Journal of Northwest Semitic Languages 30/2 (2004), pp. 99-120

Jan H Kroeze (University of Pretoria)

TOWARDS A MULTIDIMENSIONAL LINGUISTIC DATABASE OF BIBLICAL HEBREW[1]

ABSTRACT

Biblical Hebrew clauses can be and have been studied from many different angles. Over the past forty years much of this knowledge has been captured in various computer software systems and databases. Having all these electronic aids for the study of the Hebrew Bible is wonderful, but also overwhelming and even frustrating, because various tools have to be used to study different layers and to get various perspectives. Therefore, systems have been suggested or are being developed to display multi-layer analyses of Hebrew clauses, integrating the various dimensions of clausal analysis in an interlinear table format on one screen. This type of interlinear table is in fact a two-dimensional subset of three- (or multi-)dimensional linguistic data structures. The knowledge that is represented by a collection of interlinear tables can be conceptualised three-dimensionally as an information cube consisting of a cluster of clauses and analyses. Such a clause cube can be implemented on a computer using a three-dimensional array, which can be called a cyber cube. Processing arrays with nested loops makes it possible to view and manipulate the stored information in an efficient way.

1. INTRODUCTION

Biblical Hebrew clauses can be and have been studied from many different perspectives. These perspectives or layers mirror the "modules" of the human mental language machine (cf. Van der Merwe 2002:89). Over the past forty years much of this knowledge has been captured in various computer software systems and databases.[2] Van der Merwe (2002:96-97) refers to some of these products. The most basic layer is the digital representation of the Hebrew text, which can be called the transliteration layer. The second layer is the phonological layer, followed

---

1 This article is a revised and extended version of a paper read at the AIBI VII conference, Leuven, July 2004 ("Processing Hebrew clauses using three-dimensional arrays").

2 Cf. Talstra (1989:4). In 1987 ten machine-readable versions of the Masoretic Text and various Bible concordance programs already existed (Hughes 1987:343-384; 498-545).

by the morphological, morpho-syntactic and syntactic layers.[3] More advanced layers such as the semantic and pragmatic layers have received less attention, but it is very probable that knowledge bases and expert systems that deal with these layers will, increasingly, become available. Compare Link (1995) who proposes an algebraic perspective on the semantic analysis of human language and the computerised version of Dik's functional grammar for English, French and Dutch (Dik 1992). Van der Merwe (2002:94) suggests the use of the notions *topic* and *focus* to mark-up pragmatic functions in Biblical Hebrew.

From these suggestions it is already clear that there are two main approaches in creating computerised biblical information systems. According to Talstra (1989:2), the ideal linguistic database should be created by programs applying imitated rules, "otherwise a database of biblical texts will consist only of an echo of a personal, subjective knowledge and contain linguistic information not being produced by rules but by arbitrary personal choice."[4] Ultimately, however, this is an unattainable goal, because subjectivity will also influence the formulation of the linguistic rules that are to be imitated. Even Talstra & Postma (1989:20) had to admit that it is impossible to formulate and refine rules that will attain a correct analysis in all cases. Therefore, there should also be a place for systems that capture the tacit knowledge that exists in experienced exegetes' heads. Database solutions that capture existing linguistic data can fill this gap. Chiaramella (1986:129) also refers to the "strong discussion about the best way to store knowledge" (either data structures or procedural objects) and says that "successful experiments have been made for both." This article follows the second route by proposing a database that integrates multi-modular clausal analyses.

---

3 Sowa (2000:182) refers to morphological, syntactic and semantic parsing as stages in analysing a natural language sentence, saying: "Each of the three stages in sentence processing depends on a repository of linguistic knowledge." The proposed multidimensional linguistic database of Hebrew clauses can be regarded as such a repository, which integrates various layers of analysis. Also cf. Hughes (1987:497).

4 "Experimental results in cognitive psychology suggest that humans apply model-based reasoning for problem solving in a variety of domains. Consequently, a formalism that captures the representations and processes associated with model-based reasoning would facilitate the implementation of computational reasoning systems in such problem solving domains." (Glasgow & Malton, s.a.:31).

2. *THE NEED FOR INTEGRATION*

Having all these electronic aids for the study of the Hebrew Bible is wonderful, but also overwhelming and even frustrating, because various tools have to be used to study different layers and to get various perspectives. Therefore, systems have been suggested[5] or have been developed to display multi-layer analyses of Hebrew clauses, integrating the various dimensions of clausal analysis in an interlinear table format on one screen. Van der Merwe (2002), for example, suggests the use of hypertext as one possible solution to integrate various perspectives or exegetical approaches. However, it could be very difficult or even impossible to integrate all available analyses due to the huge differences in the authors' assumptions and points of departure. Compare Anderson & Forbes (2002), who demonstrate the various divergent approaches even on elementary layers such as morphology or parts of speech. A possible solution is to show the various analyses in a parallel manner, leaving the final decision to the user. Also compare De Troyer's (2002) plea for integrated biblical tools, which implies that such a semi-integrated tool could be very useful to scholars.

Interlinear tables resemble the tables found in relational databases that capture data about entities, and this gives birth to the wish to be able to do *ad hoc* queries on the stored data. A database allows easy access to the data and the possibility of adding new data easily (Tov 1989:90). This is not possible with flat files[6] or text files. RDBMSs'[7] structural and data independency feature facilitates these requirements. In order to work dynamically with the stored data, it is important to use a proper database management system, which facilitates the use of linked files and complicated search and sorting functions (Nieuwoudt 1989:102).

3. *A CLAUSE CUBE AS THE IDEAL DATA STRUCTURE*

However, these interlinear tables cannot simply be transformed into relational database tables, because there is a separate table for each record

---

5 Compare, for example, Kroeze (2002).

6 Although flat files may have rows and columns and thus look like relational database tables, they do not support relational operators such as joins, projects and selects (Hughes 1987:497).

7 Relational database management systems.

(or clause) and the rows do not represent unique records.[8] A closer inspection of an interlinear table reveals that the rows actually represent various dimensions or layers of data-analysis that are strongly linked to the elements in the upper row. This type of interlinear table is in fact a two-dimensional subset of three- (or multi-)dimensional linguistic data structures. According to Koutsoukis *et al.* (1999:7), a stack of two-dimensional spreadsheets (rows and columns) is a three-dimensional cube.[9] This can be conceptualised as a data structure that consists of a set of cubes arranged according to rows, columns and depth layers (Figure 1).

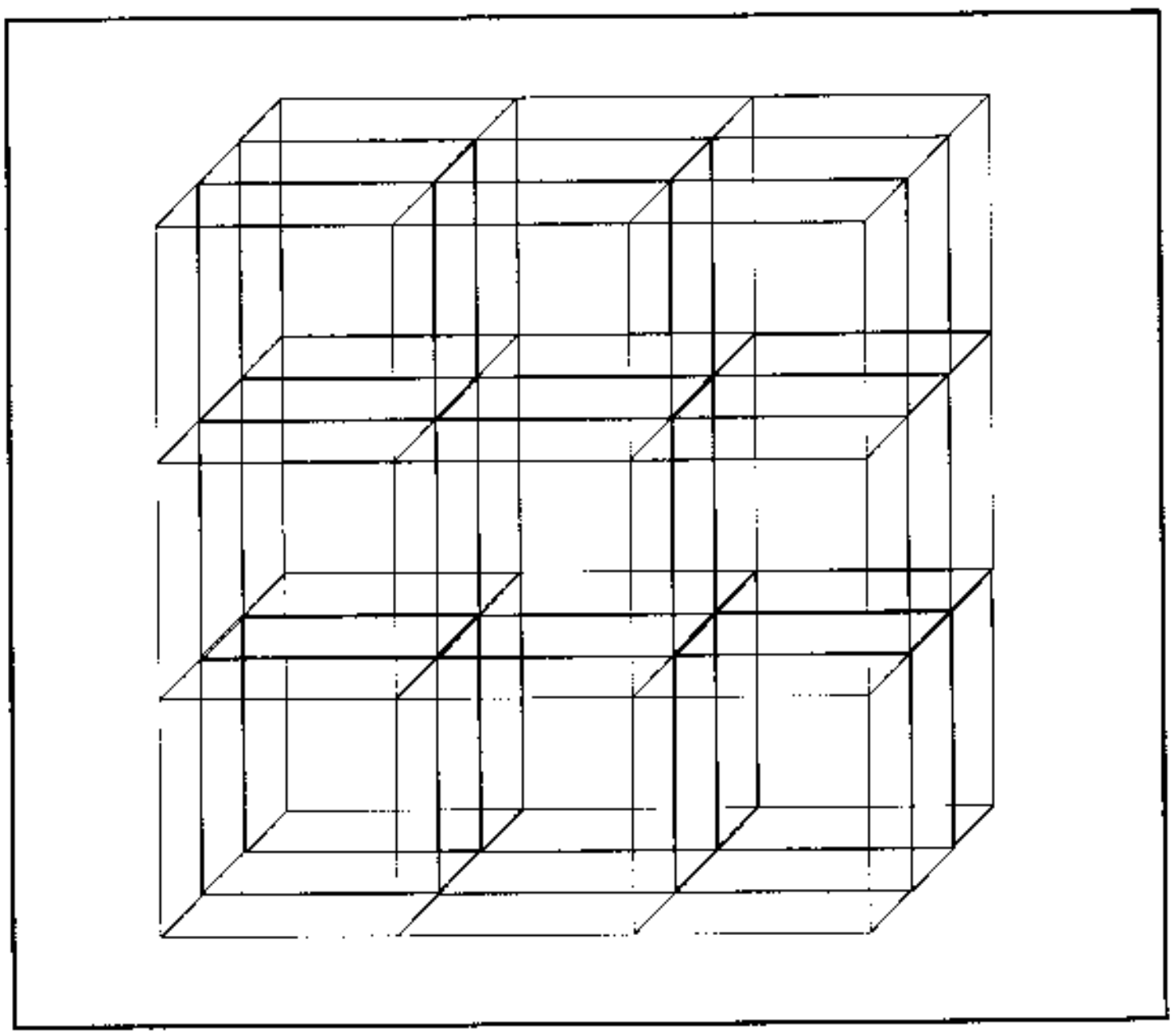

Figure 1. A three-dimensional data structure that consists of a set of 27 sub-cubes arranged according to three rows, three columns and three depth layers.

8 Chiaramella (1986:122) identified the problem of representing text in relational, hierarchical and network database management systems: "Nothing currently exist [sic] for efficient description of texts within database systems."

9 Compare Pietersma's (2002:351) discussion of an interlinear Greek-Hebrew text. According to Pietersma an interlinear text is two-dimensional because it has a vertical and horizontal dimension.

The linguistic knowledge that is represented by a collection of interlinear tables can therefore be rendered three-dimensionally as a clause cube consisting of a cluster of phrases and their analyses (see Figure 2). The horizontal dimension is divided into rows representing the various clauses – each row is a unique record or clause. The vertical dimension (columns) represents the various word groups in the clauses. Having attributes in this dimension called phrase 1, phrase 2, phrase 3, etc., at first does not seem very informative, especially if one is used to the descriptive attributes typical of relational databases. However, "it is crucial to preserve the document structure (books, chapters, verses, half-verses, words) of the data, to allow access in terms of traditional categories" (Talstra 2002:4). And this method seems to be the most straightforward way to preserve word order. Yet, the combination of these obvious attributes on the horizontal and vertical dimensions with the layers on the depth dimension is indeed very illuminating. The depth dimension represents the various layers of analysis, e.g. graphemes, syntactic functions and semantic functions. These features can be called the members of the layer dimension. The unique intersections of the members of the various dimensions are the cells, and the contents of the cells the measures (Chau *et al.* 2002:216). As in business data "the dimensions provide a 'natural way' to capture the existing real-world information structure" (Koutsoukis *et al.* 1999:11).

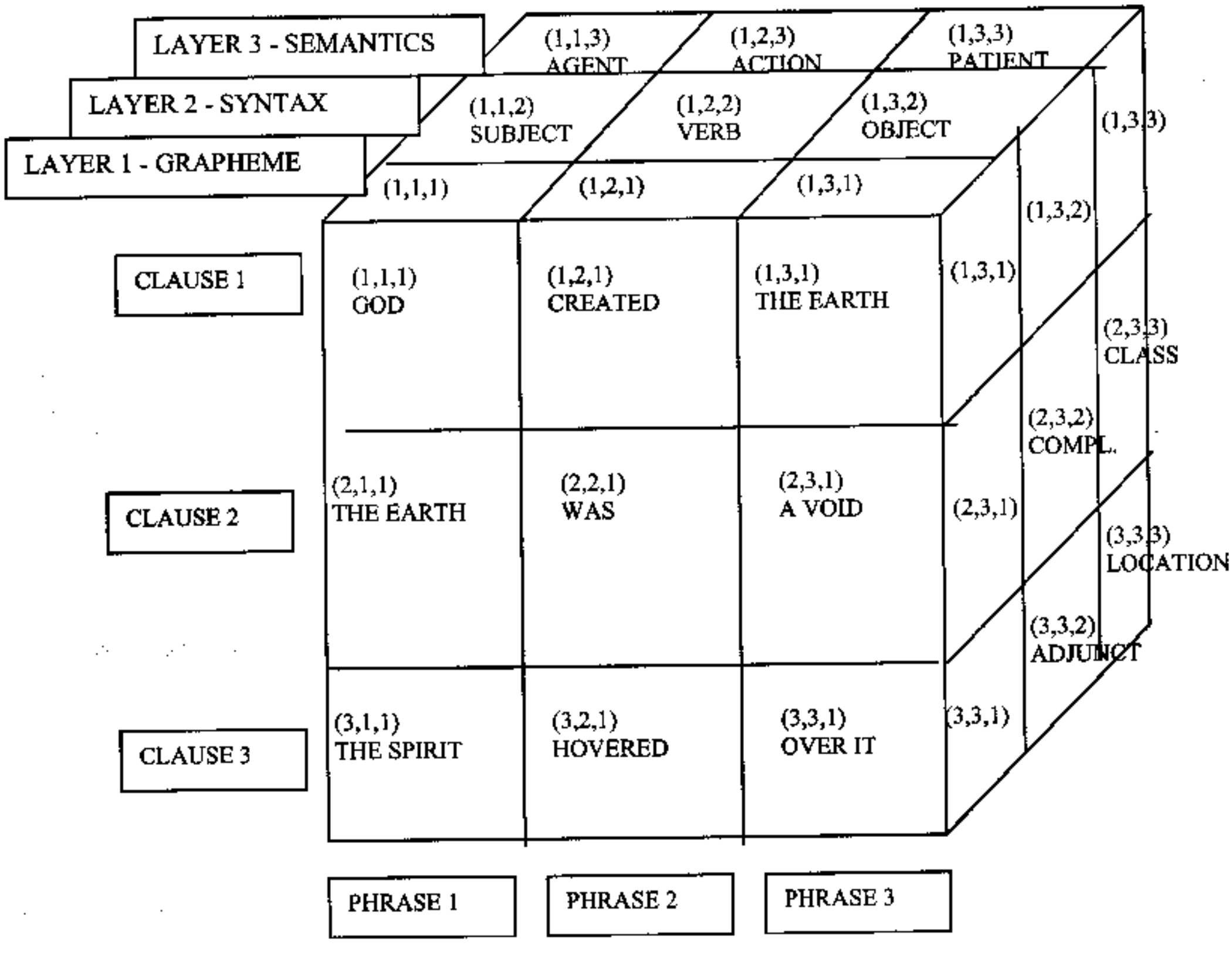

Figure 2. The knowledge that is represented by a collection of interlinear tables can be rendered three-dimensionally as a data cube consisting of layers of clauses and analyses stacked on top of each other.

4. *IMPLEMENTING THE CLAUSE CUBE IN CYBER SPACE*

Such a clause cube can be implemented on a computer using a three-dimensional array, which can be called a data cube or cyber cube. An array can be used as a knowledge representation scheme, which models entities and the relations between them in a certain problem domain, and array functions are used to generate, inspect and transform these representations (Glasgow & Malton s.a.:8). Arrays have probably already been used in many biblical information systems, for example, to sort sets of lemmatised language into sets of "identically parsed items" (Hughes 1987:502).

A data cube can easily be created in many computer languages by declaring a multi-dimensional array, e.g. in Visual Basic 6,[10] a data cube with 3 rows, 3 columns and 3 layers is declared by the following statement: "Public Clause(3,3,3) As String."[11]

The code to create such an example clause cube using a three-dimensional array in Visual Basic 6 is shown below. It captures linguistic data describing the first 14 clauses in the Hebrew Bible (Gen 1:1-5). Only the first clause is shown here (see Addendum A for the complete code). The first dimension represents the 14 rows of clauses, the second dimension the phrases with a maximum of four per clause, and the third dimension represents the layers of analysis, as follows:

- Layer 1: Clause number
- Layer 2: Transcription
- Layer 3: Translation
- Layer 4: Phrase type
- Layer 5: Syntactic function
- Layer 6: Semantic function

```
Option Explicit
Public Clause(1 To 14, 1 To 4, 1 To 6) As String
Sub Main()
Clause(1, 1, 1) = "Gen01v01a"
Clause(1, 1, 2) = "bre$it"
Clause(1, 1, 3) = "in the beginning"
Clause(1, 1, 4) = "PP"
Clause(1, 1, 5) = "Adjunct"
Clause(1, 1, 6) = "Time"
Clause(1, 2, 1) = "-"
Clause(1, 2, 2) = "bara"
Clause(1, 2, 3) = "he created"
Clause(1, 2, 4) = "VP"
Clause(1, 2, 5) = "Main verb"
```

10 Visual Basic was chosen as programming language for this experiment because it allows for more than three dimensions in arrays, as well as for extensive connectivity to database management systems (Anderson 2003:59, 116). These features will probably have to be used in more detailed and complex versions of the clause cube.

11 Such an array can be visualised as a "cube of side length m subdivided into m^3 unit cubes" (cf. Banchoff 1996:15).

```
Clause(1, 2, 6) = "Action"
Clause(1, 3, 1) = "–"
Clause(1, 3, 2) = "elohim"
Clause(1, 3, 3) = "God"
Clause(1, 3, 4) = "NP"
Clause(1, 3, 5) = "Subject"
Clause(1, 3, 6) = "Agent"
Clause(1, 4, 1) = "–"
Clause(1, 4, 2) = "et ha$amayim ve'et ha'arets"
Clause(1, 4, 3) = "the heaven and the earth"
Clause(1, 4, 4) = "NP"
Clause(1, 4, 5) = "Object"
Clause(1, 4, 6) = "Patient"
...
End Sub
```

Cyber cubes are usually used as "data cubes" to implement multidimensional databases or data warehouses [12] to enable users to "explore and analyse a collection of data from many different perspectives, usually considering three factors (dimensions) at a time" (Kay 2004). According to Kay (2004), "we can think of a 3-D data cube as being a set of similarly structured 2-D tables stacked on top of one another." In our case the data cube consists of the various interlinear clause tables all linked together in one data structure in order to enhance the analytical possibilities. Such a data warehouse is a database solution that can capture and integrate linguistic data from various sources.

One of the benefits of multi-dimensional arrays is the use of indexes referring to the specific position of a piece of data. These indexes can be used to extract subsets of the data very efficiently and quickly (cf. Kay 2004). Therefore, multidimensional arrays form the basis for multi-dimensional online analytical processing tools (OLAP). The possibility to do *ad hoc* queries is one of the essential characteristics of OLAP (Karayannidis & Sellis 2003:157). In business data cubes are used for multi-dimensional queries, e.g. how many of a certain product were sold in a specific period in a specific place? (See, for example, Marchand 2004:3.)

---

12 A data warehouse is a multidimensional analytic database that "links otherwise disparate data items" and "allows for customised user views of the data" (Koutsoukis *et al.* 1999:3).

Some programming languages, such as Visual Basic 6, even allow for the use of multi-dimensional arrays, which could represent a hypercube of clauses.[13] Such a 4-D cube consists of a series of 3-D data cubes (Kay 2004). In our application a fourth to sixth dimension could be used to break down clause constituents hierarchically into their smallest parts,[14] e.g. the NP *et-hashamayim ve'et ha'arets* (Gen 1:1) consisting of 2 NPs and a conjunction, the 2 NPs each consisting of an object marker and NP, which again consists of an article and a noun. The higher-level attributes, which represent summarised values, are called aggregates, while the lower-level attributes are called grouping attributes (Lee *et al.* 2003:124).

Although it is very easy to add another dimension (e.g. "Public Clause (3,3,3,3) As String") or even more dimensions in cyber space, it becomes more difficult to visualise these types of data structures. Multi-dimensional arrays have another downside. With every dimension added the number of memory spaces needed increases exponentially, e.g. a 3x3 table needs 9 spaces, a 3x3x3 data cube needs 27, and a 3x3x3x3 hypercube needs 81.[15] The more dimensions the hypercube has, the sparser it becomes: more and more cells are empty and this wastes memory and processing time. Although compression techniques do exist to manage the problem of sparsity, they tend to destroy the multi-dimensional data structure's natural indexing (Kay 2004).[16]

Due to huge space implications in the computer's memory and the difficulty of visualising four or more dimensions, I will here restrict

---

13 In geometry, a hypercube is a basic four-dimensional structure having 16 corners and "consisting" of (bounded by) 8 cubes. A cube is a basic three-dimensional structure and has 8 corners and "consists" of 6 squares. A square is a basic two-dimensional object and has 4 corners and "consists" of 4 lines or segments. A line is the segment between two points, a basic one-dimensional object with no corners. A point is a zero-dimensional object (Banchoff 1996:9).

14 Glasgow & Malton (s.a.:31) found that "an array representation scheme provides an effective and efficient means for spatial reasoning" and suggests that more research should be done to test its applicability to other domains including hierarchical worlds.

15 Cf. Banchoff (1996:15).

16 To solve this problem Karayannidis & Sellis (2003:156-157) proposed a chunk-based storage manager for OLAP data cubes that is both space conservative and uses a location-based data-addressing scheme. This system is also able to capture hierarchical data.

myself to three dimensions. Because it is possible to declare the exact number of rows, columns and depth layers of a three-dimensional array, enough members can be created on the depth dimension to store all modules of clausal analysis.[17]

In order to save space, as an alternative to adding more dimensions for hierarchical data, as suggested above, more members could be added on the depth dimension and symbols allowed to occupy more than one cell in each member of the array (cf. Glasgow & Malton s.a.:7, 13). In our problem space the typical hierarchical Chomskyan tree structure of the syntactic structure of a clause could be represented by such an array structure, as follows (see Figure 3):

<table>
<tr><td colspan="7">NP ((1,1,1), (1,1,2), (1,1,3), (1,1,4), (1,1,5), (1,1,6), (1,1,7))</td></tr>
<tr><td colspan="3">NP ((1,2,1), (1,2,2), (1,2,3))</td><td>Particle (1,2,4)</td><td colspan="3">NP ((1,2,5), (1,2,6), (1,2,7))</td></tr>
<tr><td>Particle (1,3,1)</td><td colspan="2">NP ((1,3,2, (1,3,3))</td><td>Particle (1,3,4)</td><td>Particle (1,3,5)</td><td colspan="2">NP ((1,3,6), (1,3,7))</td></tr>
<tr><td>Obj. marker (1,4,1)</td><td>Article (1,4,2)</td><td>Noun (1,4,3)</td><td>Conjunction (1,4,4)</td><td>Obj. marker (1,4,5)</td><td>Article (1,4,6)</td><td>Noun (1,4,7)</td></tr>
<tr><td>et (1,5,1)</td><td>ha- (1,5,2)</td><td>shamayim (1,5,3)</td><td>ve- (1,5,4)</td><td>‘et (1,5,5)</td><td>ha- (1,5,6)</td><td>‘arets (1,5,7)</td></tr>
</table>

Figure 3. A representation of a hierarchical syntactic structure using various members of the same dimension and allowing measures to occupy more than one cell of a member.

Other kinds of technology exist to implement multi-dimensional databases, such as relational online analytical processing systems (ROLAP), which are collections of cuboids or two-dimensional relational tables and do not suffer as much from the sparsity problem, but they do not have implicit indexes (Kay 2004). Although this technology can be researched to evaluate its suitability for solving our problem, I expect that, due to the rigorous table structures that are inherent in relational databases, this option does not lend itself as well as multi-dimensional arrays to capture and extract clausal data. According to Koutsoukis *et al.*

17 An alternative approach is followed by Koutsoukis *et al.* (1999:12) who combines sparse dimensions (year and season) "to create a 'conjoint dimension.'"

(1999:6) "MDDBs[18] are better suited for OLAP-type applications because of their structure and embedded functionality."[19]

One difference between a business data cube and a clausal data cube is that the former contains data that have already been processed and aggregated (Kay 2004), while a clause cube contains the basic raw data. However, Karayannidis & Sellis (2003:157) argue that, in order to support *ad hoc* queries, users should be able to drill down "to the most detailed data in order to compute a result from scratch." It could, therefore, contain hierarchical data consisting of both raw and aggregated data. Compare Chau *et al.* (2002:214): "The contents of a data warehouse may be a replica of part of some source data or they may be the results of preprocessed queries or both." A clause cube that contains hierarchical data, such as syntactic tree-structure information, will be similar to such a business data cube. Another important similarity between a business data cube and a clausal data cube is that both types of data are stable. They do not get updated or changed like data in an online transaction processing system. OLAP was developed to focus on powerful analysis of business data, rather than on the fast and efficient capturing of transaction data. These characteristics support the hunch that this technology is very suitable for the storing and analysis of clausal data.

5. *BUILDING AND USING A MULTI-DIMENSIONAL DATABASE FOR BH*

To build a clause cube one could integrate the results of various computerised clausal analysis systems. The process that one should follow is similar to the steps used for building a data warehouse, i.e. (Chau *et al.* 2002:216):

- Extraction of data from existing databases and flat files;
- Cleaning and integration of data;
- Loading of data in the data cube or hypercube;
- Transformation of data into a format that is suitable for graphical user interface.

---

18 Multidimensional databases.

19 Compare Cheung *et al.* (2001:2) for a summary of the advantages and disadvantages of both ROLAP (relational online analytical processing) and MOLAP (multidimensional online analytical processing) – they propose a combination of the two approaches. For an alternative solution compare Chun *et al.* (2004).

Once a proper multi-dimensional data warehouse has been designed and created, it can be populated using data from existing marked-up products: hypertext into hypercube! Products using the mark-up language XML[20] are especially suitable for this purpose, because the XML tags can be used to convert free text into a database. "Unlike HTML, XML is meant for storing data, not displaying it" (Holzner 2004:40). Using XML to convert existing texts into data sources for a Biblical Hebrew linguistic data warehouse will necessitate co-operation, even more than when using HTML to tag hypertext (see Bulkeley, 2002: 649), especially if the various sources are to be integrated properly.

Combining nested loops with three-dimensional arrays makes it possible to process the stored information in an efficient way. For example, it becomes possible to slice-and-dice the cyber cube of clauses to reveal various dimensions. Slicing the cube from the front reveals the Hebrew text, syntactic frameworks, semantic frameworks, etc. Slicing the cube from the top reveals subsequent clauses' multi-layer analyses. One can also drill down into the cube to reveal other information that is linked to a specific cell.

6. CONCLUSION

This experiment with a three-dimensional data structure indicated that a three-dimensional array could be used to represent inherently multi-dimensional linguistic data regarding Biblical Hebrew clauses. The various layers of linguistic knowledge that have been captured in various computer software systems can be integrated and used in an efficient way using a three-dimensional database. The captured data can be viewed and manipulated in various ways, for example, to create stacks of two-dimensional interlinear tables showing required aspects of clauses' data. In this way the three-dimensional array facilitates actions that are typical of online analytical data processing and data warehousing.

BIBLIOGRAPHY

Andersen, F I & Forbes, A D 2002. What Kind of Taxonomy is Best for Feeding into Computer-Assisted Research into the Syntax of a Natural Language? in: Cook, J (ed.) 2002, 23-42.

---

20 XML (eXtensible Markup Language) can be regarded as a subset of SGML (Standard Generalized Markup Language) (DeRose 1997:186, 233, 235).

Anderson, T 2003. *Visual Basic in Easy Steps: Fully Updated for Version 6.* Warwickshire: Computer Step.

Banchoff, T F 1996. *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions.* 2nd Edition. New York, N.Y.: Scientific American Library.

Bulkeley, T 2002. Commentary Beyond the Codex: Hypertext and the Art of Biblical Commentary, in: Cook, J (ed.) 2002, 642-651.

Chau, K W, Cao, Y, Anson, M & Zhang, J (eds) 2002. Application of Data Warehouse and Decision Support System in Construction Management. *Automation in Construction* 12, 213-224.

Cheung, D W, Zhou, B, Kao, B, Kan, H & Lee, S D (eds) 2001. Towards the Building of a Dense-Region-Based OLAP System. *Data & Knowledge Engineering* 36, 1-27. Available: http://www.ComputerScienceWeb.com (Science Direct).

Chiaramella, Y 1986. Computer Science and Text, in: *Bible et informatique, le texte: actes du premier colloque international, Louvain-la-Neuve (Belgique) 2-3-4 septembre 1985 = Bible and Computer, the Text: Proceedings of the First International Colloquium/Association internationale Bible et informatique, en collaboration avec la Faculté de théologie et le CETEDOC de l'Université catholique de Louvain-la-Neuve.*: Paris: Champion, 119-139.

Chun, S J, Chung, C W & Lee, S L (eds) 2004. Space-Efficient Cubes for OLAP Range-Sum Queries. *Decision Support Systems* 37, 83-102. Available: http://www.ComputerScienceWeb.com (Science Direct).

Cook, J (ed.) 2002. *Bible and Computer: The Stellenbosch AIBI-6 Conference, Proceedings of the Association Internationale Bible et Informatique "From Alpha to Byte", University of Stellenbosch 17-21 July, 2000.* Leiden: Brill.

Derose, S J 1997. *The SGML FAQ Book: Understanding the Foundation of HTML and XML.* Boston, MA: Kluwer Academic Press.

De Troyer, K 2002. 4Q550 in the Context of the Darius Traditions: the Need for Integration of Different Tools, in: Cook, J (ed.) 2002, 573-581.

Dik, S C 1992. *Functional Grammar in Prolog: an Integrated Implementation for English, French, and Dutch* (Natural Language Processing; 2). Berlin: De Gruyter.

Glasgow, J & Malton, A s.a. A Semantics for Model-Based Spatial Reasoning. Department of Computing and Information Science, Queen's University, Kingston, Ontario. (Unpublished document). Available:

http://www.cs.queensu.ca/TechReports/Reports/1994-360.pdf (Accessed 1 September 2004).

Holzner, S 2004. *SAMS Teach Yourself XML in 21 Days,* 3rd Edition. Indianapolis, Indiana: SAMS.

Hughes, J J 1987. *Bits, Bytes & Biblical Studies: a Resource Guide for the Use of Computers in Biblical and Classical Studies.* Grand Rapids, Michigan: Academic Books.

Karyannidis, N & Sellis, T 2003. SISYPHUS: the Implementation of a Chunk-Based Storage Manager for OLAP Data Cubes. *Data & Knowledge Engineering* 45, 155-180. Available: http://www.ComputerScienceWeb.com (Science Direct).

Kay, R 2004. Data cubes. *Computer World* 38 (13), 32. Available: http://0-web25.epnet.com.innopac.up.ac.za (Accessed 22 June 2004).

Koutsoukis, N S, Mitra, G & Lucas, C 1999. Adapting on-Line Analytical Processing for Decision Modelling: the Interaction of Information and Decision Technologies. *Decision Support Systems* 26, 1-30.

Kroeze, J H 2002. Developing a Multi-Level Analysis of Jonah Using html, in: Cook, J (ed.) 2002, 653-662.

Lee, Y K, Whang, K Y, Moon, Y S & Song, I Y 2003. An Aggregation Algorithm Using a Multidimensional File in Multidimensional OLAP. *Information Sciences* 152, 121-138. Available: http://www.ComputerScienceWeb.com (Science Direct).

Link, G 1995. Algebraic Semantics for Natural Language: Some Philosophy, Some Applications. *Int. J. Human-Computer Studies* 43, 765-784.

Marchand, P, Brisebois, A, Bédard, Y & Edwards, G 2004. Implementation and Evaluation of a Hypercube-Based Method for Spatiotemporal Exploration and Analysis. *ISPRS Journal of Photogrammetry & Remote Sensing.* (Article in press.)

Nieuwoudt, B A 1989. Computer Assisted Research of the Greek and Hebrew Bible (II), in: Talstra (ed.) 1989, 101-118.

Pietersma, A 2002. The Interlinear Model and the Septuagint, in: Cook, J (ed.) 2002, 337-364.

Sowa, J F 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Pacific Grove, CA: Brooks/Cole.

Talstra, E (ed.) 1989. *Computer Assisted Analysis of Biblical Texts: Papers Read at the Workshop on Occasion of the Tenth Anniversary of the "Werkgroep*

Informatica", Faculty of Theology, Vrije Universiteit, Amsterdam, November, 5-6. Amsterdam: Free University Press.

Talstra, E 1989. Introduction: Opening Address and Report, in: Talstra (ed.) 1989, 1-8.

Talstra, E 2002. Computer-Assisted Linguistic Analysis of the Hebrew Database Used in Quest.2, in: Cook, J (ed.) 2002, 1-22.

Talstra, E & Postma, F 1989. On Text and Tools: a Short History of the Werkgroep Informatica (1977-1987), in: Talstra, E (ed.) 1989, 9-27.

Tov, E 1989. Computer Assisted Research of the Greek and Hebrew Bible (I), in: Talstra, E (ed.) 1989, 87-99.

Van der Merwe, C H J 2002. The Bible and Hypertext Technology: Challenges for Maximizing the Use of a New Type of Technology in Biblical Studies. *Journal of Northwest Semitic Languages* 28, 87-102.

ADDENDUM A: Linguistic Data Regarding Gen 1:1-5 Represented by a Three-Dimensional Array in Visual Basic 6

```
Option Explicit

Public Clause(1 To 14, 1 To 4, 1 To 6) As String

Sub Main()

Clause(1, 1, 1) = "Gen01v01a"
Clause(1, 1, 2) = "bre$it"
Clause(1, 1, 3) = "in the beginning"
Clause(1, 1, 4) = "PP"
Clause(1, 1, 5) = "Adjunct"
Clause(1, 1, 6) = "Time"
Clause(1, 2, 1) = "-"
Clause(1, 2, 2) = "bara"
Clause(1, 2, 3) = "he created"
Clause(1, 2, 4) = "VP"
Clause(1, 2, 5) = "Main verb"
Clause(1, 2, 6) = "Action"
Clause(1, 3, 1) = "-"
Clause(1, 3, 2) = "elohim"
Clause(1, 3, 3) = "God"
Clause(1, 3, 4) = "NP"
Clause(1, 3, 5) = "Subject"
Clause(1, 3, 6) = "Agent"
Clause(1, 4, 1) = "-"
Clause(1, 4, 2) = "et ha$amayim ve'et ha'arets"
Clause(1, 4, 3) = "the heaven and the earth"
Clause(1, 4, 4) = "NP"
Clause(1, 4, 5) = "Object"
Clause(1, 4, 6) = "Patient"

Clause(2, 1, 1) = "Gen01v02a"
Clause(2, 1, 2) = "veha'arets"
Clause(2, 1, 3) = "and the earth"
Clause(2, 1, 4) = "NP"
Clause(2, 1, 5) = "Subject"
Clause(2, 1, 6) = "Zero"
```

Clause(2, 2, 1) = "-"
Clause(2, 2, 2) = "hayta"
Clause(2, 2, 3) = "was"
Clause(2, 2, 4) = "VP"
Clause(2, 2, 5) = "Copulative verb"
Clause(2, 2, 6) = "State"
Clause(2, 3, 1) = "-"
Clause(2, 3, 2) = "tohu vavohu"
Clause(2, 3, 3) = "an emptiness and void"
Clause(2, 3, 4) = "NP"
Clause(2, 3, 5) = "Copula-predicate"
Clause(2, 3, 6) = "Classification"

Clause(3, 1, 1) = "Gen01v02b"
Clause(3, 1, 2) = "wexo$ex"
Clause(3, 1, 3) = "and darkness"
Clause(3, 1, 4) = "NP"
Clause(3, 1, 5) = "Subject"
Clause(3, 1, 6) = "Zero"
Clause(3, 2, 1) = "-"
Clause(3, 2, 2) = "al pney tehom"
Clause(3, 2, 3) = "on the surface of primeval ocean"
Clause(3, 2, 4) = "PP"
Clause(3, 2, 5) = "Copula-predicate"
Clause(3, 2, 6) = "Location"

Clause(4, 1, 1) = "Gen01v02c"
Clause(4, 1, 2) = "veruach elohim"
Clause(4, 1, 3) = "and the spirit of God"
Clause(4, 1, 4) = "NP"
Clause(4, 1, 5) = "Subject"
Clause(4, 1, 6) = "Positioner"
Clause(4, 2, 1) = "-"
Clause(4, 2, 2) = "meraxefet"
Clause(4, 2, 3) = "hovering"
Clause(4, 2, 4) = "AP"
Clause(4, 2, 5) = "Copula-predicate"
Clause(4, 2, 6) = "Position"
Clause(4, 3, 1) = "-"
Clause(4, 3, 2) = "al pney hamayim"

Clause(4, 3, 3) = "on the surface of the water"
Clause(4, 3, 4) = "PP"
Clause(4, 3, 5) = "Complement"
Clause(4, 3, 6) = "Location"

Clause(5, 1, 1) = "Gen01v03a"
Clause(5, 1, 2) = "vayomer"
Clause(5, 1, 3) = "And He said"
Clause(5, 1, 4) = "VP"
Clause(5, 1, 5) = "Main verb"
Clause(5, 1, 6) = "Action"
Clause(5, 2, 1) = "-"
Clause(5, 2, 2) = "elohim"
Clause(5, 2, 3) = "God"
Clause(5, 2, 4) = "NP"
Clause(5, 2, 5) = "Subject"
Clause(5, 2, 6) = "Agent"
Clause(5, 3, 1) = "-"
Clause(5, 3, 2) = "[yehi or]"
Clause(5, 3, 3) = "[Let there be light]"
Clause(5, 3, 4) = "[EC]"
Clause(5, 3, 5) = "[Object clause]"
Clause(5, 3, 6) = "[Patient]"

Clause(6, 1, 1) = "Gen01v03b"
Clause(6, 1, 2) = "yehi"
Clause(6, 1, 3) = "Let there be"
Clause(6, 1, 4) = "VP"
Clause(6, 1, 5) = "Copulative verb"
Clause(6, 1, 6) = "State"
Clause(6, 2, 1) = "-"
Clause(6, 2, 2) = "or"
Clause(6, 2, 3) = "light"
Clause(6, 2, 4) = "NP"
Clause(6, 2, 5) = "Subject"
Clause(6, 2, 6) = "Zero"

Clause(7, 1, 1) = "Gen01v03c"
Clause(7, 1, 2) = "vayehi"
Clause(7, 1, 3) = "And there was"

Clause(7, 1, 4) = "VP"
Clause(7, 1, 5) = "Copulative verb"
Clause(7, 1, 6) = "State"
Clause(7, 2, 1) = "-"
Clause(7, 2, 2) = "or"
Clause(7, 2, 3) = "light"
Clause(7, 2, 4) = "NP"
Clause(7, 2, 5) = "Subject"
Clause(7, 2, 6) = "Zero"

Clause(8, 1, 1) = "Gen01v04a"
Clause(8, 1, 2) = "vayar"
Clause(8, 1, 3) = "And he saw"
Clause(8, 1, 4) = "VP"
Clause(8, 1, 5) = "Main verb"
Clause(8, 1, 6) = "Process"
Clause(8, 2, 1) = "-"
Clause(8, 2, 2) = "elohim"
Clause(8, 2, 3) = "God"
Clause(8, 2, 4) = "NP"
Clause(8, 2, 5) = "Subject"
Clause(8, 2, 6) = "Processed"
Clause(8, 3, 1) = "Patient"
Clause(8, 3, 2) = "[et ha'or ki tov]"
Clause(8, 3, 3) = "[that the light was good]"
Clause(8, 3, 4) = "[EC]"
Clause(8, 3, 5) = "[Object clause]"
Clause(8, 3, 6) = "Patient"

Clause(9, 1, 1) = "Gen01v04b"
Clause(9, 1, 2) = "et ha'or"
Clause(9, 1, 3) = "the light"
Clause(9, 1, 4) = "NP"
Clause(9, 1, 5) = "Subject"
Clause(9, 1, 6) = "Zero"
Clause(9, 2, 1) = "-"
Clause(9, 2, 2) = "ki"
Clause(9, 2, 3) = "that"
Clause(9, 2, 4) = "ConjP"
Clause(9, 2, 5) = "Conj"

Clause(9, 2, 6) = "-"
Clause(9, 3, 1) = "-"
Clause(9, 3, 2) = "tov"
Clause(9, 3, 3) = "good"
Clause(9, 3, 4) = "AP"
Clause(9, 3, 5) = "Copula-predicate"
Clause(9, 3, 6) = "Quality"

Clause(10, 1, 1) = "Gen01v04c"
Clause(10, 1, 2) = "vayavdel"
Clause(10, 1, 3) = "and he separated"
Clause(10, 1, 4) = "VP"
Clause(10, 1, 5) = "Main verb"
Clause(10, 1, 6) = "Action"
Clause(10, 2, 1) = "-"
Clause(10, 2, 2) = "elohim"
Clause(10, 2, 3) = "God"
Clause(10, 2, 4) = "NP"
Clause(10, 2, 5) = "Subject"
Clause(10, 2, 6) = "Agent"
Clause(10, 3, 1) = "-"
Clause(10, 3, 2) = "ben ha'or"
Clause(10, 3, 3) = "between the light"
Clause(10, 3, 4) = "PP"
Clause(10, 3, 5) = "Complement"
Clause(10, 3, 6) = "Patient"
Clause(10, 4, 1) = "-"
Clause(10, 4, 2) = "uven haxo$ex"
Clause(10, 4, 3) = "and between the darkness"
Clause(10, 4, 4) = "PP"
Clause(10, 4, 5) = "Complement"
Clause(10, 4, 6) = "Source"

Clause(11, 1, 1) = "Gen01v05a"
Clause(11, 1, 2) = "vayiqra"
Clause(11, 1, 3) = "and he called"
Clause(11, 1, 4) = "VP"
Clause(11, 1, 5) = "Main verb"
Clause(11, 1, 6) = "Action"
Clause(11, 2, 1) = "-"

Clause(11, 2, 2) = "elohim"
Clause(11, 2, 3) = "God"
Clause(11, 2, 4) = "NP"
Clause(11, 2, 5) = "Subject"
Clause(11, 2, 6) = "Agent"
Clause(11, 3, 1) = "-"
Clause(11, 3, 2) = "la'or"
Clause(11, 3, 3) = "to the light"
Clause(11, 3, 4) = "PP"
Clause(11, 3, 5) = "IndObj"
Clause(11, 3, 6) = "Patient"
Clause(11, 4, 1) = "-"
Clause(11, 4, 2) = "yom"
Clause(11, 4, 3) = "day"
Clause(11, 4, 4) = "NP"
Clause(11, 4, 5) = "Complement"
Clause(11, 4, 6) = "Product"

Clause(12, 1, 1) = "Gen01v05b"
Clause(12, 1, 2) = "velaxo$ex"
Clause(12, 1, 3) = "and to the darkness"
Clause(12, 1, 4) = "PP"
Clause(12, 1, 5) = "IndObj"
Clause(12, 1, 6) = "Patient"
Clause(12, 2, 1) = "-"
Clause(12, 2, 2) = "qara"
Clause(12, 2, 3) = "he called"
Clause(12, 2, 4) = "VP"
Clause(12, 2, 5) = "Main verb"
Clause(12, 2, 6) = "Action"
Clause(12, 3, 1) = "-"
Clause(12, 3, 2) = "layla"
Clause(12, 3, 3) = "night"
Clause(12, 3, 4) = "NP"
Clause(12, 3, 5) = "Complement"
Clause(12, 3, 6) = "Product"

Clause(13, 1, 1) = "Gen01v05c"
Clause(13, 1, 2) = "vayehi"
Clause(13, 1, 3) = "and it was"

```
Clause(13, 1, 4) = "VP"
Clause(13, 1, 5) = "Copulative verb"
Clause(13, 1, 6) = "State"
Clause(13, 2, 1) = "-"
Clause(13, 2, 2) = "erev"
Clause(13, 2, 3) = "evening"
Clause(13, 2, 4) = "NP"
Clause(13, 2, 5) = "Subject"
Clause(13, 2, 6) = "Zero"

Clause(14, 1, 1) = "Gen01v05d"
Clause(14, 1, 2) = "vayehi"
Clause(14, 1, 3) = "and it was"
Clause(14, 1, 4) = "VP"
Clause(14, 1, 5) = "Copulative verb"
Clause(14, 1, 6) = "State"
Clause(14, 2, 1) = "-"
Clause(14, 2, 2) = "voker"
Clause(14, 2, 3) = "morning"
Clause(14, 2, 4) = "NP"
Clause(14, 2, 5) = "Subject"
Clause(14, 2, 6) = "Zero"
Clause(14, 3, 1) = "-"
Clause(14, 3, 2) = "yom exad"
Clause(14, 3, 3) = "day one"
Clause(14, 3, 4) = "NP"
Clause(14, 3, 5) = "Attribute"
Clause(14, 3, 6) = "Quality"

End Sub
```