should have in its [Order] field a value which is less than that of an

af'el entry; similarly, in two *p'al* entries, one perf. and the other impf., the former should have a lower value in its [Order] field than the latter.

To achieve this, SEDRA makes use of bit manipulation by splitting the bytes of [Order] into segments. Each segment is used as a sub-field (i.e. form, tense, state, etc.) in its own right. The sub-field of the highest precedence (i.e. form) is assigned to the segment consisting of the most significant bits, while the sub-field of the lowest precedence is assigned to the segment consisting of the least significant bits. Figure 4 gives a portion of the bit segmentation of [Order] in the {WORD} record (§2.3.3).

	Form		Tense	State	Num.	Person	Gen.	...
	31 30 29 28 27 26		25 24 23	22 21	20 19	18 17	16 15	14 ... 0
0	000000	N/A	000 N/A	00 N/A	00 N/A	00 N/A	00 N/A	
1	000001	P'al	001 Perf.	01 abs.	01 s.	01 3rd.	01 c.	
2	000010	Ethp'el	010 Impf.	10 con.	10 pl.	10 2nd	10 m.	
3	000011	Pa'el	011 Impt.	11 em.		11 1st	11 s.	etc.
4	000100	Ethpa"el	100 Inf.					
5	000101	Af'el	101 Part.					
		etc.	etc.					

Figure 4. Bit segmentation of [Order] in {WORD}.

2.5. Database addresses

Each letter in the local post office bears an address which points the post-person to the location of the addressee; similarly, each instance in a database has an address which points to its location in the database. Instances of different records are kept in distinct files; all entries of the record {ROOT}, for example, are stored in one file. Each file is given a unique file number; this is similar to the postal code of an address. The following table gives the file numbers of SEDRA:

Root File	0
Lexeme File	1
Word File	2
Meaning File	3
Etymology File	4
Notes File	5

A postal code, however, only indicates the region of an address; similarly, the file number only indicates in which file an entry is stored. To find the location of an entry in a file, one needs more information. Buildings on a street are normally numbered sequentially; in the same manner, entries in

a file are given a sequential slot number: the first entry is given the number one, the second two and so on. Hence, knowing the file and slot numbers of an entry, its location can be determined; for example, the address 0:5 points to the fifth entry in the 'root file', and the address 2:154 points to the 154th entry in the 'word file'.

3. TEXT DATABASES

So far we have discussed the SEDRA database which maintains the roots, lexemes, and words of Syriac. But where is the text of the NT we are trying to create a concordance for?

From the outset, it was decided that the database should be *independent* of any external data, including texts. This made SEDRA more flexible. As a result, texts are kept in separate databases.

3.1. NT database schema

The schema of a typical text database is shown in Figure 5. It consist of one record, {TEXT}, and one set. Each entry of the record represents one word in the text. The record contains the following fields:

- [Reference] gives the reference of the word in the text, e.g. Mt 5.31.7 indicates the 7th word in Mt v.31.
- [Word-Addr] gives the database address of the word in SEDRA (§2.5).
- [Punctuation] gives the punctuation mark following the word, if any.

The set TEXT-TEXT allows the record to own itself. What does that mean? This is best explained by an example: the Syriac NT contains 2551 instances of the verb */'emar/* 'say', the first being in Mt iii.7. This first instance becomes the owner of itself and all subsequent instances of the verb. This makes finding all the instances of the word in the text fast, easy, and retrieved in order of reference.

If the text database of the Syriac NT, SEDRA-BFBS (British and Foreign Bible Society's edition), is independent of SEDRA, how then can one relate entries in the two databases?

3.2. Relating the NT database to SEDRA.

Recall (§2.2) that two records can be related in the relational model through common data fields. The field [Word-Addr] in the text database gives the address of the word in question in SEDRA. Since the address of a record can be seen as one of the record's fields, the [Word-Addr] of the text database and the address of the corresponding word in SEDRA constitute common data fields. Hence, the two databases can be related in the

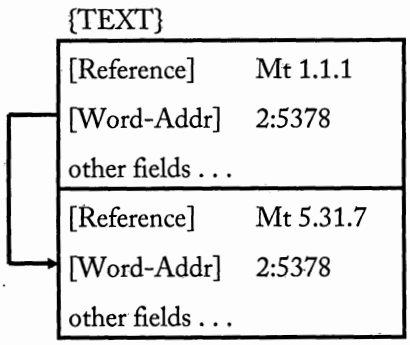

Figure 5. SEDRA-BFBS Schema.

relational model. Figure 6 shows the records of the first three words in Mt i.1 and the corresponding words in SEDRA.

SEDRA-BFBS: {TEXT}

[Reference]	Mt 1.1.1
[Word-Addr]	2:5378
[Reference]	Mt 1.1.2
[Word-Addr]	2:23454
[Reference]	Mt 1.1.3
[Word-Addr]	2:27593

SEDRA: {WORD}

At address 2:5378	ktābâ
At address 2:23454	dilīduteh
At address 2:27593	dyešu‘

Figure 6. Relating SEDRA-BFBS to SEDRA.

4. GENERATING CONCORDANCES

Now that we have a linguistic database, sc. SEDRA, and a text database, sc. SEDRA-BFBS, how can we generate a concordance?

I shall give here two accounts: an informal one aimed at general readership, and a formal one aimed at more technical readers. Both accounts describe the internal design of the concordance-generator, SyrCONC, in a brief manner.

4.1. Informal description

It was said earlier (§2.3) that SEDRA allows each root to have any number of lexeme members, and each lexeme to have any number of word members. Figure 3 (§2.3) shows an example. To illustrate the following steps by examples, let us assume that the SEDRA database contains the entries shown in Figure 3. To generate the concordance, we perform the following steps:

- 1. Retrieve the first instance of the record {ROOT}, e.g. */'skl/*, and call it the *current root*.
- 2. Output the *current root*.
- 3. Retrieve the first {LEXEME} member of the *current root*, e.g. */'eskuli'/*, and call it the *current lexeme*.
- 4. Retrieve the meanings and the etymological origin of the *current lexeme*.
- 5. Output the *current lexeme* with its meanings and etymology.
- 6. Retrieve the first {WORD} member of the *current lexeme*, e.g. */'eskuli'/*, and call it the *current word*.
- 7. Output the current word.
- 8. Search the SEDRA-BFBS database for the first occurrence whose [Word-Addr] field has the same value as the address of the *current word*, e.g. Acts xix.9, and call it the *current text* (§3.2).
- 9. Find the first member of the *current text* (§3.1) and all subsequent members; output all citations.
- 10. Find the next {WORD} member of the *current lexeme*: if any, call it the *current word*, and go to Step 7; otherwise go to the next step.
- 11. Find the next {LEXEME} member of the *current root*: if any, call it the *current lexeme*, and go to Step 4; otherwise go to the next step.
- 12. Find the next {ROOT} occurrence: if any, call it the *current root*, and go to Step 2; otherwise exit.

Readers should keep in mind that the above steps only present the general outline of SyrCONC, and not a complete algorithm.

4.2. Formal description*

The algorithm of the previous section will be described here more formally. The actual implementation of SyrCONC contains many functions dealing with various aspects (I/O, format, layout, interface) which have no place here for limitation of space. Firstly, I shall give an account of low-level

database management functions. This will be followed by the main algorithms.

4.2.1. *Database management functions*

The following low-level database management functions are assumed (data structures of record types are also assumed):

- FIRST-ROOT(), returns the first root in SEDRA (in alphabetical order).
- NEXT-ROOT(), returns the next root in SEDRA (in alphabetical order).
- FIRST-LEXEME(*rdba*), given a database address of a root, *rdba*, the function returns the first lexeme member of the ROOT-LEXEME set.
- NEXT-LEXEME(*rdba*), given a database address of a root, *rdba*, the function returns the next lexeme member of the ROOT-LEXEME set.
- FIRST-WORD(*ldba*), given a database address of a lexeme, *ldba*, the function returns the first word member of the LEXEME-WORD set.
- NEXT-WORD(*ldba*), given a database address of a lexeme, *ldba*, the function returns the next word member of the LEXEME-WORD set.
- FIND-CITATION(*wdba*), given a database address of a word, *wdba*, the function returns the first occurrence of the word in the text database, i.e. the owner of the TEXT-TEXT set (§3.1).
- FIRST-CITATION(*tdba*), given a database address of a text entry in the text database, *tdba*, the function returns the first member of the TEXT-TEXT set.
- NEXT-CITATION(*tdba*), given a database address of a text entry in the text database, *tdba*, the function returns the next member of the TEXT-TEXT set.

All functions return database addresses, or null if no record is found.

4.2.2. *The main algorithm*

The following algorithms provide key function calls. We shall proceed bottom-up.

- Finding citations. Given a database address of a word, *wdba*, the function produces all the citations of the word.

```
GET-CITATIONS(wdba)
1.  x = FIND-CITATION(wdba)
2.  t = FIRST-CITATION(x)
3.  while (t)
4.        do  output
5.            t = NEXT-CITATION(x)
```

- Processing Words. Given the database address of a lexeme, *ldba*, the following function processes the word members of the lexeme.

```
PROCESS-WORDS(ldba)
1.  w = FIRST-WORD(ldba)
2.  while (w)
3.      do  GET-CITATIONS(w)
4.          w = NEXT-WORD(ldba)
```

- Processing Lexemes. Given the database address of a root, *rdba*, the following function processes the lexeme members of the root.

```
PROCESS-LEXEMES(rdba)
1.  l = FIRST-LEXEME(rdba)
2.  while (l)
3.      do  FIND-MEANINGS(l)
4.          FIND-ETYMOLOGY(l)
            PROCESS-WORDS(l)
5.          l = NEXT-LEXEME(rdba)
```

- Processing Roots. The following function processes the roots in the database.

```
PROCESS-ROOTS()
1.  r = FIRST-ROOT()
2.  while (r)
3.      do  PROCESS-LEXEMES(r)
4.          r = NEXT-ROOT()
```

The concordance generator's main algorithm may look as follows:

```
INITIALIZE-PROGRAM()
.....
PROCESS-ROOTS()
.....
EXIT()
```

It must be stressed that the above algorithms only give a general description of SyrCONC (e.g. I/O is not shown). The actual implementation differs substantially in order to take into account side complications.

5. CURRENT RESEARCH

As of December 1992, SEDRA III contained 2050 roots, 3559 lexemes, 31079 words, 6337 meanings, 174 etymologies, and 102 notes. During 1993 all the entries from Brockelmann's lexicon were added.

Concordance generation can be improved in a number of ways: by allowing automatic morphological analysis, and minimizing the size of the database.

5.1. Automatic morphological analysis

The major task in concordance generation is the tagging of each word in the text; for example, */wdaktābaykun/* 'and your books' needs to be tagged as follows:

Root: */ktb/*
Lexeme: */ktābâ/*
Form: nominal $C_1C_2oC_3o'$ (where C_i is the i^{th} radical)
Parsing: noun, masc., pl.
Prefix: */wda-/*
Suffix: */-kun/*, masc., pl., 2nd person

Entering this information for each single word in the text is a long and tedious process. (When generating the *Concordance*, this problem was avoided since the text of the Way International was tagged, but had to be modified in some places.) If one considers generating a concordance to the OT or the works of Ephrem, this process can take years, if not decades (it took the Way International 15 years to produce their database of the NT!).

This problem is the aim of my current research at the Computer Laboratory, University of Cambridge. The research aims at designing an automatic morphological analyzer for Semitic languages using computational morphology theories. When complete, tagging of texts can be performed automatically.

5.2. Minimizing the database

As a result of having an automatic morphological analyzer, the size and structure of SEDRA will change drastically. The morphological analyzer takes as input a lexicon of morphemes and a morphological grammar, and produces all the lexemes and words automatically. As a result, SEDRA will become very small in size through eliminating the lexeme and word files.

It is hoped that SEDRA will continue to progress to serve Syriac studies.

St John's College
Cambridge CB2 1TP, U.K.
Email: george.kiraz@cl.cam.ac.uk

George Anton Kiraz