Optimal Equivalence in Bible Translation

Presented to the Computer-Assisted Research Section
of the Society of Biblical Literature
by James D. Price
November 17, 2007

Languages are vehicles for expressing and communicating information from one intelligent being to another in the form of meaningful messages.[1] A language itself is not the source of information but a means for encoding information; the information originates from an intelligent user of the language. The user employs the vocabulary and grammar of the language to encode information into a meaningful message from which an intelligent recipient may recover the information by means of a decoding process that operates in his mind. When the message needs to be translated into a second language, the information of the message must be transformed into the equivalent information needed for a user of the second language to generate the equivalent message using its vocabulary and grammar.

For a person who knows both languages well, the translation process takes place in his mind and he is able to produce a good translation rather effortlessly. This process is satisfactory for most bilingual communication. However, when the message is significantly important, such as the Word of God, the translation process must be executed with care.

This paper presents a theory of language translation known as optimal equivalence together with an explanation of experimental software implementation of the theory as applied to Biblical Hebrew and English. The theory is based on a text-linguistic model of language. It is similar to formal equivalence theory[2] in that it maintains optimal equivalence of form, and is similar to functional or dynamic equivalence theory in that it maintains equivalence at the kernel clause level. It differs from both theories in that it also maintains equivalence at the transformational level; thus the term *optimal*. It is similar to relevance theory in that it retains the equivalency of inferences and provides for the equivalency of type (2) context and for commentary of type (3). While this presentation attempts to be general in scope, covering translation between

[1] It is recognized that some artificial languages serve as a means of communication between mechanisms, but the present discussion is limited to natural languages.

[2] Formal equivalence theory maintains the equivalence of words and of the grammatical structures of phrases, clauses, and sentences. For example, it matches the grammatical structure of every different kind of Hebrew phrase with the corresponding equivalent grammatical structure in the target language. The New American Standard Bible and the King James Version are essentially formal equivalence translations.

any two languages, it focuses on Biblical Hebrew as the model for the source language and on English as the model for the target language.

The optimal equivalence theory of translation employs an analysis text-linguistic grammar[3] of the source language, a generative text-linguistic grammar of the target language, and an equivalency map that links the two together. The analysis grammar analyzes the source text, identifying (1) the meaning of each word, (2) the meaning of each kernel clause, (3) the meaning imbedded in each phrase,[4] (4) the information extracted by each back-transformation[5] including the order in which the back-transformations occur,[6] (5) the meaning of idioms,[7] and (6) the discourse information encoded in the logical sequences of clauses and clusters of clauses.[8] The equivalency map provides the corresponding equivalencies of words, phrases, kernel clauses, transformations, and the logical sequences of clauses in the target language. Using this equivalency information, the generative text-linguistic grammar of the target language generates an equivalent target text. The result is an optimally equivalent translation,[9] being both literal and literary. Figure 1 is a simplified illustration of the theory.

[3] A text-linguistic grammar is a formal set of rules defining the lawful structures of phrases, clauses, sentences, and discourses of a language. A generative grammar governs the creation of discourse and an analytical grammar directs the analysis of discourse.

[4] Phrases are understood to be derived from transformations on a part of speech with one or more dependent clauses.

[5] In the generative process, transformations encode information into the structures they alter. Back-transformations extract that information. Translations suffer degradation of information to the extent that they overlook equivalency of transformational information.

[6] In the generative process, the sequential order in which transformations occur affects meaning and encodes information. The order in which back-transformations occur must also be provided to the generative grammar of the target language in order to maintain equivalency.

[7] Idioms cannot be translated literally, but must be transformed into an equivalent idiom in the target language. This includes figurative language.

[8] The analysis of the sequences of clauses involves discourse analysis. This addresses the logical flow of thought and literary form.

[9] Equivalence is limited by how accurately the analysis grammar decodes the information in the source text, how accurately the generative grammar encodes the information into the target text, and how accurately the equivalency map defines code equivalencies. Ideally, perfect components of the theory would produce exact equivalence between source text and target text; but grammars fail to perfectly model their associated languages, and languages fail to have perfect code equivalencies. The claim of optimal equivalence is justified because all the available information in the source text is transferred to the target text as accurately as possible.

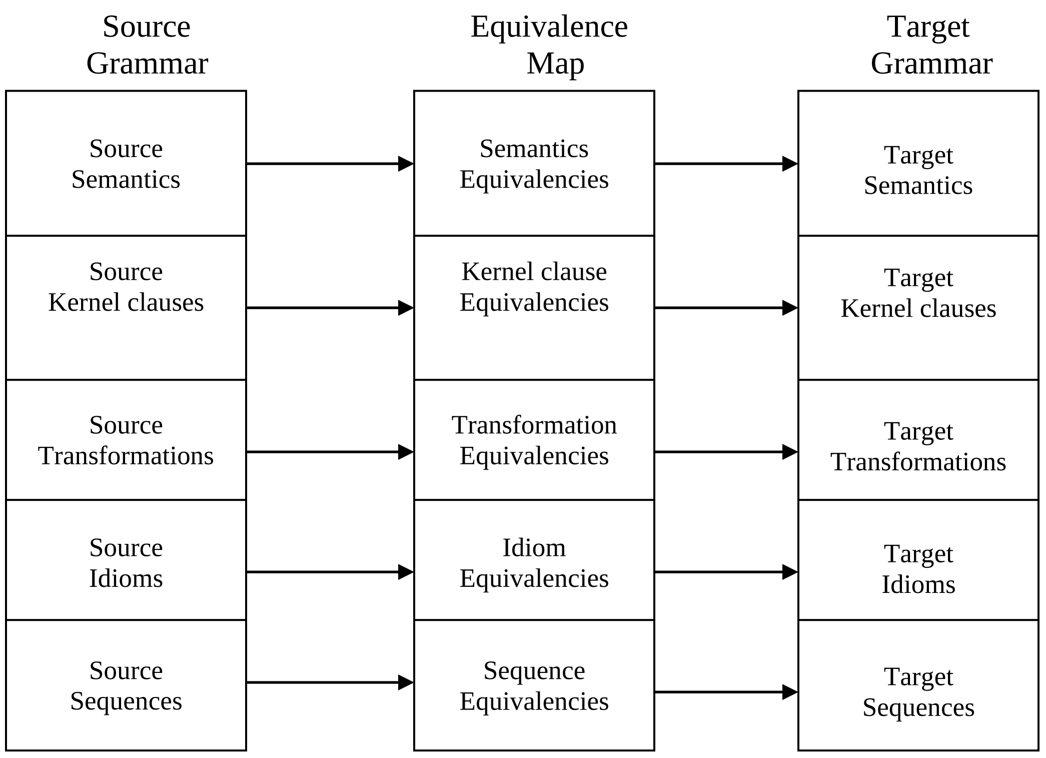


Figure 1
Optimal Equivalence Translation Theory

Selecting a Linguistic Model

Language may be formally represented by various models. For concentrating on the basic structures of simple phrases and clauses, a phrase-structure model is sufficient. For including more complex structures, a transformational model is required. However, because it has been discovered that certain operations necessary to produce meaningful discourse are beyond the capacity of conventional transformational grammars, a discourse model is required for defining the structures of meaningful discourse.

A transformational grammar employs a phrase structure model to define the structures of a basic set of kernel clauses, and it employs transformations to define the ultimate surface structures of clauses in the language it models. Transformational grammars lack the ability to perform deletions, substitutions, coordination, and subordination at the discourse level. A discourse grammar employs a transformational grammar to define the structures of clauses, and it employs discourse transformations to accommodate inter-clause operations.

A structural grammar may be either a generative or analytical grammar of the language it models. A generative grammar functions to generate phrases, clauses, sentences, and discourse in the language it models; an analytical grammar functions to analyze existing discourse in the language it models. A generative grammar aids the teaching and learning of a language; an analytical grammar aids exegesis and exposition. An analytical grammar may be likened to a mirror image of a generative grammar; the two are complementary. The writing of a good generative grammar is the proper starting point for creating a good analytical grammar.

Although some generative grammars employ unordered, context-free rules; such grammars permit the generation of grammatically correct nonsense.[10] The use of grammatical and semantic restraints on the rules, together with the use of restraints on their order, produces meaningful discourse. The implementation of such restraints requires the use of complex constituents[11] that would make the notation unnecessarily cumbersome for this work. Therefore, although complex constituents are used, the notation does not usually indicate the attributes of the constituents, leaving the necessary restraints expressed verbally in the text or in footnotes on the rules to which they apply. It is everywhere assumed, without notation, that the constituents of any rule are semantically compatible.

A Generative Text-Linguistic Model

The generative text-linguistic grammar defined in subsequent chapters attempts to provide a formal model of the following interdependent linguistic systems of a language: orthography, morphology, syntax, and text. It assumes without formal notation that a system of semantic restraints exercises governance over the text grammar. For the purposes of this work, a text-linguistic grammar is defined as a transformational grammar of clauses governed by a transformational grammar of text.

Transformational Grammar of Clauses

A transformational grammar of clauses consists of (1) a set of symbols representing the basic constituents of the language, (2) a set of phrase-structure rules that define the syntax of the kernel clauses of the language, and (3) a set of clause transformations that generate the surface structures of the clauses of the language.

[10] Colorless odors sing tasteless honey.

[11] A complex constituent has one or more attributes, such as number and gender, the values of which are governed and defined by the rules of the grammar.

A kernel clause is a clause whose syntax is in its elementary form with its grammatical values initialized at their default value. That is, a kernel clause has the default word order and is in the declarative mode, the indicative mood, the active voice, the present tense, etc. Phrase-structure rules are replacement rules that generate the elementary phrases and kernel clauses of the language and that have the following basic form:

$$A = B + C$$

where A, B, and C are symbols representing constituents of the language; the sign = means "replace"; and the sign + means "concatenation"—that is, in above rule, B followed by C. Thus the rule is interpreted to read "replace A with B followed by C."

Clause transformations may add or delete constituents, and alter the sequential order of the constituents of a clause. They may also alter the values of the grammatical variables of the constituents. By means of transformations, the clause grammar may generate any of the possible forms a clause may assume. Surface structure is the form of a clause as found in a text. Deep structure consists of the kernel clause(s) and transformations required to generate the surface structure of a clause. The total information contained in a clause consists of the basic information of the kernel clause plus the information added to the clause by the transformations used to create its surface structure. This total information is known as deep-structure information.

Transformational Grammar of Phrases

Phrases consist of a part of speech with an accompanying modifier. In this work, phrases are regarded as the result of one or more transformations on a part of speech with a dependent clause. The modifier is the remnant of the dependent clause varying from a single word to almost the entire clause.

Transformational Grammar of Text

A transformational grammar of text (or discourse) consists of (1) a set of symbols representing the basic text constituents, (2) a set of phrase-structure rules that defines the sequences and interrelationship of the clauses of the text, and (3) a set of text transformations that generate the surface structures of the text of a given discourse in the language. It begins with a set of clauses generated by the associated clause grammar.

(1) The symbols of the text grammar are the same as for the clause grammar, except that the symbol S may now represent a single clause or a defined cluster of clauses.

(2) The phrase-structure rule for the text grammar is as follows:

$$S_a = S_b + C + S_c$$

where the subscripts represent differing index numbers that distinguish the clauses or clause clusters from one another. The conjunction C may be different for each iteration of the rule depending on the type of relationship that exists between the clauses or clusters involved. Although the rule defines only a conjoined pair of clauses or clusters, the rule may be extended to triplets, quadruplets, etc. By means of this one rule, a sequence of interrelated clauses is generated that constitutes a text. The order in which clauses and clusters are joined to form a text depends on literary genre, form, logic, temporal relationships, and coherence of the discourse under construction. This is the stage of discourse construction when the values of tense, aspect, mood, prominence, etc., are determined.

(3) Text transformations operate on the surface structure of the text to add elements of coherence and consistency of reference. In order to accomplish this task, the transformations may (a) add the markers of determination, (b) further delete redundant or unnecessary constituents, (c) replace a constituent with a substitute, or alter the sequence of clauses.

An Analytical Text-Linguistic Model

An analytical text-linguistic grammar functions as a mirror image of its corresponding generative grammar, operating in the reverse order to that of the generative grammar.

An analytical grammar begins with an existing discourse in the language it models, and from the discourse it recovers all the information encoded in it. The analysis grammar models the thinking process of an expositor decoding a discourse and extracting the information from it. It provides the grammatical, syntactical, and back-transformational rules needed to analyze a discourse, beginning with words and phrases, continuing with clauses, and ending with a logical analysis of the flow of thought.

The analysis grammar imposes grammatical and semantic restraints on how its rules are permitted to operate. These restraints consist of grammatical and semantic questions that must be satisfied. According to the analytical model of discourse, a discourse grammar consists of an input of meaningful sequences of words E, a parsing grammar W, a clause grammar F, a phrase grammar U, a sequence grammar D, and a set of output data U.

The parsing grammar W analyzes each word in E deriving its root, stem, class, part of speech, and all its attribute values. The phrase grammar U analyzes each phrase extracting the principal word and identifying its dependent clauses. The clause grammar F analyzes each clause, identifying its dependent clauses, identifying its constituents, clause type, and other attribute values. The sequence grammar D analyzes each clause cluster, identifying the protasis and apodosis, the type and kind of dependency, and other attribute values. All decoded information is recorded in the output data U in a systematic order to provide an exhaustive exegetical analysis of E when the grammar is complete. The analysis grammar may be represented formally as[12]

$$U = \int E \bullet W \bullet F \bullet U \bullet D$$

The system iterates repeatedly over the input data E and the current state of the output U to progressively advance the state of the analysis as each new bit of information is decoded. The process continues until the analysis is complete. The iteration cycle may operate on one clause at a time, but at the end of a discourse, it should iterate through the entire text in order to thoroughly accommodate the operations of the sequence grammar.

The Operating System

The operating system may be viewed as a computer program that manages the operation of the discourse grammar. It has a memory that enables it to keep track of the state of the discourse grammar as it progressively generates the clauses of a given message, and to keep track of the history of the generative process. The current content of the memory is the common knowledge[13] pertinent to a given message at any step in the generative process.

The use of a computer program with a memory should not be regarded as outside the scope of generative grammars. After all, a discourse grammar is supposed to model in some

[12] The symbol $\int$ signifies iteration, and the symbol $\bullet$ signifies mutual exchange of information.

[13] The memory contains the “old information” of the message up to the clause currently being generated. The author (user) of the system may also refer to information contained in the common knowledge he assumes his potential readers already know. This kind of “old information” is contained in the knowledge base discussed later.

fashion the process used by the human mind to generate discourse; and the human mind makes use of its reasoning and memory in this process. Also the human memory contains a somewhat encyclopedic knowledge of the universe of discourse that enables a person to make semantic decisions about the words to be used in a given message. For this reason, the operating system should have access to a dictionary, a knowledge base, and an inference machine.[14] In this way, it emulates what an intelligent user does when he generates a message. Figure 2 represents an operating system with its access to the user, the dictionary, the knowledge base, the inference machine, and the various parts of the discourse grammar.

The system is not to be understood as functioning in a simple linear fashion, that is, that W operates on **E**, F operates on the product of W, U operates on the product of F, and D operates on the product of U, producing the final analysis of the message. On the contrary, the system iterates. Initially the clause grammar F analyzes some simple clauses that the phrase grammar U may use to analyze some phrases. These phrases in turn are used by F to analyze more complex clauses. The system iterates back and forth between F and U until all clauses have been analyzed up to the current state defined by the input data **E**. The sequence grammar D operates on the clauses analyzed by U and F deriving discourse structures. At times, D will produce a compound clause that must be used by U to generate an even more complex phrase. So the system iterates repeatedly between F, U, and D until the entire message has been analyzed. At each step of the iterative process the grammar updates its memory to record the current state of the process. The system advances through successive iterations to a terminal state that produces the desired analysis.

The Dictionary

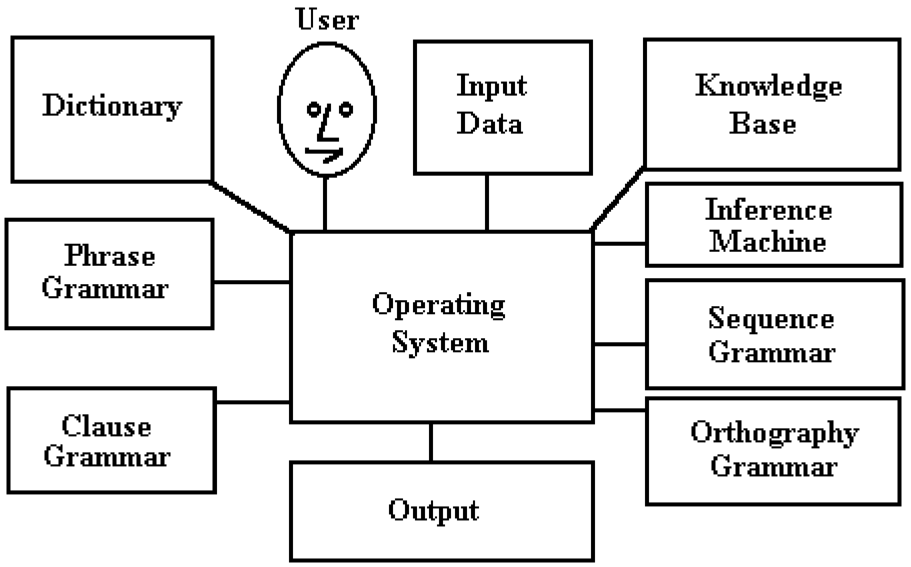
The dictionary in this case is a digital Hebrew dictionary that contains all the Hebrew words used in the Bible. It contains an entry for each entity, concept, relationship, or action to which a given word may refer. It identifies a word's part of speech, its class, and the values of its grammatical[15] and semantic attributes.[16] The inclusion of this additional information in the

[14] The inference machine is a separate computer program that makes logical inferences from the information in the data bases of the operating system. It is discussed on page 20.

[15] Grammatical attributes consist of number, gender, person, state, determination, negation, aspect, tense, mood, emphasis, etc. Not all of these attributes are active for any given part of speech. The values of some of these attributes are constants fixed by the dictionary. Others are variables determined by the operation of the system. By the time a word reaches its final position in the discourse all its active characteristic values are defined.

dictionary enables the grammar to view words as complex constituents with attributes that it can use to impose its grammatical and semantic restraints on the analysis of discourse.

Figure 2
The Operating System



The dictionary must be dynamic so that it is capable of being modified by the operating system as new information is acquired. Currently available dictionaries do not contain all the information specified above. An intelligent user already has the needed information in his encyclopedic knowledge of the language and the real world. Information not yet in the dictionary may be gradually added to it by the operating system.

The Knowledge Base

The knowledge base is a semantic model of the Biblical world containing information about the complex interrelationships of the elements of the world that enables the operating system, by means of the inference machine, to make intelligent decisions about the input data and other data files in the system. For example, it has recorded within its data base the information needed by the inference machine to deduce the relationship between son, father, grandfather, etc. It "knows" the relationship between house, building, temple, hut, etc. It "knows" the kinds of things that may be predicated about the vocabulary entries in the dictionary. Also, it has the capacity, by means of the inference machine and user interrogation, to update its knowledge—

[16] Semantic attributes, to name a few, consist of semantic categories, semantic characteristics, semantic capabilities, and for verbs, kinds of subjects and kinds of objects.

that is, to learn about the world of discourse as it interacts with the operating system and the user. For Bible translation, the content of the knowledge base needs to be sufficient only for the world of the Bible; it is type (2) context for the generation of Hebrew discourse.

The Inference Machine

The inference machine is a separate computer algorithm[17] that makes logical inferences from the information in the data bases of the operating system. The inferences are based on the knowledge and lawful relationships recorded in the knowledge base. By means of the inference machine, the operating system can make decisions that will avoid the necessity of asking the user trivial questions, and avoid generating trivial output data.

A Computer Aid for Translating Hebrew to English

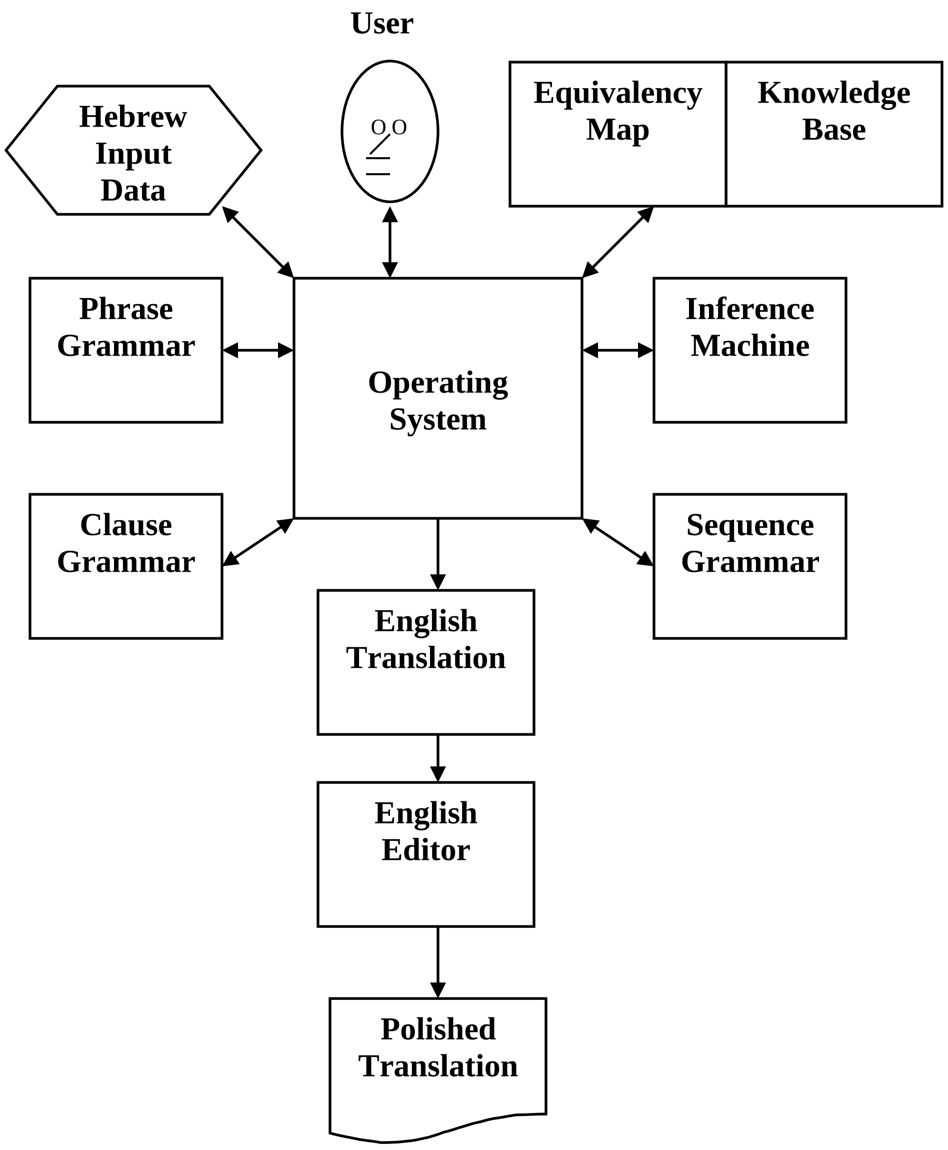
Previous sections describe an optimal equivalence theory of translation and a computer-aided system for analyzing Hebrew discourse. This section describes a computer aid for translating the Hebrew Bible into English. This system can serve as a template for providing a computer aid for translating the Hebrew Bible into any other language.

Figure 3 is a diagram of the translation system that assists a user in translating the Hebrew Bible into English. It consists of an operating system that manages the interchange of information between the Hebrew input data, the knowledge base, the equivalency map, the various English generative grammars, and the user. The knowledge base maps the Hebrew knowledge base to an equivalent English knowledge base. The equivalency map defines Hebrew-English equivalencies of words, phrases, clauses, transformations, and idioms. The user in this case is a person well-informed in Hebrew and English vocabulary, grammar, and syntax.[18] The initial translation passes through an English editor who polishes the wording of the text into good idiomatic English phraseology without introducing distortion.

[17] The algorithm is a separate computer program containing a set of rules for solving logical inferences.

[18] For translating into other languages, the user must be informed in the target language and have an informed native consultant.

Figure 3
The Translation System



The Hebrew input data consists of the output of the Hebrew analysis grammar, that is, an exhaustive analysis of a selected portion of the Hebrew Bible.[19] The analysis is wholly redundant, that is, the Hebrew analysis grammar restored all elements that were elided in the surface

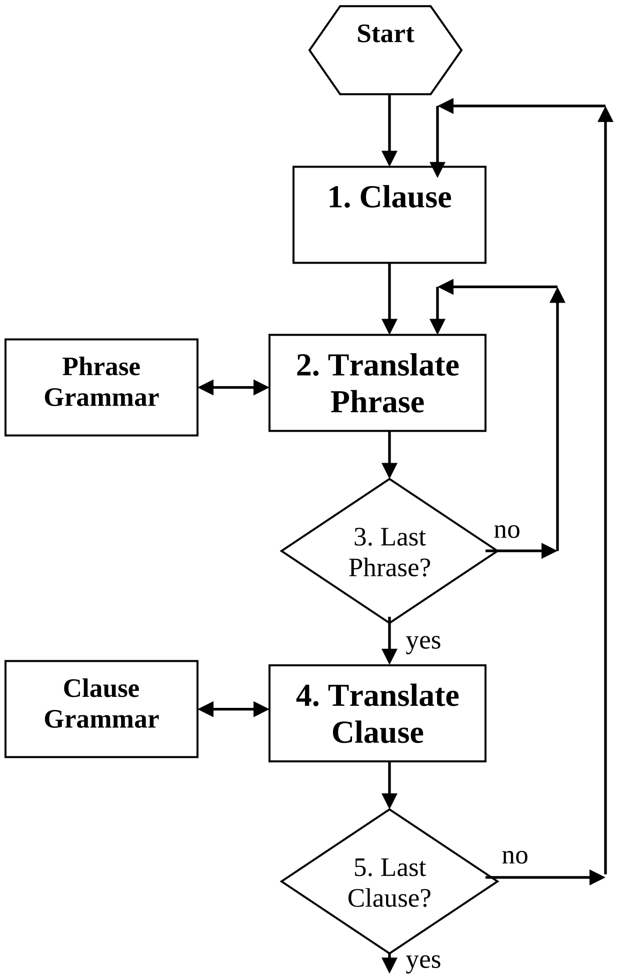
[19] Once such an analysis has been completed and validated, it may be stored in electronic form and used as the input for translating the passage into any other language.

text, and it replaced every pronoun and substitute by its antecedent. The analysis grammar also provides the full text of all clauses that are dependent on some part of speech.[20]

The Operating System

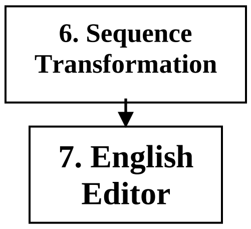
Figure 4 is a flow diagram of how the operating system manages the translation of a segment of the Hebrew Bible such as a paragraph or an episode.[21] The operation of each box is described in the material that follows.

Figure 4
The Operating System



[20] These dependent clauses provide the deep-structure derivation of all phrases. The provision of such clauses minimizes ambiguity and the possibility of a translator misunderstanding a phrase.

[21] The Hebrew analysis grammar marks the end of paragraphs and episodes.



1. Clause

This operation of the system receives a clause from the Hebrew input, having all its constituents in fully defined form.[22] It presents the constituents of the clause to the next function one at a time for translation.

2. Translate Phrase

This operation translates each constituent, treating it as a phrase and keeping the translation attached to the symbol that represents it. The operation of this function is outlined in more detail in Figure 5.

3. Last Phrase?

This operation determines whether the current phrase is the last constituent of the clause. If not, it returns the operation to (2) to translate the next phrase. This continues until all phrases of the clause have been translated, then the translated constituents are sent to operation (4).

4. Translate Clause

This operation arranges the translated constituents in the correct English order for the indicated clause type[23] as specified by the clause grammar. Semantic compatibility is tested here. If incompatibility is detected, the English vocabulary is adjusted to correct the problem.[24]

5. Last Clause?

This operation determines whether the current clause is the last one of the paragraph or episode, if not, it returns the operation to (1) to process the next clause. This continues until all clauses in the current segment are

[22] The constituents are represented by symbols which have a complete definition of all its composite Hebrew elements attached to it. At this stage of translation, the constituents are the subject, the verb, and the predicate complements, plus any adverbial modifiers.

[23] The clause type is defined in the input data as previously determined by the Hebrew analysis grammar. The equivalency map defines the equivalent English clause type.

[24] Alternate English equivalents of the involved Hebrew words are expected to resolve the incompatibility, otherwise, the user is asked for compatible words and the information in the knowledge base is revised.

translated, then the paragraph or episode is presented to sequence transformation (6) for minimizing redundancy.

6. Sequence Transformation

The sequence transformation deletes redundant constituents or replaces them with an equivalent substitute such as a pronoun.

7. English Editor

The English editor polishes the wording of the translation to smooth out any lack of good idiomatic English phraseology.

Figure 5 (next page) is an expansion of the inner workings of operation (2) above. It describes how Hebrew phrases are translated into English. The following discussion articulates the operation of the individual sections within Figure 5.

8. Next Constituent

This operation receives the next constituent of the clause being translated. The constituent is then evaluated by the next operation.

9. Is this item a phrase?

This operation determines whether the constituent is a single word or a phrase. A single word is immediately translated and the rest of the procedure is bypassed. Phrases are passed to the next operation.

10. Translate Word

When the constituent is a single Hebrew word, the equivalent English word replaces the Hebrew word attached to the associated symbol.

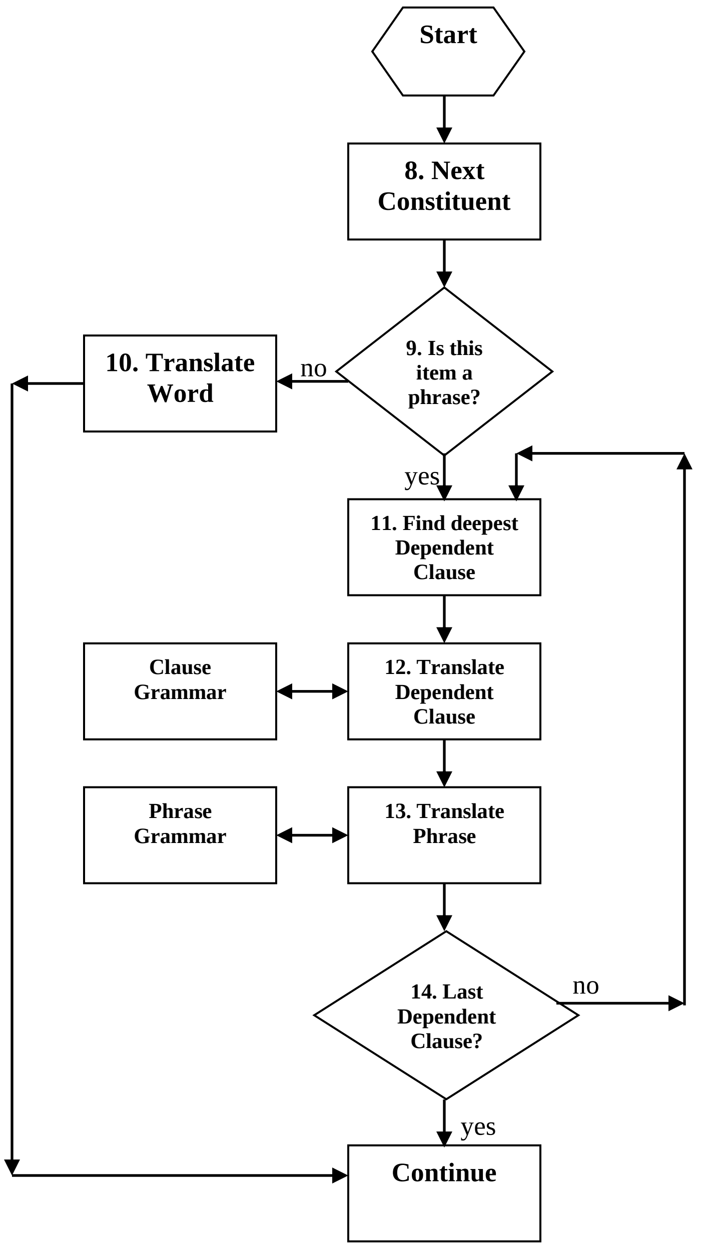
11. Find deepest Dependent Clause

Phrases have the general form $X \mid S(X)$, where $S(X)$ is a clause about X that is dependent on constituent X. Complex phrases have more than one imbedded dependent clause. This operation finds the most deeply imbedded clause and presents it for translation first.

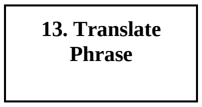
12. Translate Dependent Clause

This operation translates the dependent clause like that which takes place in Figure 4, except that its constituents have already been translated by the time the process reaches this stage. The inner workings of this operation are not repeated here.[25] The translation steps are described later.

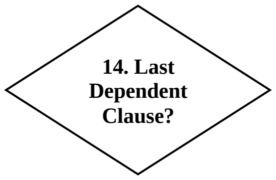
Figure 5
Translate Phrases


Start
8. Next Constituent
9. Is this item a phrase?
no
10. Translate Word
yes
11. Find deepest Dependent Clause
Clause Grammar
12. Translate Dependent Clause
Phrase Grammar
13. Translate Phrase
14. Last Dependent Clause?
no
yes
Continue

[25] The software coding here must avoid the problem of a self-imbedding procedure. This problem is solved by writing an independent clause translation procedure at this point. Because the clause will always be the most deeply imbedded one, no imbedding will occur.



This operation translates the phrase based on the indicated phrase type according to the associated rules in the phrase grammar. Then to the phrase it attaches the dependent clause from (12) above as a marginal note.



This operation determines whether the current clause being translated is the last dependent clause imbedded in the phrase being processed. If so, the process stops and returns to box (3) in Figure 4; if not the last dependent clause, the process returns to box (11) to find the next dependent clause. The process repeats until all the imbedded clauses in the given phrase have been translated.

Translating Phrases

Except for prepositional phrases, the Hebrew data input presents a phrase as a part of speech[26] with an attached dependent clause, along with the associated phrase type. The equivalent English phrase type indicates how the phrase is to be translated into English. According to the translation procedure outlined above, a dependent clause is translated prior to the phrase in which it is imbedded. Thus, a phrase is translated by providing the English equivalent of the part of speech under consideration, along with the pertinent English words from the translated dependent clause, as indicated by the syntax rule of the given phrase type. The words of the phrase have semantic compatibility because their compatibility was validated when the dependent clause was translated. The translated dependent clause is attached to the phrase as a marginal note. In addition, the Hebrew input data includes words and phrases that have been marked for emphasis, focus, or prominence; these marks require a corresponding marginal note in English. These notes provide commentary for the subsequent English editor in order to forestall possible misunderstanding.

Prepositional phrases are translated differently. Hebrew idiom and English idiom are quite different with respect to prepositions. The English equivalent of a Hebrew preposition depends on the object of the preposition and the semantic environment of the phrase—that is, the particular adverbial attribute to which the preposition refers. This must be determined by interrogating the knowledge base or the user. Thus, a prepositional phrase is translated by providing the English equivalent of the preposition followed by the translation of its object. This does not

[26] The part of speech may be a noun, adjective, adverb, or verb. Prepositional phrases are handled differently.

include prepositional phrases used as complements of a verb; these are discussed under the section on translating clauses.

Translating Clauses

By the time a clause is ready to be translated, all its constituents have been translated, such as its subject, verb, object, and any adverbial phrases. Thus, apart from checking semantic compatibility, the constituents are ready to be assembled in the syntactic order specified by the associated English clause type. Although the constituents of the Hebrew clause are semantically compatible as determined by the Hebrew analysis grammar, the initial choice of English equivalents for those Hebrew words may not exhibit the same compatibility. Consequently, the corresponding semantic compatibility of the English words must be verified before the translation is complete. The knowledge base (or the user) must be interrogated to see whether the English equivalent of the Hebrew subject may validly govern the English equivalent of the Hebrew verb of the clause, and whether the English equivalent of the Hebrew verb may validly govern the English equivalent of the associated Hebrew object(s). If incompatibility is encountered, the knowledge base (or user) must provide the alternate English equivalent of the associated Hebrew word that satisfies semantic compatibility.

Hebrew verbs govern their complements differently than English verbs do. Where a Hebrew verb governs a particular object in the accusative case, the equivalent English verb may govern the object by means of a preposition. Conversely, where a Hebrew verb governs a particular object by means of a preposition, the equivalent English verb may govern it in the accusative. On the other hand, where a Hebrew verb governs a particular object by means of a preposition, the equivalent English verb may govern the object by means of a different kind of preposition. There are no general rules. However, the knowledge base identifies how English verbs govern various objects, so the equivalencies should turn out in good English idiom.

English Sequence Transformation

Like Hebrew, English also employs elision and substitution in order to reduce unnecessary redundancy. Although the practice is similar in both languages, it is better to permit English grammar to govern how these linguistic practices function in English in order to have a more idiomatic translation. Examples are not provided here.

Elision

English elides redundant constituents if their omission does not result in obscurity or ambiguity. This is true for words, phrases, predicates, and even for conjunctions if they link the same kind of constituent.

Substitution

When elision would result in obscurity or ambiguity, English employs substitution in order to avoid unnecessary redundancy. The redundant constituent is replaced by a simple substitute depending on the part of speech involved; there should be no intervening similar part of speech with which the substitute may be confused as an antecedent. When the redundant constituent is a noun or noun phrase, the substitute is usually a pronoun in agreement with its antecedent for number, gender, person, and case; if pronoun substitution is not possible, the redundant noun is retained, marked as determinate.

This completes the description of a computer aid for translating the Hebrew Bible into English. An experimental computer program has been written that emulates the system described in this chapter. With the assistance of an informed user, it successfully translates narrative portions of the Hebrew Bible.[27] It writes an exhaustive analysis of the Hebrew text together with its equivalent English translation, including a tree diagram of the Hebrew syntax of each clause. The translation corresponds with the expectations of the optimal equivalence theory. The computer program is still in the process of development, particularly with respect to the production of the knowledge base[28] and the word equivalency map.[29] It is expected that the Hebrew analysis grammar will need further fine tuning to account for some of the rare peculiarities of the language; this includes the corresponding English equivalencies that will result. However, the system is sufficiently successful to warrant continued research and development. The appendix contains samples of analysis output.

[27] The user sometimes must identify completeness, compatibility, antecedents, or dependent clauses. This dependence on the user will diminish as development of a knowledge base and equivalency map advances.

[28] The necessary files exist for the production of a Hebrew-English knowledge base. However, the task requires considerable manual editing, together with processing Hebrew text through the translating system.

[29] A preliminary word equivalency map exits, but it is not yet tied into a knowledge base.

Governing Principles

Several basic principles govern how the translation software operates; they are as follows:

(1) The analysis of the Hebrew text is governed by syntax rather than by the accents.

(2) The analysis procedure works from bottom to top rather than top to bottom.

(3) The analysis procedure uses multiple passes, always analyzing the most deeply embedded constituents first.

(4) The analysis procedure uses an ordered set of analysis rules, requiring all lower level rules to be satisfied before moving to a higher ordered rule. The hierarchy is as follows:

Hierarchy 1 Analyze Pronouns;[30]
Hierarchy 2 Analyze Verb Phrases type 1;
Hierarchy 3 Analyze Verb Phrases type 2;
Hierarchy 4 Analyze Numbers;
Hierarchy 5 Analyze Adverb Phrase;
Hierarchy 6 Analyze Construct Adjective Phrase;
Hierarchy 7 Analyze Adjective Adverb Phrase;
Hierarchy 8 Analyze Noun Adjectives;
Hierarchy 9 Analyze Noun Adverbs;
Hierarchy 10 Analyze Noun Nouns;
Hierarchy 11 Analyze Construct Noun Phrases;
Hierarchy 12 Analyze Number Noun;
Hierarchy 13 Analyze Noun Number;
Hierarchy 14 Analyze Object Markers;
Hierarchy 15 Analyze Preposition Phrases;
Hierarchy 16 Analyze Compounds type 1;
Hierarchy 17 Analyze Compounds type 2;
Hierarchy 18 Analyze Imbedded Clauses;
Hierarchy 19 Find Clause Breaks;
Hierarchy 20 Find Subjects;
Hierarchy 21 Find Predicate Complements;
Hierarchy 22 Find Verb Predicates;
Hierarchy 23 Find Clauses type 1;
Hierarchy 24 Find Clauses type 2;
Hierarchy 25 Insert Missing Subjects;
Hierarchy 26 Add Adverb Phrases type 1;
Hierarchy 27 Add Adverb Phrases type 2;
Hierarchy 28 Analyze Idioms.

[30] These are names of actual analysis procedures that implement specific syntactical analysis rules. The order is not crucial for all procedures, but successful analyses result from this prescribed order.

(5) The knowledge base is organized according to part of speech, each part of speech having a different object type depending on its semantic, syntactic, and grammatical characteristics. Cross-connecting links define its internal structure.

(6) Every successful operation of an analysis procedure generates analytical output statements, preparing the order of generative operations for the target translation.

(7) Every analytical output statement has an equivalent synthesis rule for generating a corresponding component of the target translation.

(8) Synthesis of the target translation works from bottom to top following the order of the analysis output statements.

Description of the Software

The following is a description of the experimental software designed to assist translating from Hebrew to English. The program is written in Turbo Pascal using Delphi 5 to provide user interface. Currently the system is not connected to a knowledge base, so the user must answer all the questions and requests evoked by the program. The questions are of the following general types:

(1) Is this constituent complete (fully developed)?

(2) Are these constituents semantically compatible for the proposed relationship?

(3) Please provide the indicated information.

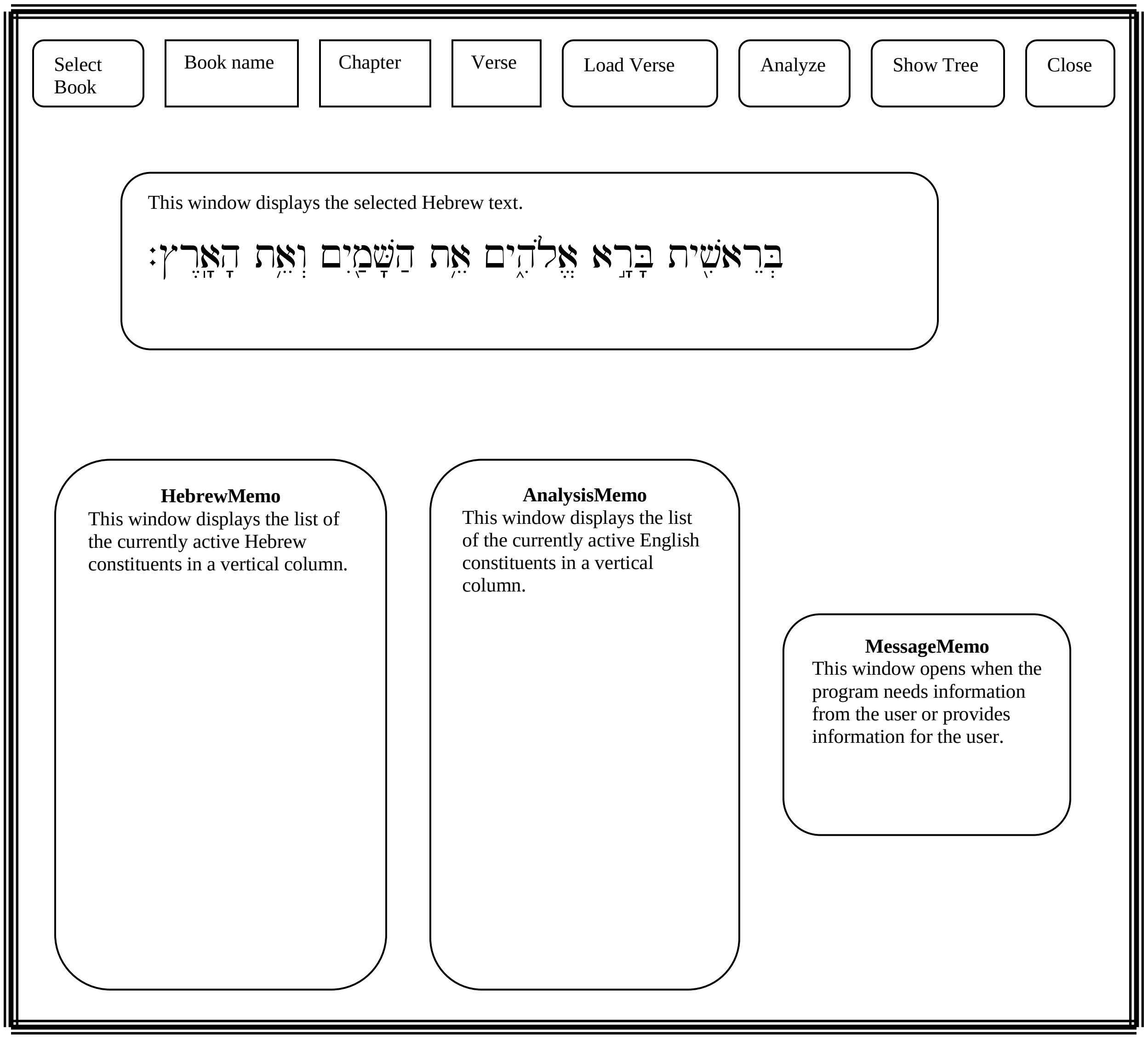
(4) Please identify the breaks between clauses.

The User Interface

The next page displays the structure of the current user interface. It has a button for selecting a book in which translation is to be made; clicking the button opens a drop-down radio button window with a button for each book. Selecting a book brings up toggle windows for selecting chapter and verse,[31] a button for loading the Hebrew text, a button for initiating the analysis, a button to display a tree diagram, and a button to close the program. When the Analyze button is clicked three memo windows are opened, one for displaying the Hebrew text of the selected verse, one for displaying a list of the currently active Hebrew constituents, and one for displaying a list of the currently active English constituents. A fourth memo window opens when the program needs input from the user or provides information to the user.

[31] The present program works with one verse at a time.

User Interface


Select Book
Book name
Chapter
Verse
Load Verse
Analyze
Show Tree
Close
This window displays the selected Hebrew text.
בְּרֵאשִׁ֖ית בָּרָ֣א אֱלֹהִ֑ים אֵ֥ת הַשָּׁמַ֖יִם וְאֵ֥ת הָאָֽרֶץ׃
HebrewMemo
This window displays the list of the currently active Hebrew constituents in a vertical column.
AnalysisMemo
This window displays the list of the currently active English constituents in a vertical column.
MessageMemo
This window opens when the program needs information from the user or provides information for the user.

The following contains an explanation of the computations and a logical flow chart for the syntactic analysis software for analyzing and diagramming Hebrew discourse. The central core of the program consists of two arrays of records: (1) the constituent array, and (2) the analysis array. Each record contains slots for the necessary data for a Hebrew word or phrase consisting of slots for an index number, a Hebrew word (or phrase), a lemma, a part of speech, a class, a set of morphological and syntactical codes, a parent, and a translation.

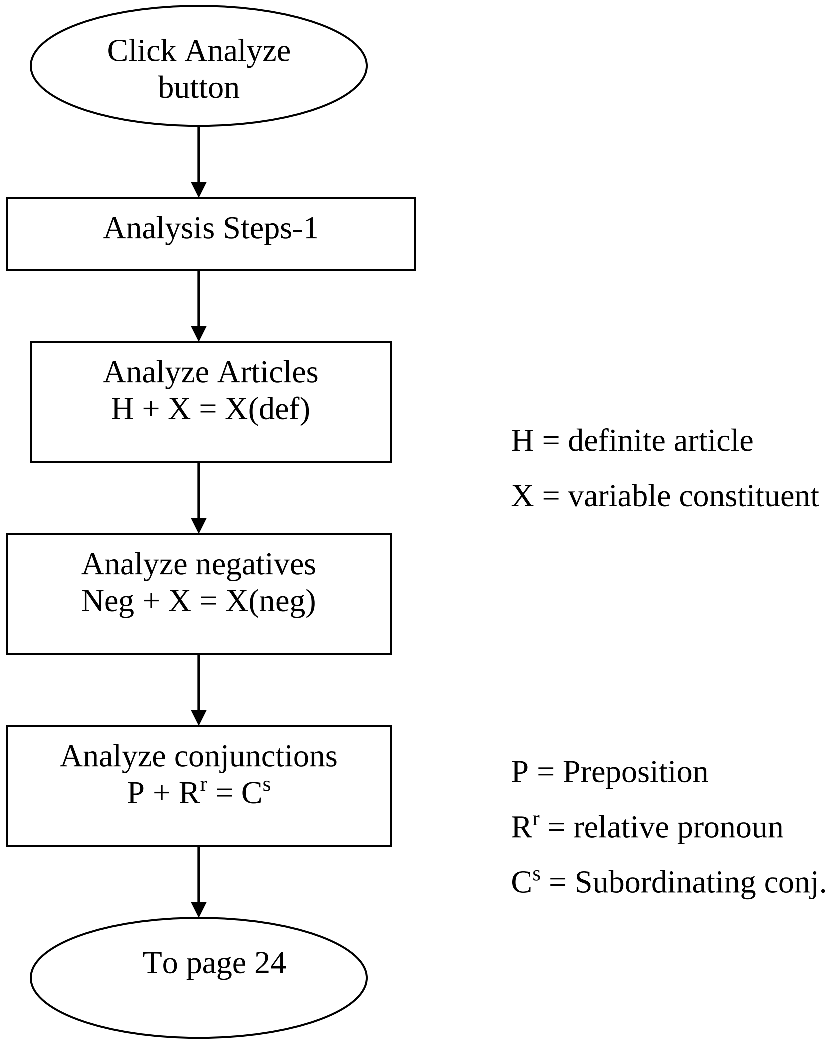
Initially, both arrays are loaded with the words of a selected verse of the Hebrew Bible. The words and associated data are obtained from the Westminster Morphological Hebrew Old Testament as modified to contain an English equivalent for each word (usually that of the NKJV).[32]

An ordered sequence of analysis rules operates on the elements in the analysis array. The rules are phrase-structure replacement rules of the form A + B = C in which A and B are contiguous elements in the analysis array, and C is a newly created constituent that is added to the constituent array and that replaces A and B in the analysis array. The data values for C are determined by the given rule and the associated requirements of concord. The rule cannot operate unless elements A and B are syntactically complete and grammatically and semantically compatible in the given linguistic environment. If the software cannot determine the completeness and compatibility of A and B, it will ask the user to decide. The rule makes a complete pass through all the elements in the analysis array, operating on all the elements that satisfy its conditions.

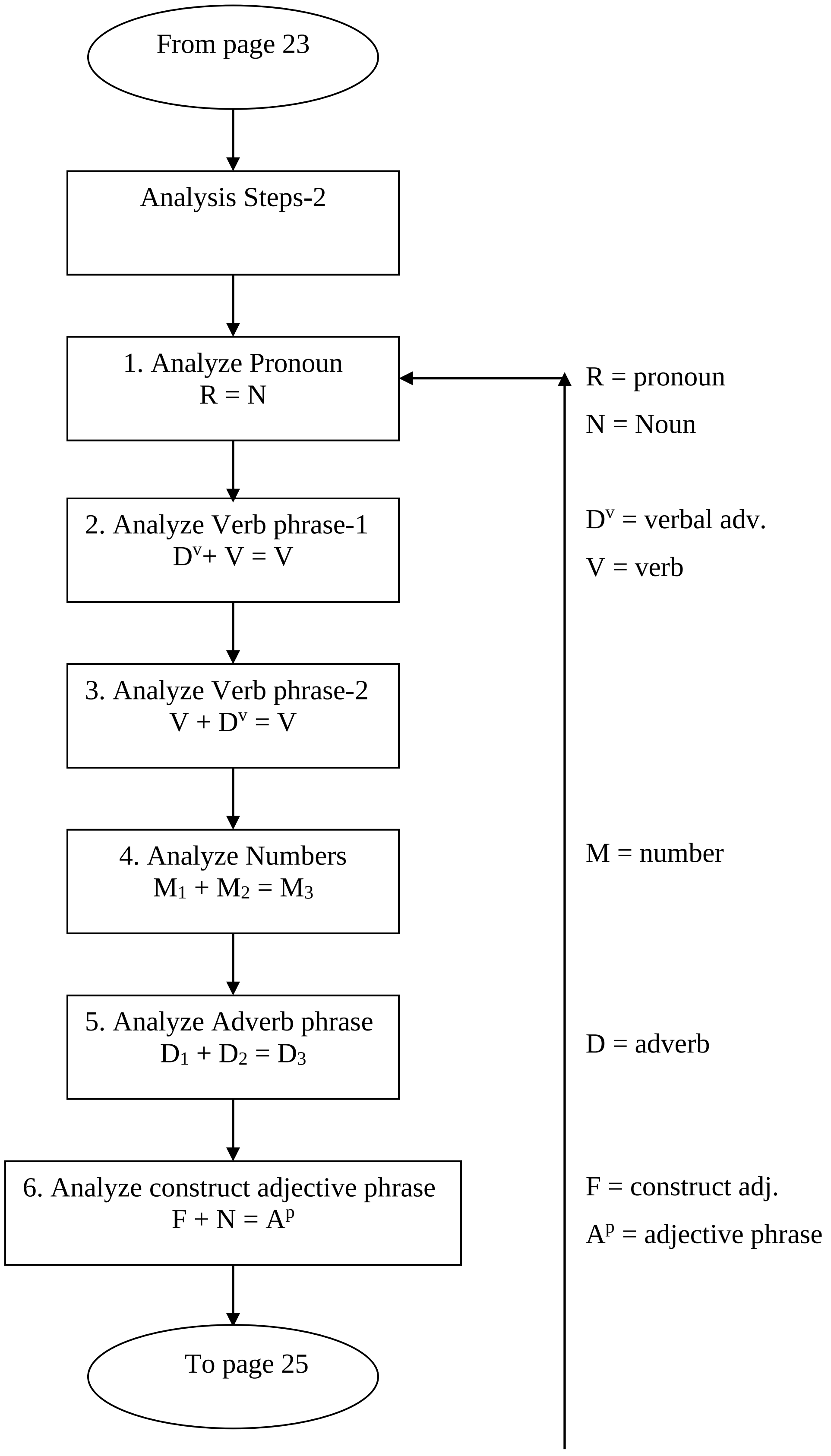
As the rules continue to operate, the elements in the constituent array increase in number, accumulating information about the analysis and translation; on the other hand, the elements in the analysis array diminish in number until no rules can be satisfied, at which time the analysis is

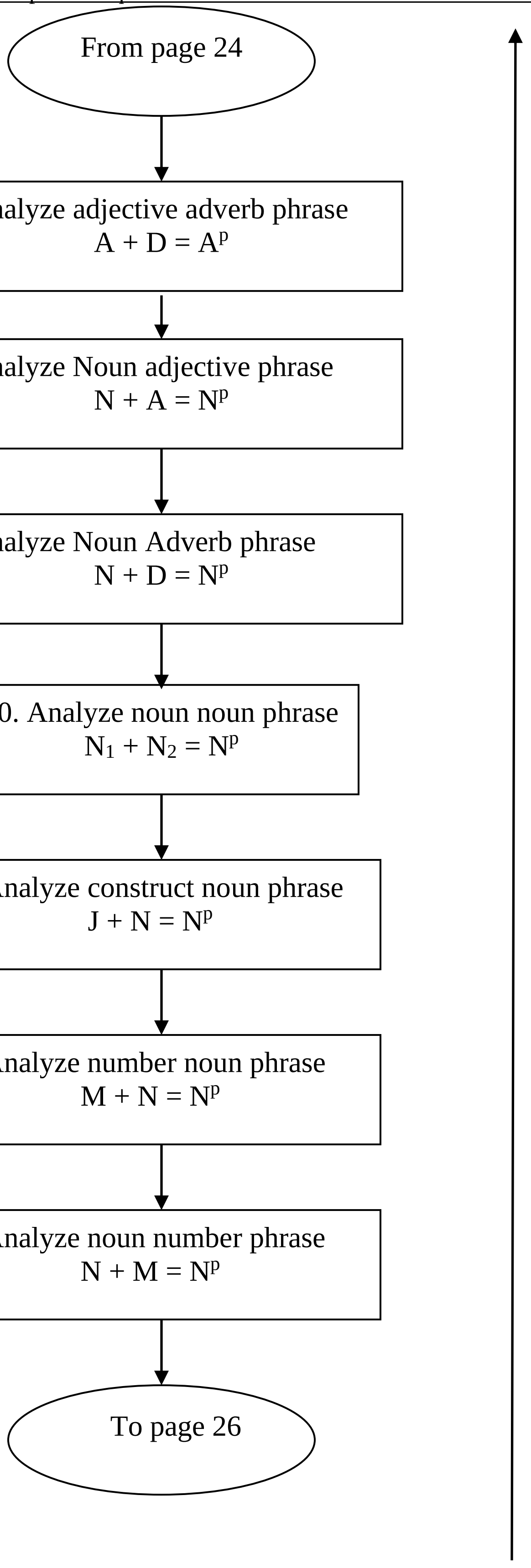
[32] The use of English words is a preliminary substitute for a lexicon and knowledge base for assigning semantic equivalents. English is the target language for the experimental design of the software, but, the translation part of the program can be accommodated to any target language.

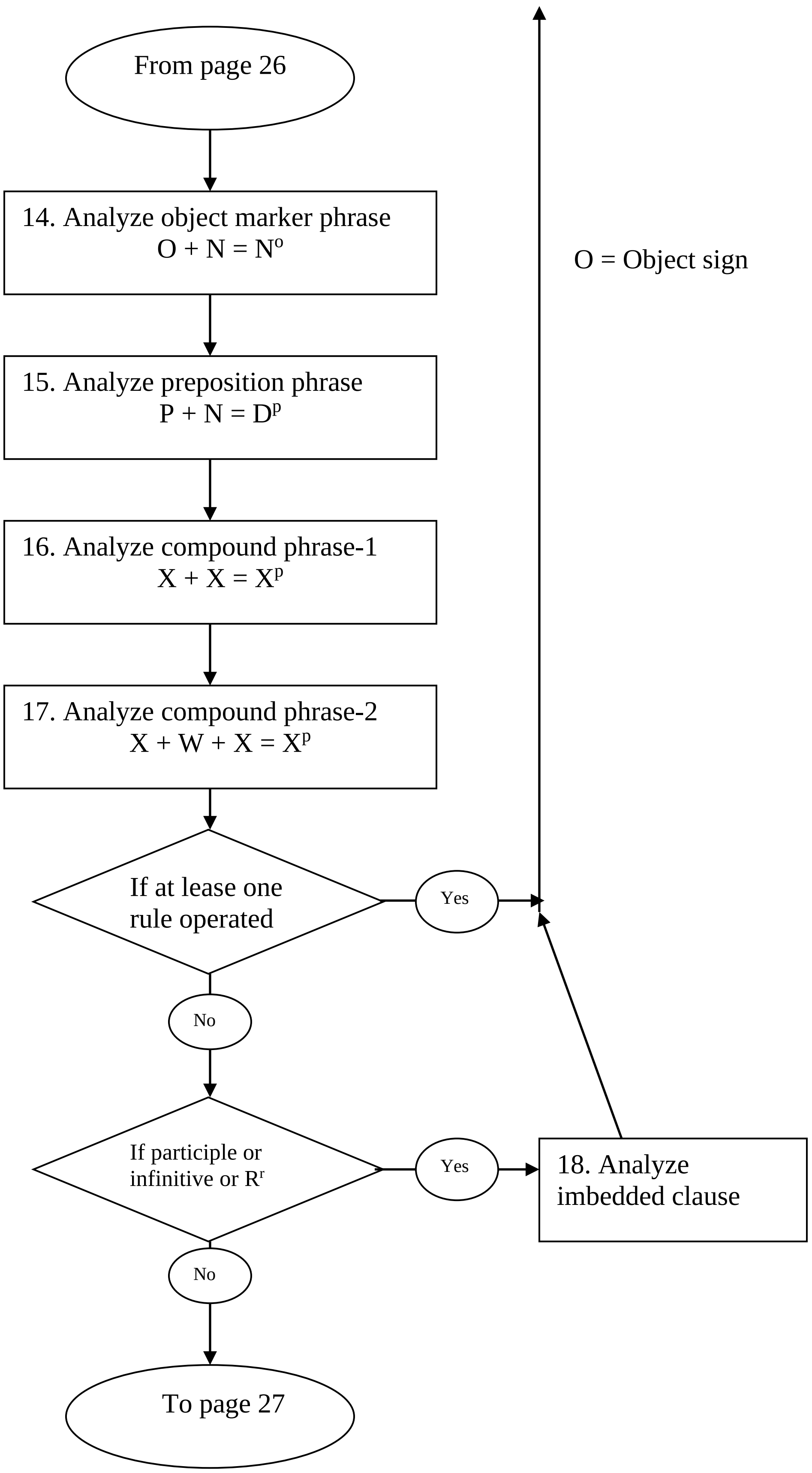
completed. The following flow chart outlines the skeleton of the program, indicating the step number and which rule is functioning. Subsequent discussion describes the logic of each step in more detail.

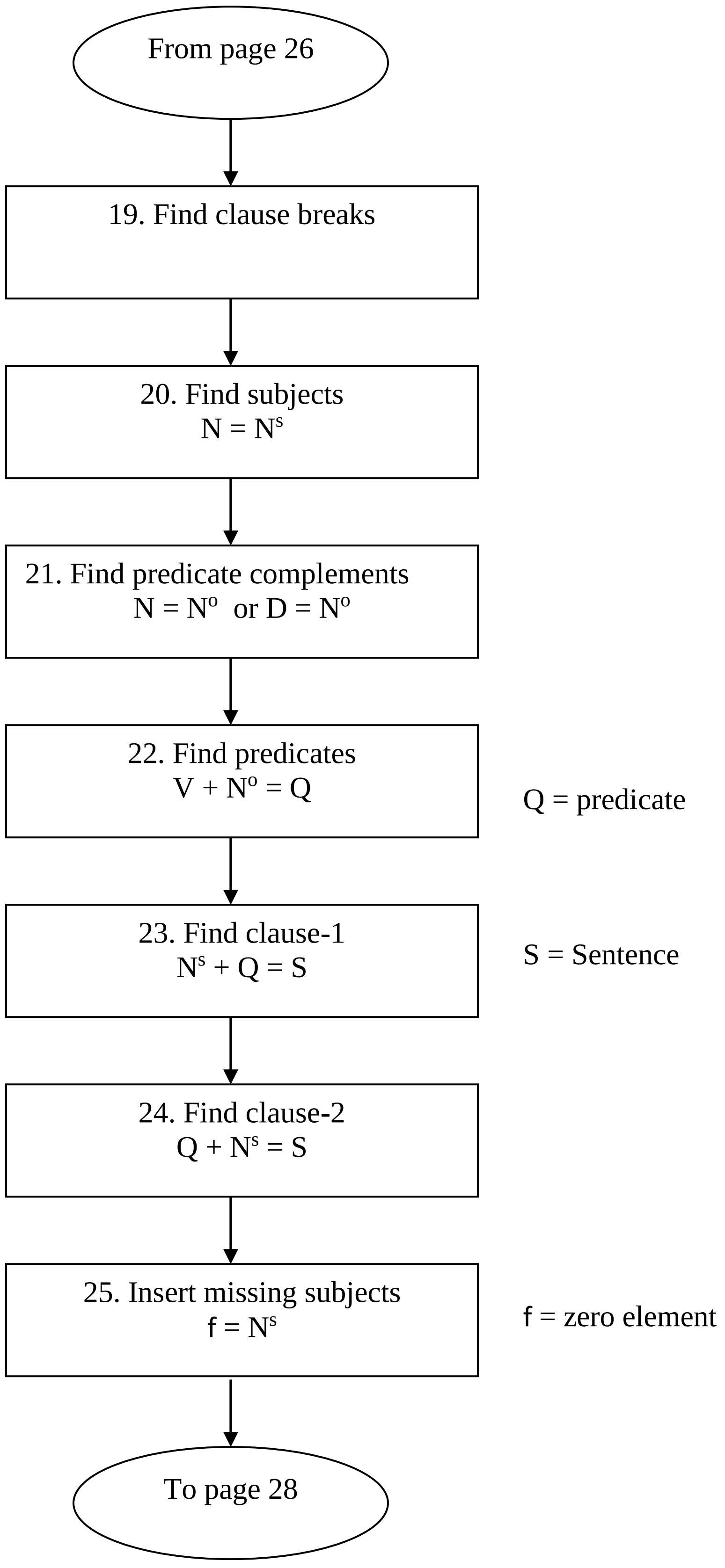


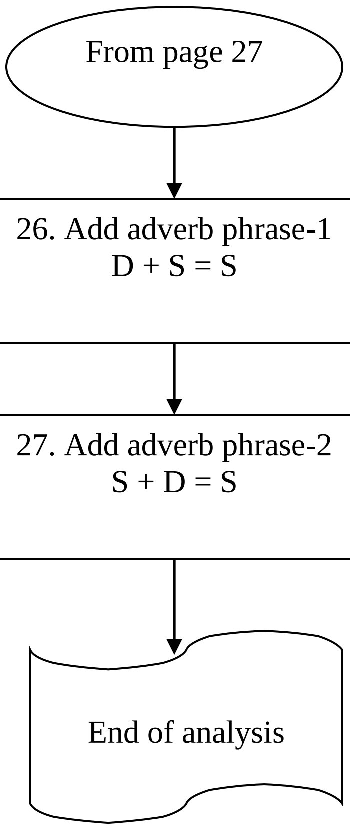
Procedure “AnalysisSteps-1” calls Procedures “AnalyzeArticles,” “AnalyzeNegatives,” and “AnalyzeConjunctions” that perform initial, non-iterative operations discussed later. It then initializes the step variable to 1 and calls Procedure “AnalysisSteps-2.”


From page 23
Analysis Steps-2
1. Analyze Pronoun
$R = N$
2. Analyze Verb phrase-1
$D^v + V = V$
3. Analyze Verb phrase-2
$V + D^v = V$
4. Analyze Numbers
$M_1 + M_2 = M_3$
5. Analyze Adverb phrase
$D_1 + D_2 = D_3$
6. Analyze construct adjective phrase
$F + N = A^p$
To page 25
R = pronoun
N = Noun
D^v = verbal adv.
V = verb
M = number
D = adverb
F = construct adj.
A^p = adjective phrase


From page 24
7. Analyze adjective adverb phrase
$A + D = A^p$
8. Analyze Noun adjective phrase
$N + A = N^p$
9. Analyze Noun Adverb phrase
$N + D = N^p$
10. Analyze noun noun phrase
$N_1 + N_2 = N^p$
11. Analyze construct noun phrase
$J + N = N^p$
12. Analyze number noun phrase
$M + N = N^p$
13. Analyze noun number phrase
$N + M = N^p$
To page 26


From page 26
14. Analyze object marker phrase
O + N = N^o
O = Object sign
15. Analyze preposition phrase
P + N = D^p
16. Analyze compound phrase-1
X + X = X^p
17. Analyze compound phrase-2
X + W + X = X^p
If at lease one rule operated
Yes
No
If participle or infinitive or R^r
Yes
18. Analyze imbedded clause
No
To page 27


From page 26
19. Find clause breaks
20. Find subjects
$N = N^s$
21. Find predicate complements
$N = N^o$ or $D = N^o$
22. Find predicates
$V + N^o = Q$
Q = predicate
23. Find clause-1
$N^s + Q = S$
S = Sentence
24. Find clause-2
$Q + N^s = S$
25. Insert missing subjects
$f = N^s$
f = zero element
To page 28



Operations Common to All Rules

The following operations are performed by every analysis rule and are not reiterated in the subsequent discussion of the analysis rules:

(1) The rule sweeps through the elements of the Analysis Array until it finds a sequence of elements that correspond with those to the left of the equal sign of the given rule.

(2) The contiguous elements in the Analysis Array that correspond with the left-hand elements of the rule are checked for grammatical concord,[33] syntactic completeness[34] and semantic compatibility[35] in the given linguistic context. If the logic of the program cannot determine these conditions, then it asks the user to decide.

[33] Grammatical concord varies from rule to rule; some rules do not involve this type of concord.

[34] A constituent is complete if no evidence of a modifier follows it, and if there is no evidence of it being a member of a compound constituent. This evidence varies from rule to rule.

[35] Semantic compatibility will eventually be determined primarily by a future knowledge base; in the meantime, the user makes this decision.

(3) If the left-hand elements are not in grammatical concord, syntactically complete, and semantically compatible, the rule is skipped, and the sweep continues; otherwise the program creates a new constituent corresponding to the right-hand element of the rule, assigns a Constituent Array index number to the new element, computes the values of all its variables, adds it as the next element in the Constituent Array, and completes the remaining operations that follow.

(4) The index number of the new element is assigned as the parent variable of the Analysis Array constituents that match the left-hand elements of the current rule. This operation provides the data for the subsequent construction of a tree diagram.

(5) A translation of the resultant phrase (or clause) is constructed based on the existing translation of the Analysis Array constituents that match the left-hand members of the given rule. The translation words are arranged according to the syntax of the target language generative rule that is equivalent to the Hebrew analysis rule currently in operation, including the equivalent transformations. This practice provides phrase-for-phrase and clause-for-clause equivalency throughout the entire procedure. Idioms that may be automatically detected are accommodated within the same operation. The resultant translation is assigned as a variable to the newly created constituent—the right-hand member of the rule.

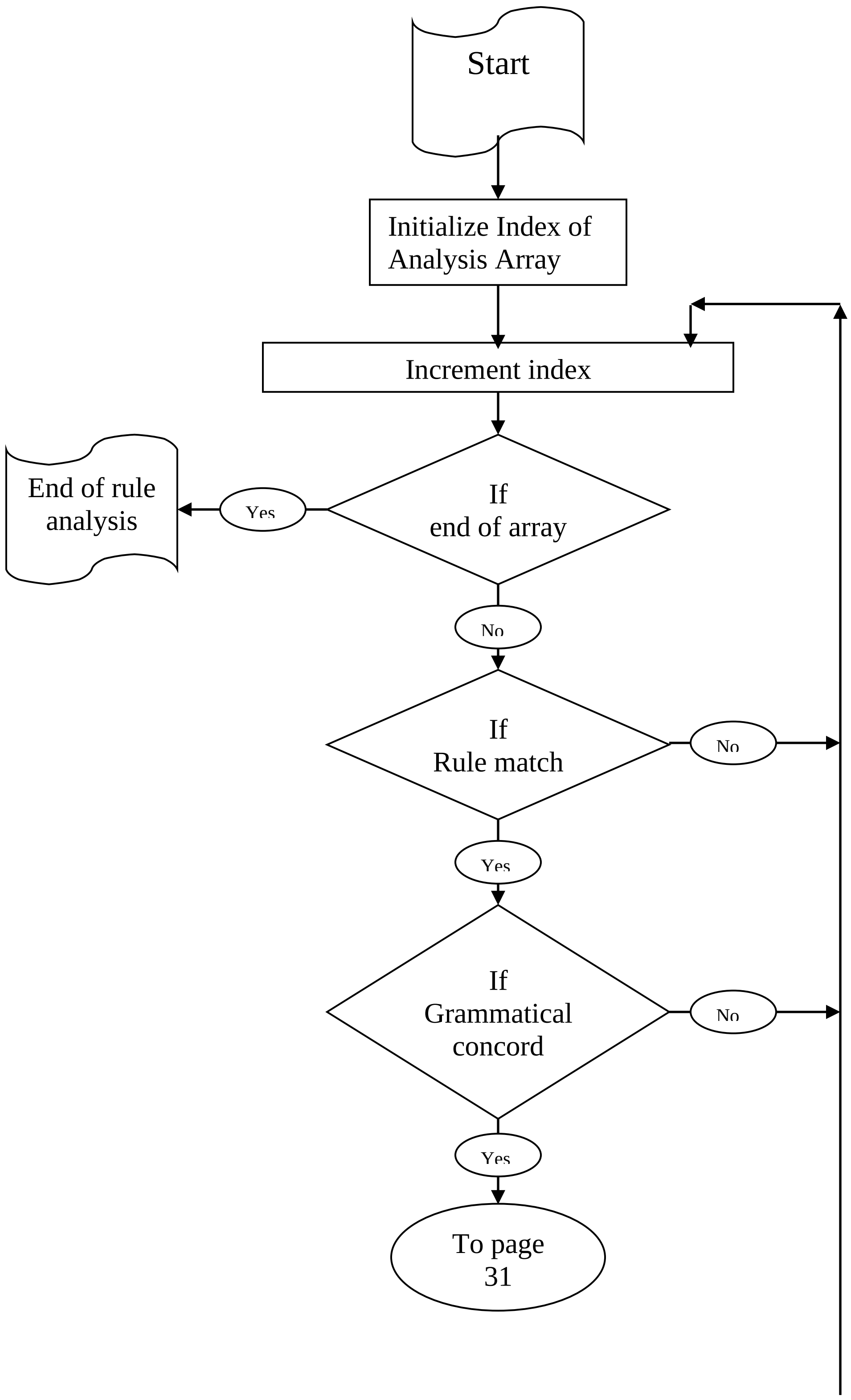
(6) The elements in the Analysis Array that match the left-hand elements of the rule are removed from the Analysis Array, and the newly created right-hand element takes their sequential place. The Analysis Array is then refreshed by making the remaining elements again contiguous; and the HebrewMemo[36] and the AnalysisMemo[37] windows are updated to display the new sequence of analysis elements. These windows keep the user informed of the progress on the analysis.

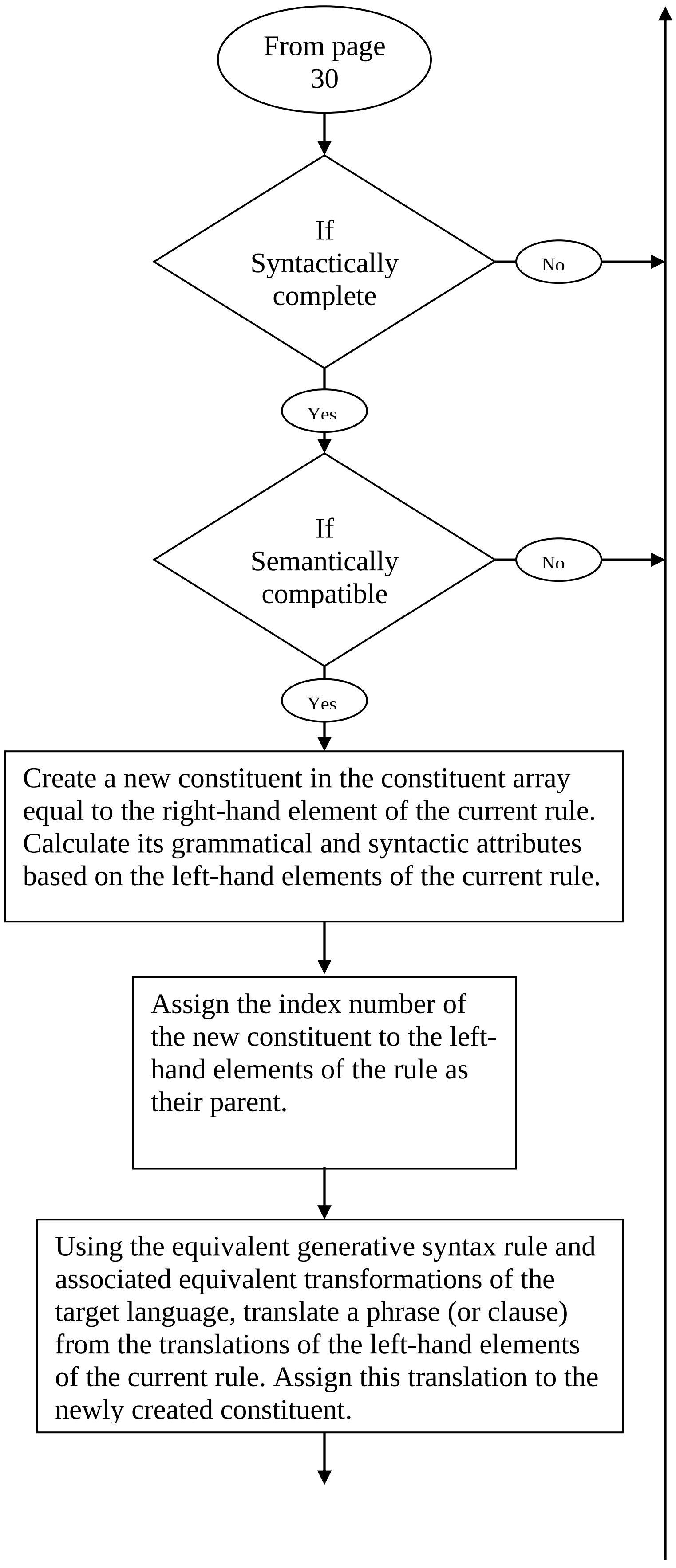
(7) A report is generated and displayed in the MessageMemo window to inform the user of the successful operation of the rule, and a similar report is generated and written to the output file for providing a hardcopy history of the analysis. This report includes the rule under opera-

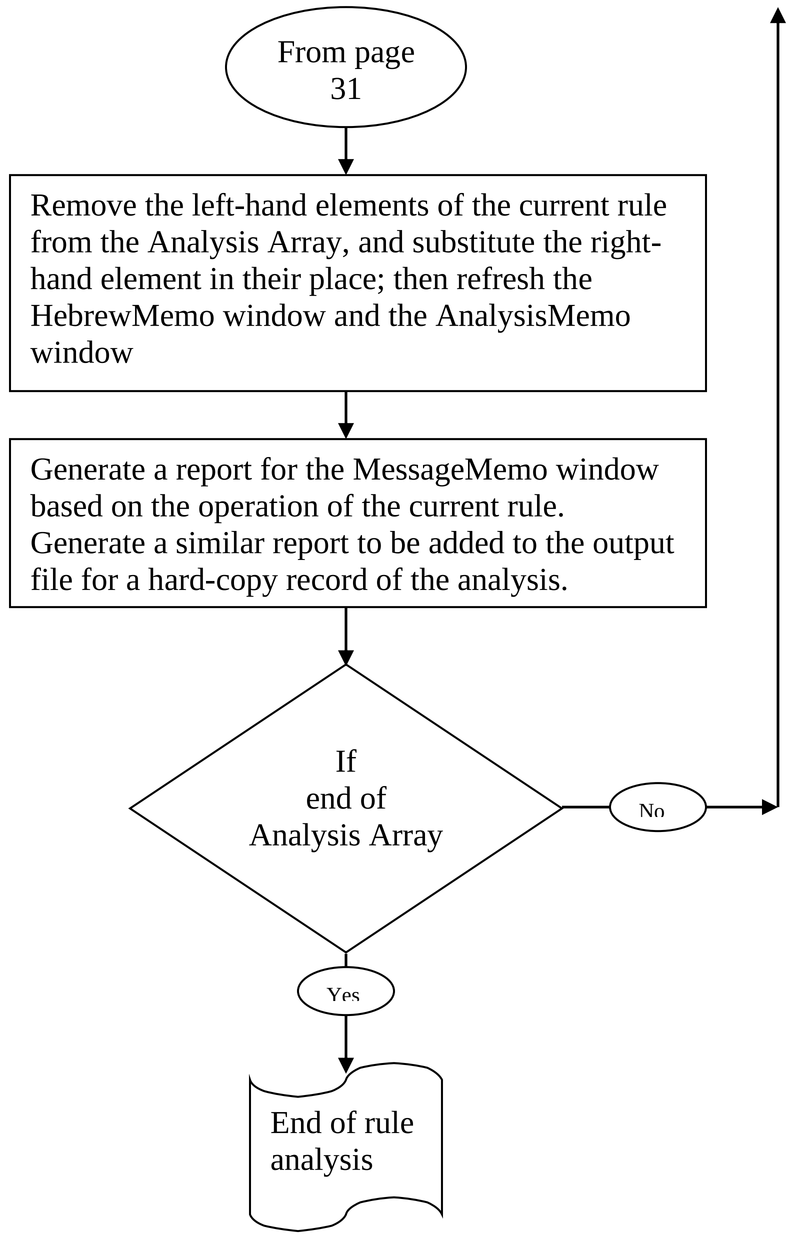
[36] The HebrewMemo window displays the current Hebrew constituents active in the analysis, that is, the index numbers and Hebrew phrases that have not as yet been combined with other constituents into higher level constituents.

[37] The AnalysisMemo window provides the same information as the HebrewMemo window except that the part of speech of the constituents is also listed and this window contains the translation of the constituent instead of its Hebrew equivalent.

tion, the explicit identity of its elements, and the resultant translation. The following flow chart diagrams the common operations for each analysis rule.




From page 30
If Syntactically complete
No
Yes
If Semantically compatible
No
Yes
Create a new constituent in the constituent array equal to the right-hand element of the current rule. Calculate its grammatical and syntactic attributes based on the left-hand elements of the current rule.
Assign the index number of the new constituent to the left-hand elements of the rule as their parent.
Using the equivalent generative syntax rule and associated equivalent transformations of the target language, translate a phrase (or clause) from the translations of the left-hand elements of the current rule. Assign this translation to the newly created constituent.


From page 31
Remove the left-hand elements of the current rule from the Analysis Array, and substitute the right-hand element in their place; then refresh the HebrewMemo window and the AnalysisMemo window
Generate a report for the MessageMemo window based on the operation of the current rule. Generate a similar report to be added to the output file for a hard-copy record of the analysis.
If end of Analysis Array
No
Yes
End of rule analysis

AnalysisSteps-1 Procedures

The following is a discussion of the logical operations for each procedure called by Procedure “AnalysisSteps-1”:

Procedure “AnalyzeArticles” makes one pass through the constituents in the Analysis Array operating the rule H + X = X(det); that is, it replaces any definite article and the constituent that follows it with the same constituent marked as determinate.

Procedure “AnalyzeNegatives” makes one pass through the constituents in the Analysis Array operating the rule Neg + X = X(neg); that is, it replaces any negative particle and the constituent that follows it with the same constituent marked as negated.

Procedure “AnalyzeConjunctions” makes one pass through the constituents in the Analysis Array operating the rule $P + R^r = C^s$; that is, any preposition followed by אֲשֶׁר is replaced by a subordinating conjunction C^s with its corresponding translation.

AnalysisSteps-2 Procedures

The following is a discussion of the logical operations for each procedure called by Procedure “AnalysisSteps-2”; the procedure called depends on the value of the variable “Step”:[38]

Step 1, Procedure “AnalyzePronouns” operates the rule R = N, which asks the user to identify the antecedent of the pronoun; the rule then replaces the pronoun with its antecedent enclosed in brackets. For example, if the pronoun is *he* and the antecedent is *Abraham*, the replacement noun phrase would be “he [= Abraham].” N acquires the person, gender, number, case, and determination of R. The same replacement is made in the target language.

Step 2, Procedure “AnalyzeVerbPhrase-1” operates the rule $D^v + V = V$, where D^v is the cognate infinitive absolute of V. The verb phrase and translation are marked for the emphasis of certainty or intensity. An English translation retains the same order.[39]

[38] If a rule procedure makes a complete pass through the elements of the Analysis Array without being satisfied, it indexes the variable “Step” by one, advancing to the rule next in order.

[39] Hebrew rules are interpreted in Hebrew right-to-left order, and the English rules are interpreted in left-to-right order.

Step 3, Procedure “AnalyzeVerbPhrase-2” operates the rule $V + D^v = V$, where D^v is the cognate infinitive absolute of V. The verb phrase and translation are marked for the emphasis of continuousness. An English translation retains the same order.

Step 4, Procedure “AnalyzeNumbers” operates the rule $M_1 + M_2 = M_3$ where two contiguous numbers are replaced by a number phrase.[40] An equivalence procedure transforms the Hebrew numbers into those of English.

Step 5, Procedure “AnalyzeAdverbPhrase” operates the rule $D_1 + D_2 = D_3$ where an adverb (phrase) D_2 modifies an immediately preceding adverb, D_2 being syntactically complete and semantically compatible with D_1 as a modifier. The equivalent English order is $D_2 + D_1$.

Step 6, Procedure “AnalyzeConstructAdjectivePhrase” operates the rule $F + N = A^p$ where N defines a restraint on the adjective F, such as naming the part to which it applies or the characteristic of which it is a value. The equivalent English phrase is “F of N.”

Step 7, Procedure “AnalyzeAdjectiveAdverbPhrase” operates the rule $A + D = A^p$ where adverb D modifies adjective A. The equivalent English phrase is D + A.

Step 8 , Procedure AnalyzeNounAdjectives” operates the rule $N + A = N^p$. The equivalent English order is A + N, where A carries the definite article, if any.

Step 9 , Procedure “AnalyzeNounAdverbs” operates the rule $N + D = N^p$. The equivalent English order is the same, N + D.

Step 10, Procedure “AnalyzeNounNouns” operates the rule $N_1 + N_2 = N^p$, where N_2 is an appositive. The equivalent English order is the same, $N_1 + N_2$.

[40] Compound numbers involving a conjunction are handled by Step 17.

Step 11, Procedure "AnalyzeConstructNounPhrases" operates the rule $J + N = N^p$. The equivalent English phrase is "J of N" which, in special cases, may be transformed into "N's J."

Step 12, Procedure "AnalyzeNumberNoun" operates the rule $M + N = N^p$. The equivalent English order is the same, $M + N$.

Step 13, Procedure "AnalyzeNounNumber" operates the rule $N + M = N^p$. The equivalent English order is the reverse, $M + N$, where M carries the definite article, of any.

Step 14, Procedure "AnalyzeObjectMarkers" operates the rule $O + N = N^o$. This rule has no English equivalent. The noun phrase is merely marked as an object of the related verb phrase.

Step 15, Procedure "AnalyzePrepositionPhrases" operates the rule $P + N = D^p$. The order is the same in English, $P + N$. All prepositional phrases are initially defined as adverbial phrases. A later step (21) transforms adverbial phrases that complement a verb into an accusative noun phrase, N^o.

Step 16, Procedure "AnalyzeCompoundsI" operates the rule $X_a + X_b = X^p$. The equivalent English order is "X_a, X_b."

Step 17, Procedure "AnalyzeCompoundsII" operates the rule $X_a + W + X_b = X^p$. The equivalent English order is "X_a and X_b." If the rule operates more that once in immediate succession, the equivalent English order is X_a, X_b, X_c, . . . , and X_n. If the Hebrew text is already in that structure, steps 16 and 17 operate in tandem to produce the same structure in English.

Step 18, Procedure "AnalyzeImbeddedClauses" has no specific initial syntactic rule. By the time the analysis reaches this step, all the phrases that can be analyzed have been analyzed. This procedure sweeps in reverse order through the Analysis Array searching for a participle, an infinitive construct, or a relative pronoun. If it finds one, it locates the end of the associated clause, and, within the limits of the given clause, it sweeps through the elements of the clause while operating steps 19 through 27 in order. These steps complete the analysis of the clause and then transform the clause into its equivalent part of speech. Thus an infinitive construct clause is transformed into its equivalent noun phrase, $S^i \Rightarrow N^p$; a participle clause into its equivalent adjective phrase, $S^{va} \Rightarrow A^p$; and a relative pronoun clause into its equivalent noun phrase $S^r \Rightarrow N^p$. English translates a participle as a participle and an infinitive absolute as an infinitive (or a gerund). After completing the analysis of an imbedded clause, the procedure resets the step index to one, causing the program to analyze any phrases that may now be open for analysis by the creation of a new noun or adjective phrase. Thus the program cycles through steps 1 through 18 until all imbedded clauses are analyzed, and then advances to step 19.

Step 19, Procedure “FindClauseBreaks” finds the boundaries of the clauses within a given verse. By this stage in the analysis, all remaining conjunctions in the Analysis Array stand between clauses. Thus conjunctions mark clause boundaries. If the verse has more verbs than conjunctions (excluding an initial conjunction), then the user is asked to identify the unknown clause boundary.

Step 20, Procedure “FindSubjects” operates the rule $N = N^s$. It checks the grammatical concord and the semantic compatibility of a given noun phrase with the verb within the boundaries of the associated clause. If the phrase is compatible, it is marked in the nominative case and identified as the subject of the clause. The phrase has already been translated. A noun phrase that is not marked as the subject of the clause is automatically marked as an object.

Step 21, Procedure “FindPredicateComplements” operates the rule $D = N^o$ and $N = N^o$. Adverb phrases[41] are tested for semantic compatibility as a complement the verb of the clause. Those that are compatible are transformed into predicate complements, N^o. Any unmarked noun phrases are also transformed into complements. Translation has already been made.

[41] In step 15 all prepositional phrases were interpreted as adverb phrases. Because Hebrew verbs often govern a complement by means of a preposition, this step identifies such prepositional phrases as complements of the verb.

Step 22, Procedure “FindVerbPredicates” operates the rule V + N^o = Q. This procedure is flexible. The verb may have from zero to two complements. The procedure finds the verb first and then its complement, if any. A complement that precedes the verb is marked as position prominent. A clause with no verb is marked as verbless and treated as defective copulative clause. The English verb is followed by its complements in the order found in the Hebrew text.

Step 23, Procedure “FindClauses1” operates the rule N^s + Q = S. This rule follows the assumed natural order of Hebrew clauses—subject, verb, object. English follows the same order.

Step 24, Procedure “FindClauses2” operates the rule Q + N^s = S. This rule follows the marked order of verb, subject, object, or verb, object, subject. If the verb is with Waw consecutive, then the clause is marked so; otherwise the clause is marked as having its verb in position prominence. English usually translates the clause as subject, verb, object; however, a position prominent object may be retained in English.

Step 25, Procedure “InsertMissingSubjects” operates the rule ϕ = N^s. Some clauses have no named subject because the subject is a pronoun inferred by the conjugate form of the verb. If the program finds a clause with no named subject, this step asks the user to provide the elided subject. The English translation attaches this subject to the subject pronoun implicit in the verb phrase.

Step 26, Procedure “AddAdverbPhrases1” operates the rule D + S = S. An adverb phrases that does not get absorbed into the phrases of a clause are regarded as an adverbial modifier of the clause. In the rule of this step, the adverb phrase is in first position prominence. It is mark so, and kept in first position in the English translation.

Step 27, Procedure “AddAdverbPhrases2” operates the rule S + D = S. This step is the complement of step 26. The adverb phrase is regarded as in its normal position. The English translation keeps the phrase in final position.

Tree Diagrams

The program provides two tree diagrams of the analyzed verse: one is generated by the internal software package provided by Delphi-5. Procedure "TreeView" constructs the tree deriving the data from the parent attribute of the constituents in the Constituent Array (WordAr-ray). The tree may be viewed on screen at any stage of analysis.

The second tree diagram is generated by Procedure "DrawTree." This procedure also makes use of the data from the parent attribute of the constituents in the Constituent Array. However, this tree is not constructed until the analysis of a verse is completed, and it is produced in the output file for viewing on hard copy. The appendix provides two examples of the output of the software together with the tree diagrams generated by the software. At this stage the resultant translations have not yet been edited by an English stylist and editor. The software needs much further experimental development, but the current results are very promising.

Appendix

This appendix provides two examples of the output of the software. The first is the output for Genesis 1:1, a very simple verse consisting of one sentence. The second is the output for Esther 6:8, selected because it has three embedded relative pronoun clauses, two of which are coordinate modifiers of the same noun—the horse.

Example 1:

Output data for 01gn 1:1

Reference = 1:1

בְּרֵאשִׁ֖ית בָּרָ֣א אֱלֹהִ֑ים אֵ֥ת הַשָּׁמַ֖יִם וְאֵ֥ת הָאָֽרֶץ׃

[1]	בְּ-	In
[2]	רֵאשִׁ֖ית-	the beginning
[3]	בָּרָ֣א-	created
[4]	אֱלֹהִ֑ים-	God
[5]	אֵ֥ת-	#
[6]	הַ-	the
[7]	שָּׁמַ֖יִם-	heavens
[8]	וְ-	and
[9]	אֵ֥ת-	#
[10]	הָ-	the
[11]	אָֽרֶץ׃-	earth

H[6]-the + Nc[7]-heavens ==> Nc[7]-heavens

H[10]-the + Nc[11]-earth ==> Nc[11]-earth

"Nc[7]-the heavens" is a complete phrase governed by the sign of the direct object O[5]-#

O[5]-# + Nc[7]-the heavens ==> No-[12]the heavens

"Nc[11]-the earth" is a complete phrase governed by the sign of the direct object O[9]-#

O[9]-# + Nc[11]-the earth ==> No-[13]the earth

"Nc[2]-the beginning" is a complete phrase governed by preposition "P[1]-In "

P[1]-In + Nc[2]-the beginning ==> Dp-[14]In the beginning

No[12]-the heavens + W[8]-and + No[13]-the earth ==> N-[15]the heavens and the earth

Nc[4]-God ==> Ns-[4]God

Dp[14]-In the beginning ==> Dp-[14]In the beginning

V[3]-created + No[15]-the heavens and the earth ==> Q-[16]created the heavens and the earth

The verb is in first position prominance

Q[16]-created the heavens and the earth + Ns[4]-God ==> S-[17]God created the heavens and the earth

"Dp[14]-In the beginning" modifies "S[17]-God created the heavens and the earth "

The adverb phrase is in first position prominence

Dp[14]-In the beginning + S[17]-God created the heavens and the earth ==> S-[18]In the beginning God created the heavens and the earth

Syntactic Tree of 01gn 1:1

```
 |Treex[0]-Trunk
 |Sx[18]-In the beginning God created the heavens and the earth
    |Dp[14]-In the beginning
    |   |Px[1]-In
    |   |Nc[2]-the beginning
    |Sx[17]-God created the heavens and the earth
       |Ns[4]-God
       |Qx[16]-created the heavens and the earth
          |Vx[3]-created
          |No[15]-the heavens and the earth
             |No[12]-the heavens
             |   |Ox[5]-#
             |   |Nc[7]-heavens
             |       |Hx[6]-the
             |Wx[8]-and
             |No[13]-the earth
                |Ox[9]-#
                |Nc[11]-earth
                   |Hx[10]-the
```

End of Tree Diagram

Example 2:

Output data for 17es doc

Reference = 6:8

יָבִ֙יאוּ֙ לְב֣וּשׁ מַלְכ֔וּת אֲשֶׁ֥ר לָֽבַשׁ־בּ֖וֹ הַמֶּ֑לֶךְ וְס֗וּס אֲשֶׁ֨ר רָכַ֤ב עָלָיו֙ הַמֶּ֔לֶךְ
וַאֲשֶׁ֥ר נִתַּ֛ן כֶּ֥תֶר מַלְכ֖וּת בְּרֹאשֽׁוֹ׃

[1] יָבִ֙יאוּ֙- let be brought
[2] לְב֣וּשׁ- a robe
[3] מַלְכ֔וּת- royalty
[4] אֲשֶׁ֥ר- which
[5] לָֽבַשׁ־- has worn
[6] בּ֖וֹ- #
[8] הַ- the
[9] מֶּ֑לֶךְ- king
[10] וְ- and
[11] ס֗וּס- a horse
[12] אֲשֶׁ֨ר- which
[13] רָכַ֤ב- has ridden
[14] עָלָיו֙- on it
[16] הַ- the
[17] מֶּ֔לֶךְ- king
[18] וַ- and
[19] אֲשֶׁ֥ר- which
[20] נִתַּ֛ן- has placed
[21] כֶּ֥תֶר- a crest
[22] מַלְכ֖וּת- royalty
[23] בְּ- on
[24] רֹאשֽׁוֹ׃- it head

H[8]-the + Nc[9]-king ==> Nc[9]-king

H[16]-the + Nc[17]-king ==> Nc[17]-king

R[7]- ==> Nx-[26] [= the robe]

R[15]-it ==> Nx-[27]it [= the horse]

R[25]-it ==> Nx-[28]it [= the horse]

Jc[2]-a robe + Nc[3]-royalty ==> Np-[29]a robe of royalty

Jc[21]-a crest + Nc[22]-royalty ==> Np-[30]a crest of royalty

"N[28]-it [= the horse]" is a complete modifier of "Jc[24]- head "

Jc[24]- head + N[28]-it [= the horse] ==> Np-[31] head of it [= the horse]

P[6]-# + N[26]- [= the robe] ==> Dp-[32]# [= the robe]

P[14]-on + N[27]-it [= the horse] ==> Dp-[33]on it [= the horse]

"Np[31]- head of it [= the horse]" is a complete phrase governed by preposition "P[23]-on "

P[23]-on + Np[31]- head of it [= the horse] ==> Dp-[34]on head of it [= the horse]

Np[30]-a crest of royalty ==> No-[30]a crest of royalty

Dp[34]-on head of it [= the horse] ==> No-[34]on head of it [= the horse]

V[20]-has placed + No[30]-a crest of royalty + No[34]-on head of it [= the horse] ==> Q-[35]has placed a crest of royalty on head of it [= the horse]

Ns-[37] [=someone] + Q[35]-has placed a crest of royalty on head of it [= the horse] ==> S-[37] [=someone]has placed a crest of royalty on head of it [= the horse]

Rr[19]-which + S[35]- [=someone]has placed a crest of royalty on head of it [= the horse] ==> Nr[38]-which [=someone]has placed a crest of royalty on head of it [= the horse]

Nc[17]-the king ==> Ns-[17]the king

Dp[33]-on it [= the horse] ==> No-[33]on it [= the horse]

V[13]-has ridden + No[33]-on it [= the horse] ==> Q-[39]has ridden on it [= the horse]

The verb is in first position prominance

Q[39]-has ridden on it [= the horse] + Ns[17]-the king ==> S-[40]the king has ridden on it [= the horse]

Rr[12]-which + S[40]-the king has ridden on it [= the horse] ==> Nr[41]-which the king has ridden on it [= the horse]

Nr[41]-which the king has ridden on it [= the horse] + W[18]-and + Nr[38]-which [=someone]has placed a crest of royalty on head of it [= the horse] ==> N-[42]which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse]

Nc[11]-a horse + Nr[42]-which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse] ==> Np-[43]a horse which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse]

Nc[9]-the king ==> Ns-[9]the king

Dp[32]-# [= the robe] ==> No-[32]# [= the robe]

V[5]-has worn + No[32]-# [= the robe] ==> Q-[44]has worn # [= the robe]

The verb is in first position prominence

Q[44]-has worn # [= the robe] + Ns[9]-the king ==> S-[45]the king has worn # [= the robe]

Rr[4]-which + S[45]-the king has worn # [= the robe] ==> Nr[46]-which the king has worn # [= the robe]

Np[29]-a robe of royalty + Nr[46]-which the king has worn # [= the robe] ==> Np-[47]a robe of royalty which the king has worn # [= the robe]

Np[47]-a robe of royalty which the king has worn # [= the robe] + W[10]-and + Np[43]-a horse which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse] ==> N-[48]a robe of royalty which the king has worn # [= the robe] and a horse which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse]

Np[48]-a robe of royalty which the king has worn # [= the robe] and a horse which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse] ==> Ns-[48]a robe of royalty which the king has worn # [= the robe] and a horse which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse]

V[1]-let be brought ==> Q-[49]let be brought

The verb is in first position prominence

Q[49]-let be brought + Ns[48]-a robe of royalty which the king has worn # [= the robe] and a horse which the king has ridden on it [= the horse] and which [=someone]has placed a crest of royalty on head of it [= the horse] ==> S-[50] a robe of royalty which the king has worn # [= the robe] and a horse which the king has ridden on it [= the horse] and which [=someone] has placed a crest of royalty on head of it [= the horse] let be brought

```
Syntactic Tree of 17es txt 6:8
  |Treex[0]-
  |Sx[50]-
      |Ns[48]-
      |   |Np[43]-
      |   |   |Nc[11]-
      |   |   |Nr[42]-
      |   |       |Nr[38]-
      |   |       |   |Rr[19]-
      |   |       |   |Sx[35]-
      |   |       |       |Ns[36]-
      |   |       |       |Qx[37]-
      |   |       |           |Vx[20]-
      |   |       |           |Np[30]-
      |   |       |           |   |Jc[21]-
      |   |       |           |   |Nc[22]-
      |   |       |           |Dp[34]-
      |   |       |               |Px[23]-
      |   |       |               |Np[31]-
      |   |       |                   |Jc[24]-
      |   |       |                   |Nx[28]-
      |   |       |                       |Rx[25]-
      |   |       |Wx[18]-
      |   |       |Nr[41]-
      |   |           |Rr[12]-
      |   |           |Sx[40]-
      |   |               |Ns[17]-
      |   |               |   |Hx[16]-
      |   |               |Qx[39]-
      |   |                   |Vx[13]-
      |   |                   |Dp[33]-
      |   |                       |Px[14]-
      |   |                       |Nx[27]-
      |   |                           |Rx[15]-
      |   |Wx[10]-
      |   |Np[47]-
      |       |Np[29]-
      |       |   |Jc[2]-
      |       |   |Nc[3]-
      |       |Nr[46]-
      |           |Rr[4]-
      |           |Sx[45]-
      |               |Ns[9]-
      |               |   |Hx[8]-
      |               |Qx[44]-
      |                   |Vx[5]-
      |                   |Dp[32]-
      |                       |Px[6]-
      |                       |Nx[26]-
      |                           |Rx[7]-
      |Qx[49]-
          |Vx[1]-
End of Tree Diagram
```