Chapter 8
Conclusion

8.1 Introduction

The main research question of the thesis, *How can XML be used to build an exploitable linguistic database of the text of the Hebrew Bible?*, directly reflects the title of the thesis. This question was answered by dividing it up into various sub-questions that correspond to the six content chapters (Chapters 2-7). Each chapter contains a theoretical part in which related literature is discussed, as well as one or more empirical experiments applying the theory in a practical case study. To conclude this research a short summary[141] of the salient points will be provided, after which the secondary research questions will be revisited to determine if and to what extent these issues have been addressed and answered.

The scope of the empirical study is limited to the Hebrew text of Genesis 1:1-2:3. The issue of scalability is not addressed in the thesis's experiments. Although small texts could easily be plugged in and analysed, assuming that the correct XML structure and codes are used for tagging, larger texts such as the whole Hebrew Bible could create memory and other implementation problems, issues which should be addressed in follow-up work. The focus of this thesis is, however, on the contribution that the use of these technologies may make for linguistic researchers, and not primarily on the implementation problems. Therefore, issues of scalability and large file structures are excluded.

[141] This summary of the thesis is an abridged and revised version of a paper accepted by the 2008 International Conference on Information Resources Management (Conf-IRM), Niagara, Ontario, Canada, May 2008 ("From Tags to Topic Maps: Using Marked-up Hebrew Text to Discover Linguistic Patterns"), submitted under the names of the candidate and his supervisors (Kroeze, Bothma and Matthee).

8.2 Summary of thesis contents

In order to create new knowledge a researcher always needs raw data. If no data is available, a data set should first be set up, which often implies the explication of tacit knowledge. In the case of languages, for example, the linguistic categories that are implicit in the text, should be spelled out in a consistent format to enable the researchers, or their computer programs, to ferret out hidden nuggets of knowledge embedded within the networks of functions on a specific level or even between various language modules. XML is a mark-up language that allows the researcher to create a unique tag set to serve his/her design purpose. XML tags contain meaningful information and therefore differ from HTML tags that are mainly used to tell an internet browser how to format and display data. To display an XML file one needs a style sheet to fulfil the same purpose. The XML tags may contain semantic information or may be used to structure the data in the file. This latter function, which may be used to simulate a database's columns and rows, is used in this project to store each clause's verse number as a primary key, the phrases in the clause, their translations, word group identifiers, syntactic and semantic functions. The XML file is a flat representation of a threedimensional data structure or data cube, in which one dimension represents the clauses, another the phrases and the third the linguistic modules.

Although the XML data set contains all the information needed for exploration, the verbosity caused by the mark-up makes it very difficult for a human researcher to spot any interesting patterns. In fact, the visibility of the structuring tags actually hides more than it reveals. Therefore, a computer program is needed to manipulate the data set to make the discovery process more human-friendly. Various software packages exist that are XML-enabled and some of these could certainly have been used for this purpose. However, in order to better understand and explain the various phases in the process, it was decided to write new code to represent and manipulate the data set in an efficient data structure in temporary computer memory (RAM). Visual Basic 6 (VB6) was used for this experiment because it allows the use of threedimensional arrays and the creation of an executable file that can easily be disseminated. The array uses indexes to number and structure the various data elements stored in it, thus eliminating the need to use variables to keep track of word and clause order.

The array data structure also facilitates procedures to represent various meaningful representations of subsets of the data. To view the data in a human-friendly way relevant subsets should be extracted from the data cube in the computer's memory. The concepts of slicing and dicing have been borrowed from data warehousing theory and practice. Although a typical data warehouse contains aggregated data, while the suggested linguistic data cube contains detailed data, the data structure is also threedimensional. Slicing off one layer from the Gen. 1:1-2:3 data cube reveals a twodimensional subset that can easily be represented on paper or a computer screen. The most obvious slices of a linguistic data cube are those where each slice shows one clause's multidimensional analysis. Stacking all of these slices onto each other will rebuild the cube. Again, array processing is used to slice each clause's integrated analyses as a twodimensional table. Dicing refers to the unveiling of a specific piece of data within the data set. It is used in this project for search functions. The program can search the array to find a required clause number, a whole phrase or a part of a string; when found, all the information related to the relevant clause is displayed.

Although it logically makes more sense to start with a databank in XML format and then move on to the representation of the data in the computer's temporary memory, the order of these research sections has, actually, been reversed in this study. Due to the verbosity of XML it was easier to first hardcode the clause cube in an array structure and to create the first version of the XML file programmatically.

Once XML has been identified as a suitable medium for permanent storage of the data, and a threedimensional array for temporary, in-memory storage, one has to find a way to convert the data between the two mediums (round-tripping). Using the string processing facilities of VB6 the text-based XML file is opened within the program. Each line is read from the file using a loop with the same number of iterations than the number of clauses in the file. Line by line the XML tags are stripped away and the remaining linguistic data is stored into the array's variables. It is essential that the program code should reflect the structure of the XML file exactly to ensure that the linguistic data is inserted into the correct places within the array. If the data is only used for viewing purposes or as a data source for more complex manipulation, the contents of the array remain unchanged and do not have to be re-saved on permanent storage. The XML databank may simply be opened and reused again at a later stage. When the program does save the data and the resulting version is an exact copy of the source file, the process is called ideal round-tripping.

From a practical point of view, however, it does not make sense to replace existing data with an exact copy. But users will often feel the need to change or update the data itself. Having the data in an array in memory makes it quite easy to add other typical database functionalities (create, update and delete), which could typically be used by end-users to refine or expand the data set.

After editing, the data should be stored on the hard disk again for retrieval at a later stage. This process is a reversal of the method discussed above. The text file is opened and the old data overwritten. The algorithm adds the relevant XML tags to the linguistic elements before they are appended to the new file. One could also give the resulting file a different name to ensure the retention of the original data that may be used as a backup when necessary. Even if the data is changed by the user, the basic structure of the XML file should remain exactly the same. Adding or removing clauses, will, however, change the number of loop iterations needed when reading or writing the data from and to the XML file. Currently, a new XML version of all the data is created when data is edited by end-users, and a more elegant solution should be researched in order to allow incremental updates.

The slicing and dicing features referred to above, however useful, could easily have been simulated by a simple HTML or word processing implementation as well. The real power of the data cube is only demonstrated when one realises that the same data set may now be used to perform other and more advanced processing functions. Finding new patterns in the linguistic data may be regarded as an example of text data mining.

Another meaningful slice is one that isolates the semantic layer. This allows the researcher to study the various combinations of semantic functions or semantic role frameworks. In this experiment a procedure is used to identify unique frameworks and to count their frequencies. It first slices off the semantic layer and sorts the elements in each row; after concatenating each row to a string, the rows are ordered and the unique frames counted. The researcher then uses this information to compare existing definitions of semantic functions. Even though the experimental data set is very small, a number of interesting combinations indicates that the definitions of some of the semantic functions should be revised, at least for Biblical Hebrew. For example, one instance was found of the semantic function of purpose which is embedded in a state predication. According to the current definition of purpose this should not have been possible since purpose may only occur in controlled predications (actions and positions). Another interesting result prompts the grammarian to reconsider the possibility of manner occurring in states. Another experiment looks at the mapping of semantic functions onto syntactic functions and also prompts interesting hypotheses.

These procedures illustrate how the process of text mining may catch up on gaps in existing linguistic knowledge, for example when a general linguistic theory is applied to and tested on a specific language. The results of these experiments represent new information that may be used to create knowledge, but they are rather static. The user is presented with one set of semantic frameworks in a text-based format and (s)he should work through the document in a linear way.

An interactive visualising tool enables the researcher to look at a data set from various perspectives. An example of such a tool is a graphical topic map that shows all the relationships between phrases, semantic roles and syntactic functions in the data set. A new program, using the same XML data set, was created to facilitate link analysis between these nodes.[142] By adding or deleting filters the user may focus on subsets within the numerous relations. A filter may be selected, for example, to

[142] This graphical topic map tool was created by Jan C.W. Kroeze, a student at the University of Pretoria.

isolate all phrases with the semantic function of product. The graph that is produced shows that, in Gen. 1:1-2:3, the semantic function of product is realised by the syntactic functions of object and complement. When the user hovers with the mouse over one of the phrases, more clause detail is shown in a tool-tip.

The experiments conducted in this thesis demonstrate the process of creating knowledge by means of the interpretation of information extracted by digital tools from raw linguistic data. It proves that XML is a suitable mark-up language to build a threedimensional data structure that captures information of various language modules. It shows that array technology provides an efficient way, not only to round-trip this data to and from computer memory, but also to view and manipulate it in order to reveal linguistic relations embedded in the marked-up data. It explores graphical visualisation as a powerful, experimental way to search for patterns in the data set.

8.3 Revisiting the research questions

Taking into account the overview of the thesis given above, the secondary research questions that have been mapped to Chapters 2-7 will now be revisited.

Chapter 2 addressed the following research question: *How can multidimensional Biblical Hebrew linguistic data be captured and stored in the computer's temporary memory using a programming language such as Visual Basic?*

This chapter experimented with a threedimensional data structure, using Visual Basic 6, and found that a threedimensional array could be used to represent inherently multidimensional linguistic data regarding Biblical Hebrew clauses. Various layers of linguistic knowledge can be integrated by stacking various modules of analysis onto each other. On a linguistic level the corresponding elements are linked by means of the phrases constituting the clauses. On a programmatic level they are linked by means of the indexes which are inherent and essential to the array structure. An array, as such, cannot be stored permanently on a hard disk. Therefore, a VB6 program module was used in the initial phases of this project to declare and hardcode the contents of the array.

Chapter 3 addressed the following research question: *How can multidimensional Biblical Hebrew linguistic data be processed with a programming language such as Visual Basic?*

This chapter explored data warehousing and online analytical concepts to find ways to render meaningful subsets of linguistic data stored in a threedimensional array. Concepts like slicing and dicing were adjusted to make them useful for the processing of linguistic data. The captured data can be viewed and manipulated in various ways, for example to create stacks of twodimensional interlinear tables showing required aspects of clauses' data.

Chapter 4 addressed the following research question: *How can multidimensional Biblical Hebrew linguistic data be stored permanently to allow a stable environment for editing and processing?*

This section tried to find a more elegant solution for the permanent storage of the databank. Due to its flexibility XML technology was chosen to build a text-based databank. The experiment indicated that XML is indeed a very suitable mark-up technology that can be used to permanently store the linguistic data in a separate databank because it allows users to create their own tag sets which may simulate a multidimensional database structure. Due to XML's platform-independency it also enables the re-use of the data on other platforms. Various layers of linguistic data were captured in an XML document using the phrase as the basic building block of the data cube. XML's inherent hierarchical nature was used to add parallel levels of linguistic modules that are linked to the primary building blocks. This implies that more layers may be added when needed. Currently, the following modules are tagged: a phonological representation and phrasal translation, as well as the morpho-syntactic, syntactic and semantic analyses of the various phrases.

Chapter 5 addressed the following research question: *How can linguistic data be recovered from and saved to a permanent storage device (such as an XML database)?*

In order to satisfy the requirement of finding a stable platform for the data, while also allowing editing and advanced processing, round-tripping was investigated, i.e. the procedures needed to import the data into the VB6 program and to export it again to an external storage medium (XML file). The procedures made extensive use of string processing. Various viewing and searching functions were discussed. In addition, create, update and delete functionalities were added to enable users to populate and edit the clause cube while it is in the array state and to save these updates both to the RAM and on permanent storage in XML format. These processes proved to be suitable for the efficient storage, transfer and processing of linguistic data.

Chapter 6 addressed the following research question: *How can linguistic data be explored to unveil hidden patterns in and between the various language modules?*

This chapter focused on the benefits of text data mining facilitated by the preceding technologies. Some data mining concepts were applied in two experiments by aggregating aspects of the semantic and syntactic modules tagged in Genesis 1:1-2:3. A computer-assisted exploration of the semantic data captured in the XML database of Genesis 1:1-2:3 illustrated the rigour enforced by such a text-mining venture. It also prompted a few hypotheses regarding the definition of semantic functions. The second experiment investigated the mapping of semantic and syntactic functions that were revealed. This endeavour reinforced the finding of a rigorous research method referred to above. Not only did it point out more tagging errors, but it also challenged existing assumptions about the syntactic and semantic theories that have been applied to the Hebrew text.

Chapter 7 addressed the following research question: *How can visualisation be used to enhance text-mining of multidimensional linguistic data?*

In this chapter, projects have been suggested (one of which was implemented) that could use the XML-based data cube of Genesis 1:1-2:3 in visualisation ventures to clearly show linguistic patterns uncovered by means of a computer program. The experiment to use an interactive graphical topic map to explore the mapping of syntactic and semantic functions showed that visualisation may be used to create user-friendly interfaces that may facilitate easier and more intuitive mining of linguistic data.

Since all secondary research questions have been answered satisfactorily, one may also conclude that the main purpose of the thesis has been attained. Although the candidate believes that all knowledge is provisional, there are exciting indications that XML may be used as a permanent and independent platform for the permanent storage and integration of linguistic data. Such a databank may again be used by various algorithms, programming languages and visualisation techniques for in-depth processing.

8.4 Future research

Regrettably, however, not all issues can be covered in one research project. In fact, each piece of research creates new problems and raises more questions that need to be addressed. Therefore, there are various serious issues that need to be investigated in follow-up work. Some of these problems are:

- Embedded phrases and clauses create challenges for the preservation of the original word order. More elegant possibilities could be researched to replace the current solution to put embedded clauses in brackets and then analyse them separately.
- Scalability issues needs to be investigated to facilitate the processing and visualisation of aggregate data of larger sections.

- The results of other existing Biblical information systems should be re-used and integrated rather than creating and tagging a whole new dataset. Transformation of the data into these systems will probably give rise to compatibility problems.
- The storage of aggregated linguistic data in the data cube should be researched. Linguistic data, for example, can be drilled up or down along the lines of a syntactic tree structure, and this issue should be explored in further research. A separate dimension could, for example, be declared for each module. Each dimension could contain various levels to capture the hierarchical data of a specific module. (Currently, all language modules contain detailed data only and form part of the cube as layers of one dimension.)
- More advanced search facilities could be implemented. It should, for example, also be possible to do *ad hoc* searches on two (or more) parameters through the clause cube in order to find unusual combinations such as the use of the object marker *'et* to express adjuncts,[143] or for strange syntactic-semantic mappings such as a (prepositional) complement expressing a patient.
- A typical clause cube will contain much repetition because the same syntactic and semantic functions and structures are used over and over again. Depending on the number of dimensions a lot of empty cells can also be present (sparse data). With reference to numeric and other rigidly structured data '[t]he ability of a multidimensional DBMS to omit empty or repetitive cells can greatly reduce the size of the cube and the amount of processing' (Connolly and Begg, 2005). The applicability of this feature to linguistic data should be explored. The denser the data can be organized the more efficient the storage and processing will be.
- Since the use of multidimensional database management systems and XML query languages may supply exciting, alternative storage and processing possibilities, these should also be experimented with. Software is currently being developed to transform 'XML documents to and from databases'

[143] Cf. Gesenius, Kautzsch & Cowley (1976: 372-376).

(Kroenke, 2005). The use of XML technology will also facilitate the transmission of clause cube data over the internet (ibid).[144]

Despite these and other areas that still need to be researched, the candidate trusts that this thesis makes a contribution to the field of computational linguistics by illustrating how Linguistic Information Systems, as a humanistic endeavour, may be implemented to further enhance the impressive body of work that has already been done during the past forty years in Biblical Hebrew computing. Although these findings may only be valid with reference to the small experiments of this study, one may conclude by expressing one's trust and hope that there are promising indications that the use of linguistic information systems in larger texts could lead to a variety of knowledge-creation ventures.

[144] The internet was the stimulus for the confluence of document processing and database technology, facilitated to a great extent by XML (Kroenke, 2004).