bible society making the bible heard

British & Foreign Bible Society

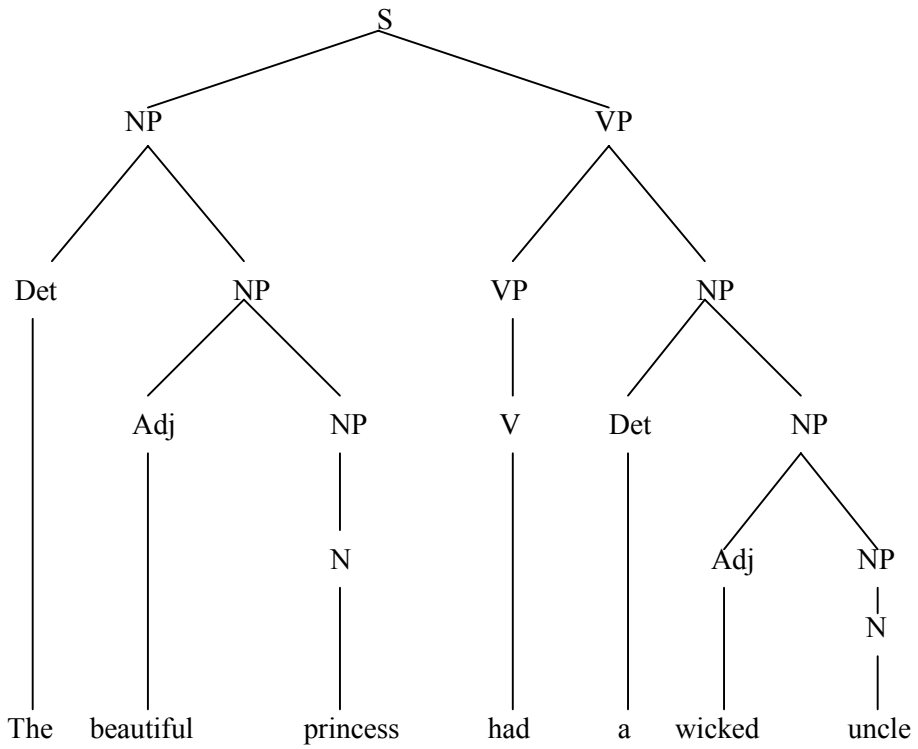
Masoretic Hebrew Cantillation and Constituent Structure Analysis

This paper is written as background information to accompany *The Masoretes and the Punctuation of Biblical Hebrew*. In that paper the authors have described the Masoretic cantillation in terms of constituent structure trees. Here we provide a simple explanation of such trees and attempt to demonstrate the mathematical validity of classifying the cantillation system as a form of constituent structure analysis. Finally, rules are constructed from that analysis to control valid subdivision of verses in the Masoretic text.

1. Constituent structure trees

Constituent Structure trees have been used by linguists since Bloomfield[1] to provide a clear and graphic representation of the grammatical components of a sentence. Although they were originally intended to be a purely visual device, subsequent work on the mathematical structure of trees (notably by Harris[2] and Chomsky[3]) has made them highly amenable to computer analysis in a wide range of disciplines.

One of the most frequent uses of Constituent Structure Analysis in linguistics is in descriptions of Phrase-Structure Grammars. Here is a typical analysis of a simple English sentence as the product of a constituent structure tree:



[1] Leonard Bloomfield, *Language*. New York 1933, London 1935.
[2] Zellig S. Harris, *Methods in Structural Linguistics*, Chicago 1951.
[3] Noam Chomsky, *Syntactic Structures*, The Hague 1957.

The tree consists of five parts:

1. The root. By convention CS trees are represented upside-down, so the root is at the top labelled S(entence).

2. A set of Nodes. The root itself is the primary node which divides the structure into two logical halves. Each node subdivides the data underneath it into a further two parts. The node is then called the *mother* of the pair underneath it, which are therefore its *daughters*. Two daughter nodes with the same mother node are referred to as *sister* nodes.

3. A set of Labels which are written at the nodes. Which labels we use will entirely depend on the nature of the analysis. In this case it is grammatical, and the labels are N(oun) P(hrase), V(erb) P(hrase), Det(erminer), Adj(ective), N(oun) and V(erb).

4. A set of 'Leaves' or terminals which is the output from the analysis (here labelled Det, Adj, N ...).

5. A set of 'formatives'. In a Phrase Structure tree these are the actual words of the sentence under analysis placed under their appropriate leaves: *The*, *beautiful*, *princess* etc.

To construct such a tree we need two types of rule:

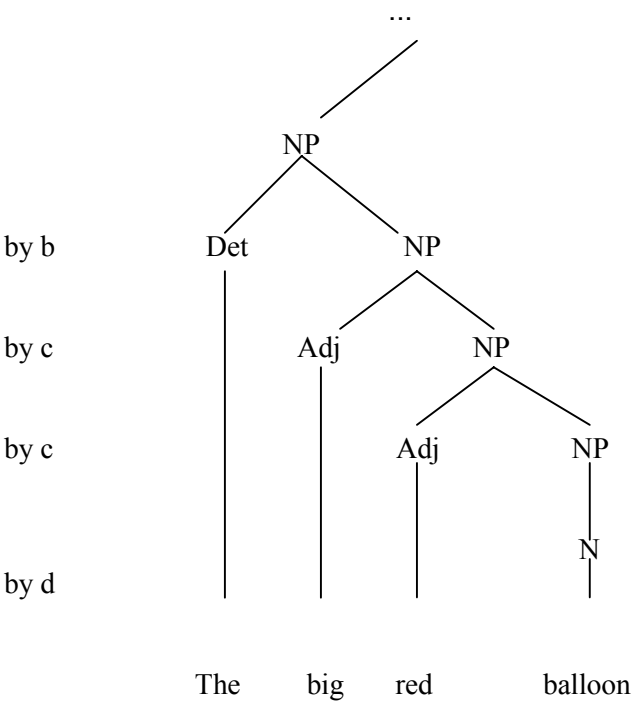
1. 'Rewrite' rules. A typical rewrite rule has the form S→NP VP. This states that a node labelled S may be divided into two nodes, the first of which is to be labelled NP and the second VP. The rewrite rules for the above tree are:

a. S→NP VP	(Sentence may divide to Noun Phrase + Verb Phrase)
b. NP→Det NP	(Noun Phrase may divide to determiner + another Noun Phrase)
c. NP→Adj NP	(Noun Phrase may divide to leaf adjective + another Noun Phrase)
d. NP→N	(Noun Phrase may not divide but just be a leaf Noun)
e. VP→VP NP	(Verb Phrase may divide to Verb Phrase + Noun Phrase)
f. VP→V	(Verb Phrase may not divide but just be a leaf Verb)

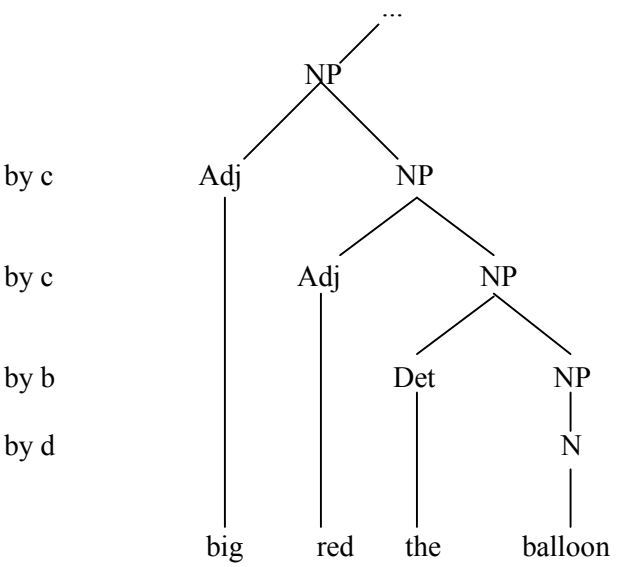
2. 'Instantiation' rules. These turn *leaves* into *words* and have the theoretical form:
V → *come, go, do, make, take, buy, sell*...

The main difference to note is that while rewrite rules draw on a finite (usually quite small) set of terms, instantiation rules can draw on an unbounded set – in this case the set of all possible verbs (in all possible tenses) in the language being used.

In addition to these rules we normally need a set of ‘constraints’ on the rules. A constraint is simply a device which specifies that a given rule may not be applied under certain circumstances. From the above rewrite rules we could derive



which is prefectly acceptable, but we could also derive



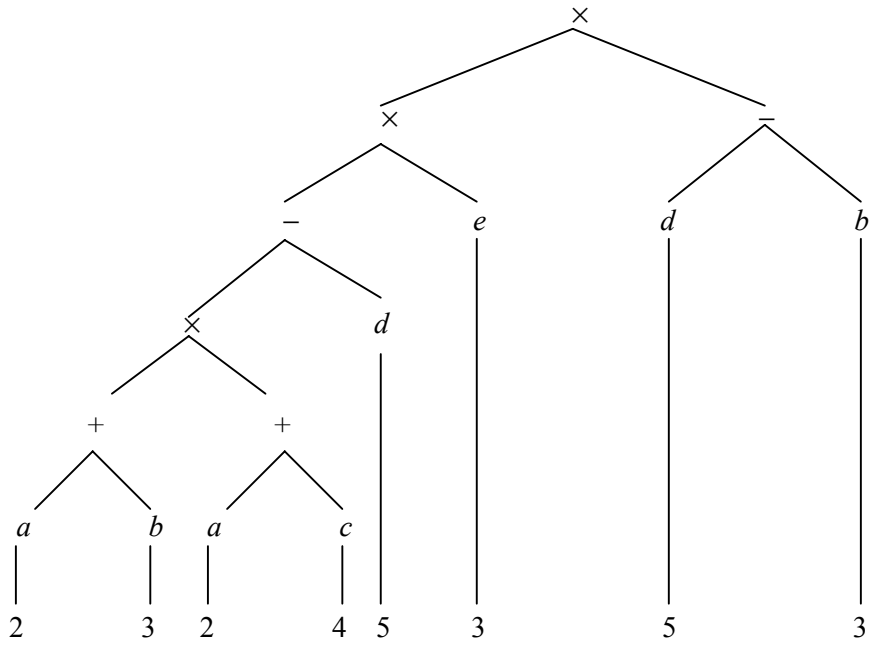
which is not.

An unacceptable output is conventionally signalled by an asterisk, so writing "*big red the balloon" means that this is not a acceptable noun phrase in English.
A typical constraint on the rewrite rules given might read:
*b/c— which is to be read as
'You cannot use rule b' (*b)
'in the context of' (/)
'following rule c' (c—).

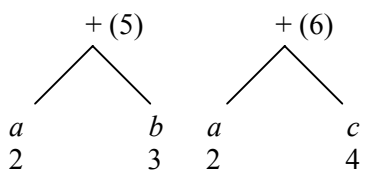
Constituent Structure trees can be used for many purposes other than illustrating Phrase Structure Grammars. Consider the following equation:

$$z = ((((a + b) \times (a + c)) - d) \times e) \times (d - b)$$

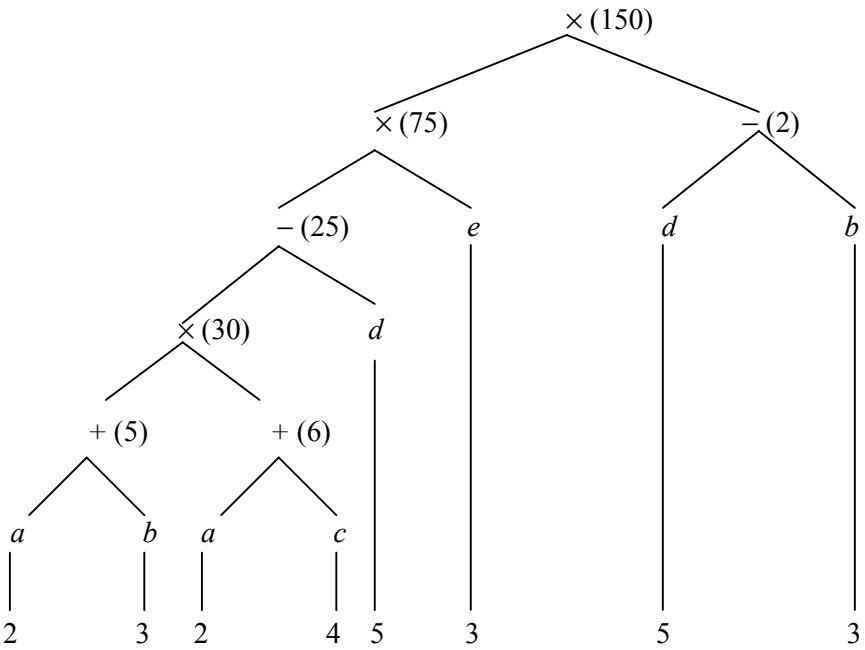
This may be expressed as a CS tree with the (finite) set of arithmetic operators as node labels, the (finite) set of variables as leaves and the (non-finite) set of numbers as instantiations. Let as assume the instantiation rules:
$a = 2$, $b = 3$, $c = 4$, $d = 5$, $e = 3$



To 'solve' the equation we simply work from the bottom of the tree to the top performing the node operation (+, −, ×, ÷) on each pair of sisters in turn. Here we write the result as an extension to the node label of their mother so:



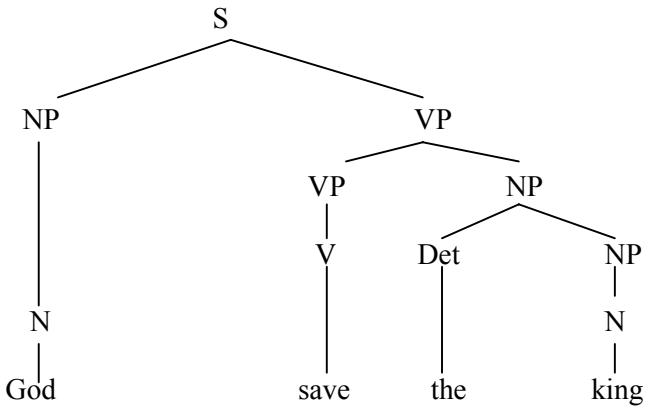
Completing the tree in this way we have:



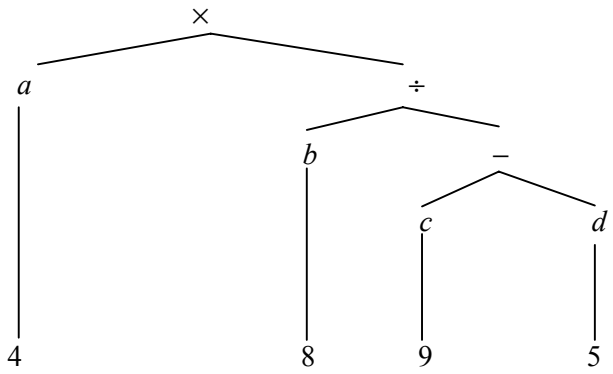
Another CS tree could be produced to show the stages in cooking a stew, where the labels would be the finite set of possible culinary operations and the formatives would be the non-finite set of all possible ingredients and quantities.

It should be stressed that Labels are not in themselves Nodes. The following trees describe quite different processes:

a.

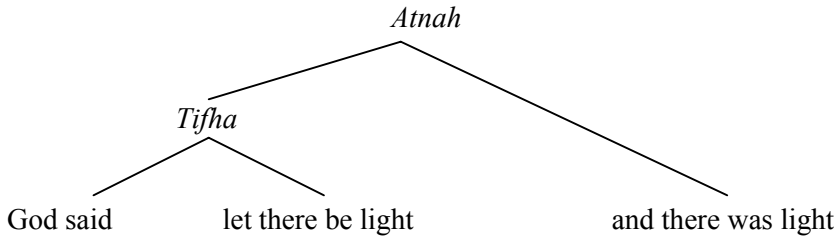


b.

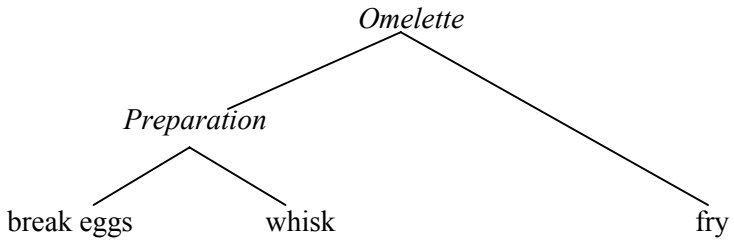


Tree *a* describes the phrase structure grammar of an English sentence and tree *b* the equation *z* = *a* (*b* ÷ (*c* – *d*)) but the trees have precisely the same structure. They are in fact *the same tree*, but with different labels and formatives. This is perhaps the major advantage that tree structures have for the mathematician: carefully used they enable us to find structural connections between processes in quite different disciplines and to subject them to the same mathematical analysis: parsing a sentence in Greek or Hebrew may be structurally identical to solving a problem in pure mathematics or physics or to designing a computer program.

It so happens that the following tree describes the structure of a Hebrew verse (Gen 1.3)[4]:



but the same structure could describe completely different processes such as:



It is this generality of application which makes the CS tree an invaluable tool for describing the 'atomic' structure of verses in the Hebrew Scriptures.

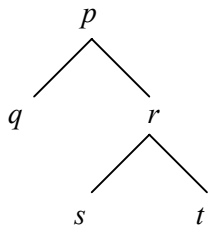
[4] On Atnah see *The Masoretes and the Punctuation of Biblical Hebrew* p.12, on Tifha ibid. pp. 15f.

2. Relationships in tree structures

Harris's mathematical treatment of trees enables us to demonstrate the validity of such a description. To do so, we must first explain three fundamental relationships between nodes in any valid CS tree.

2a. *Dominance*

A node *p* is said to *dominate* another node *q* if there is a connected sequence of *descending* branches from *p* to *q*.



In this diagram *p* dominates *q*, *r*, *s* and *t*. Also *p* is said to *immediately dominate* its daughters *q* and *r* because the path from p to either of these does not pass through any intermediate nodes. Similarly *r* immediately dominates *s* and *t*. However, *q* does not dominate *r*, *s* or *t* because the path from *q* to any of these involves ascending to *p* as well as descending. The only node of a tree which is not dominated by another is the *root* – in this case *p*.

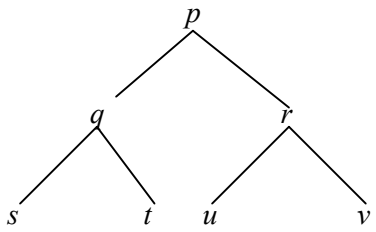
Two nodes *q* and *r* are said to be *clause mates* if
1) neither dominates the other and
2) both are dominated by the same node.

In the above tree, *q* and *r* are clause mates dominated by *p*. *s* and *t* are clause mates dominated by *r. q*, *s* and *t* are clause mates dominated by *p*.

2b. *Precedence*

The term *precedence* is used in a precise technical sense which must not be confused with its non-mathematical usage.

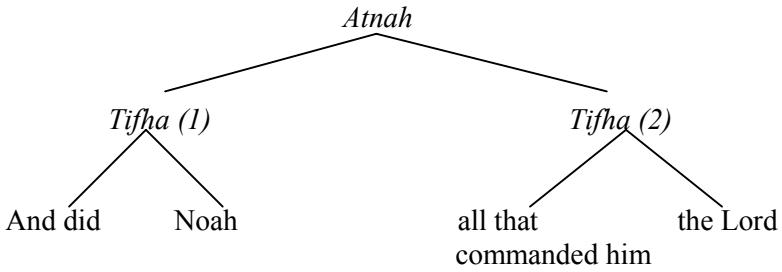
A node *q* is said to *precede* another node *r* if
1) *q* comes before (to the left of) *r*
2) *q* neither dominates nor is dominated by *r*.



In this tree, *q* precedes *r* because
1) *q* comes before *r* and
2) there is no entirely downward path either from *q* to *r* or from *r* to *q*.

For the same reasons, *s* precedes *t, u* and *v*. But although *q* comes to the left of *p* it does not technically *precede* it because *p* dominates *q*. By the same token, *s* does not precede either *q* or *p* because a purely downward path can be traced to *s* from either. On the other hand, *q*, *s* and *t* all precede *r* because
1) they come before it and
2) a path between any of them and *r* must go up as well as down.

Some care must be exercised in considering precedence relations in cantillation. Consider the following tree (Gen 7.5):



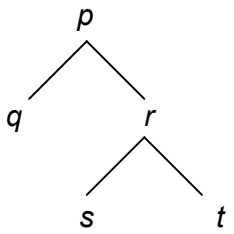
Viewed purely as punctuation marks in the text, the first Tifha comes before the Atnah, which in turn comes before the second Tifha. When they are viewed as nodes this is no longer true. Each node 'rules' and represents a domain. The domain of the first Tifha is *And did Noah*; the domain of the second is *all that commanded him the Lord*. So the first Tifha precedes the second – the path from one to the other needs to ascend to Atnah. But the domain of Atnah is the whole sentence. No *part* of the sentence can precede the *whole* sentence: the first Tifha is dominated by Atnah and does not precede it *as a node*.

2c. *Command*

In some discussions of cantillation the word 'command' is used to describe the hierarchic relationship between disjunctive markers: Atnah is said to 'command'

Zaqef etc[5]. Our use of the word here is entirely different and is set purely in the mathematical context of constituent structure trees.

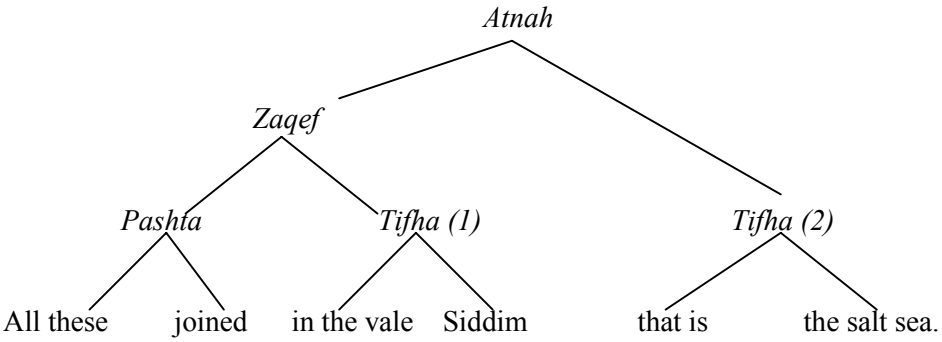
A node *x* commands a node *y* if
1) neither dominates the other and
2) *x* is *immediately dominated* by another node *z* which also dominates (but not necessarily immediately) *y*. In the following sub-tree



q commands r, *s* and *t* because *q* is *immediately dominated* by *p* which in turn dominates *r* (immediately), as well as *s* and *t* (not immediately). For the same reason *r* commands *q*.

On the other hand, *s* and *t* do not command q since they are immediately dominated by *r* which does *not* dominate *q*.

In Gen 14.3 we find:



Here Zaqef and Tifha (2) command each other, as do Pashta and Tifha (1). Tifha (2) commands Pashta and Tifha (1) as well as Zaqef but Pashta does not command Tifha (2).

[5] e.g. S. Bohlius, *Scrutinium S.S. ex accentibus*, Rostock 1636, classified the accents as *Imperatores, Reges, Duces, Comites*. Further categories were added by later grammarians, both Christian and Jewish. In this system the Imperatores are said to 'command' the Reges which in turn command the Duces. This hierarchic use of the word is quite different from the mathematical usage in this paper.

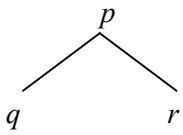
3 Valid Constituent Structure trees

To be mathematically valid, a CS tree must fulfil three conditions.

3a. *The single root condition*
In every well-formed tree there is exactly one root node which dominates all other nodes. In the case of rabbinic cantillation this condition is satisfied by Atnah (or occasionally but equally validly by Tifha) in the punctuation of the 21 books[6].

3b. *The exclusivity condition*
In every well-formed tree any two nodes *x* and *y* stand in a precedence relation if and only if they do not stand in a dominance relation.

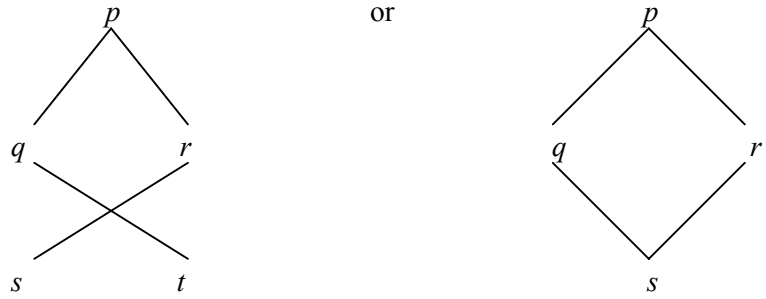
In the tree:



q precedes *r* because there is no dominance relation between them. But *p* neither precedes nor follows *q* because it dominates *q*. We have already shown (2b above) that this condition holds for the cantillation system.

3c. *The nontangling condition*
In every well-formed tree if two nodes *x* and *y* stand in a precedence relation such that *x* precedes *y* then all nodes dominated by *x* must precede all nodes dominated by *y*.

This condition eliminates structures such as



Since, in the cantillation system, output strings are strictly ordered by precedence and no node can be directly dominated by more than one other node the nontangling condition is inevitably fulfilled[7].

[6] On the expression 'the 21 books' see *The Masoretes and the Punctuation of Biblical Hebrew* 2c p.10
[7] There is an apparent infringement of the nontangling condition in Gen 1.16. We have argued in *The Masoretes and the Punctuation of Biblical Hebrew* p.22 that this is best understood as a Masoretic correction to the received text and does not conflict with the rules here given.

It follows that the Masoretic cantillation system can be expressed as a valid Constituent Structure Analysis of each verse of the Hebrew Bible.

Such an expression has many benefits. Perhaps the two greatest are that it is completely computable and that it enables us to define closely the 'atoms' of a verse and the ways in which these atoms may legitimately be combined to produce longer 'molecules'.

4. **Computability**

Any structure which can be drawn as a valid tree can also be expressed unambiguously in purely mathematical notation[8]. This means that it is always possible to construct a set of functions which will completely and validly analyse a verse into its precise component structure.

5. **The substructure of Hebrew verses**

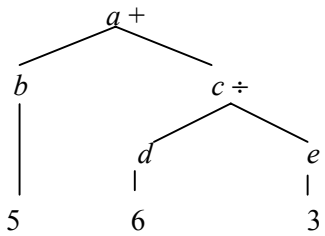
Clearly the smallest unit of a verse is found at a leaf. In the cantillation, a leaf may consist of

a: a single word.
b: two or more words connected by maqqef [9].
c: two or more words (possibly including *b*) connected by conjunctive accents[10].

In the case of *b* and *c* it is illicit (as far as the Masoretes are concerned) to further subdivide the leaf, so in Gen 1.1 *God created*, although two words, forms a single leaf because the words are connected by the conjunctive *munah*: the act of creation cannot be separated, linguistically or conceptually, from the creator himself.

Leaf nodes may be extended to form valid sub-trees by the following rule: *a node x may be extended to a sub-tree of clause mates xy if and only if x commands y and y commands x.*

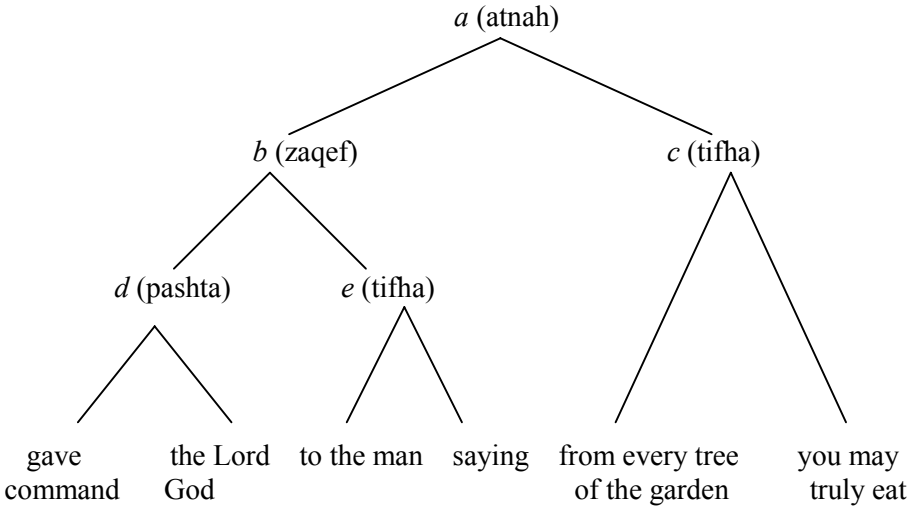
The easiest way of illustrating this rule it by considering once more an 'arithmetic' tree:



[8] The tree component of a *command* relation, for example, can be precisely expressed as $(x, y) \notin \boldsymbol{D} \wedge (y, x) \notin \boldsymbol{D} \wedge (\exists z \in \boldsymbol{N})((x, z) \in \boldsymbol{B} \wedge (z, y) \in \boldsymbol{D})$, and that of *clause mates* as $(x, y) \notin \boldsymbol{D} \wedge (y, x) \notin \boldsymbol{D} \wedge (\exists z \in \boldsymbol{N})((x, z) \in \boldsymbol{B} \wedge (y, z) \in \boldsymbol{B})$. Such formulae may be less transparent to most humans than equivalent tree segments, but (unlike lines drawn on paper) they can be translated directly into computer code.
[9] v. *The Masoretes and the Punctuation of Biblical Hebrew* p.7
[10] v. *The Masoretes and the Punctuation of Biblical Hebrew* 2c2 pp.23f.

Here *b* and *c* command each other, as do *d* and *e*. *b* also commands *d* and *e*, but neither of these commands *b*. So the procedure must be to add *b* to the whole of *c*, calculated by dividing *d* by *e*. If 5 is first added to either 6 or 3 the result will be invalid.

The same logic can be applied to cantillation trees. The tree structure of Gen 2.16 looks like this:

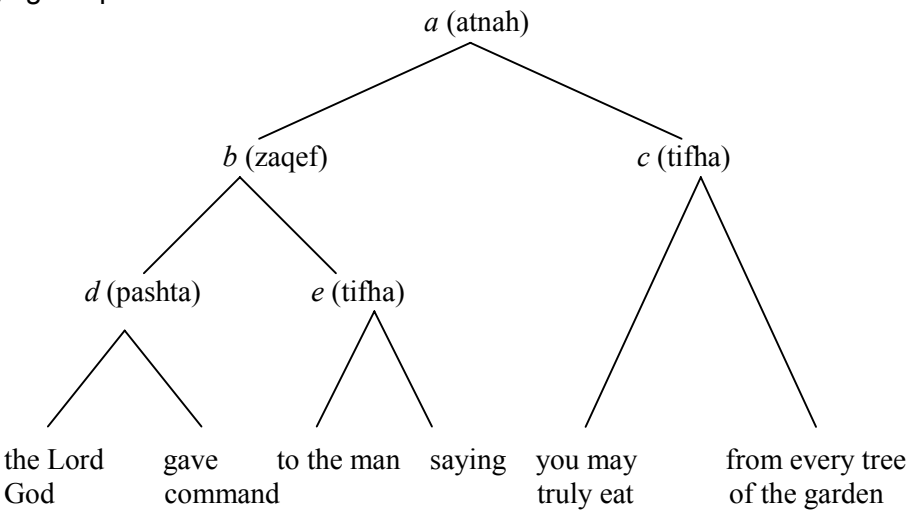


The leaf *saying* commands the leaf *to the man* and vice-versa, so *saying* may be expanded to the phrase *saying to the man*. On the other hand *gave command* and *to the man* do not stand in a command relationship so the extension **gave command to the man* is barred. The phrase *saying to the man* occupies the whole tifha node *e*, which commands the pashta node at *d* and also the leaf *gave command*. *d* also commands *e*, but *gave command* does not, so **gave command to the man saying* is barred. Only the pashta node *d* both commands and is commanded by *e*, so the required phrase expansion must be *the Lord God gave command saying to the man*. This phrase occupies the entire subtree at *b*, which only commands and is commanded by *c*, so the next permitted extension is to the entire verse.

It is not mere pedantry which drove the rabbis to bar the phrase *gave command to the man*. In English this appears to be a perfectly valid substring of the verse, but in Hebrew it would have been ambiguous: a natural interpretation would have been *gave command concerning (what was to happen to) the man*. The same construction is found in Gen 12.20 where *Pharaoh gave command about Abram*. Here the command was not given *to* Abram but to others *about* Abram. The cantillation serves to clarify the meaning of the text and to circumvent a potential ambiguity.

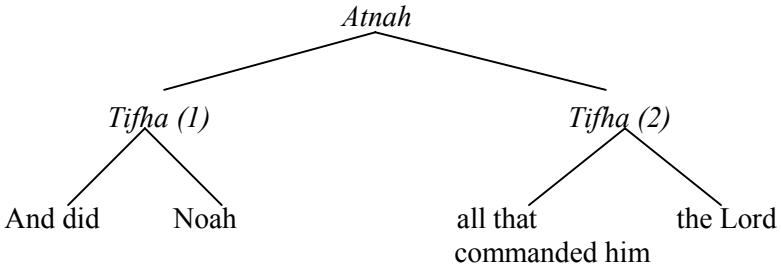
6. **Other uses of the analysis**

In effect, the punctuation imposed on the biblical text by the Masoretes can be used as a 'ready-made' constituent structure for many functions of a Chomsky-type grammar. It is possible to write transformational rules to re-map the output leaves to a different language structure while ensuring that the integrity of the text is maintained. Thus the above tree of Gen 2.16 can be rewritten for English by changing the precendence relations to:

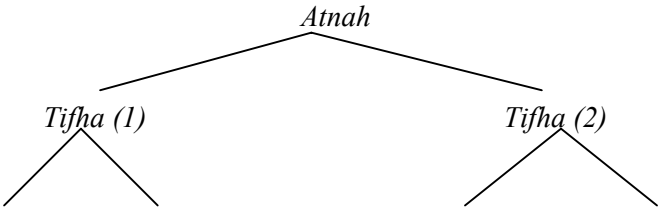


All that is involved in this transform is a precendence rule for English of the form: Subject → Verb → (indirect) Object.

Transforms will frequently involve exchanging components between adjacent nodes:



needs, in English, to become



And Noah	did		all that	commanded him
			the Lord	

but this is no more complex than writing transforms which are already well understood in computational linguistics, such as that between active and passive sentences.

It is true that much work remains to be done if translators are to make full use of the cantillation system, but the essential tools for that work have been in place for a thousand years.

David Robinson BFBS MAT Team
Elisabeth Levy Bible Society in Israel

February 2002

Select Bibliography

The founder of Constituent Analysis was
L Bloomfield
Language 1933

The mathematics described in this paper was developed by
Z Harris
Methods in Structural Linguistics 1951
(Reprinted as *Structural Linguistics* 1961)
String Analysis of Sentence Structure 1962
N Chomsky
Syntactic Structures 1957
Aspects of the Theory of Syntax 1965
The Logical Structure of Linguistic Theory 1975

Excellent introductions may be found in
J Lyons
Introduction to Theoretical Linguistics 1968
R Wall
Introduction to Mathematical Linguistics 1972