# Genesis 1:1-3 in Graphs: Extracting Conceptual Structures from Biblical Hebrew

Ulrik Petersen
Aalborg University
Department of Communication and Psychology
Kroghstræde 3
DK – 9220 Aalborg East
Denmark

ulrikp@hum.aau.dk
http://ulrikp.org/MA[1]

**Citation:** Petersen, Ulrik, Genesis 1:1-3 in Graphs: Extracting Conceptual Structures from Biblical Hebrew, SEE-J *Hiphil* 4 [http://www.see-j.net/hiphil] (2007), Accessed DD.MM.YY

**Abstract**

Automatically transforming text to conceptual graphs has long been a goal of the Conceptual Graphs community, starting with Sowa and Way's seminal paper in 1986. We have developed a method for transforming Old Testament texts in Hebrew into English-based conceptual graphs, and in this paper, we report on our method and its results. The method utilizes the text itself, a syntactic analysis of the text, an ontology of the text, plus some transformational rules. The end result is CGs with "shallow semantics" which can be deemed by a human to represent faithfully, if somewhat sterilely, a possible meaning of the original text. We argue that the method is general enough that it could be applied to texts in other languages.

## 1 Introduction

One application of Conceptual Graphs which has pervaded CG research since the very beginning is that of transformation of natural language text into a knowledge base using CGs. Such a knowledge base could have many uses: semantic searches (Nicolas, Moulin and Mineau 2003), question-answering (Velardi, Pazienza and De Giovanetti 1988), narratological reasoning (Schärfe and Øhrstrøm 2000), and formation of the foundation of dialogue-based systems.

In this paper, we report on our own method for transforming natural language text into CGs. The method has been developed for Biblical Hebrew, but, as we argue later on, it is plausible that it would work for other languages as well. We have applied the method to a portion of the text of the Hebrew Bible, and have demonstrated that the method works for this text.

The method takes as input five classes of data: First, the Hebrew text itself. Second, a ready-made syntactic analysis of the text, which, together with the text, have been provided by the Werkgroep Informatica at the Vrije Universiteit Amsterdam under Prof. Dr. Eep Talstra (Talstra and Sikkel 2000). Collectively, the text plus its analysis are called "the WIVU database." Third, an ontology of the text is used. Fourth, a number of lexicons. And fifth, a set of rules for transforming the text to CGs.

Our method proceeds in four stages: First, the ontology is automatically constructed by matching a Hebrew-English dictionary with WordNet (Fellbaum 1998). Second, the WIVU syntax is transformed to more traditional syntax trees. This is necessary because: a) the syntactic analysis is not really a tree, and b) it has units which are far too large for the method to work. Exactly how the syntax is transformed is outside of the scope of this paper, but suffice it to say that the resulting trees have units with only a few immediate constituents, and resemble traditional phrase structure trees. Third, this

1 The present article is a summary of the author's MA thesis. See (Petersen 2004b) for more information.

transformed syntax is used as a guide to transforming the text into "intermediate CGs". These CGs have almost no syntax left, but are still "not quite good enough." Therefore, a fourth stage uses CG-based rules to transform the "intermediate graphs" to "semantic graphs." These semantic graphs have almost no traces of Hebrew syntax left, and represent a "good enough" possible translation of the Hebrew source text into English-based CGs.

The rest of the paper is laid out as follows. First, we give a literature review. Second, we describe our method for automatically constructing the ontology. Third, we describe our method for transforming the input text to CGs. Fourth, we present our results. Fifth, we discuss our method and its results. And finally we conclude the paper.

## 2 Literature review

As we see it, two methods are described in the literature for transforming text to conceptual graphs. One uses a lexicon of canonical graphs and joins them, guided by a syntax tree. This method could be dubbed "syntax-directed joining of canonical graphs". This method was pioneered by Sowa and Way in their seminal paper from 1986 (Sowa and Way 1986), but many others have followed in a similar vein (Sowa 1988, Velardi, Pazienza and De Giovanetti 1988, Fargues, Landau, Dugourd and Catach 1986). Here we have only mentioned some of the more interesting references; a lot of references have been left out due to space-restrictions. This method works roughly as follows: At word-level, a lexicon of canonical graphs is used to give meaning to the individual lexical items. These canonical graphs are then joined using a syntax-tree from a parser as the guiding factor in deciding in which order to perform the joins. This is basically the method of (Sowa and Way 1986, Sowa 1988), and is followed with only slight variations in the other works cited above.

The other method described in the literature could be dubbed "ontology-guided, syntax-driven, and rule-based joining and refinement of graphs". At the lowest linguistic level, namely that of words, this method uses an ontology rather than a lexicon of canonical graphs as the starting point for its generation of conceptual graphs. Like the method of Sowa and Way, however, it then uses a syntax tree to drive the composition of conceptual graphs above word-level. After the whole syntax-tree has been traversed, and one or more complete CGs have thus been generated, a set of rules are applied for refinement of the graphs. The main work in this vein seems to be Caroline Barrière's PhD thesis (Barrière 1997), while another is Nicolas, Mineau and Moulin's article from ICCS 2002 (Nicolas, Mineau and Moulin 2002), later brought to fruition in (Nicolas 2003, Nicolas, Moulin and Mineau 2003). Barrière's work mainly follows the process outlined above.

The work of Nicolas et al., however, takes a slightly different approach. In this work, the syntax-tree is at first transformed to "syntax-graphs", which are simply a CG-representation of the syntax tree which are full of syntax and have almost no semantics. Rules are then applied using standard CG-matching algorithms in order to transform these "syntax graphs" into "semantic graphs".

Our own work is mainly an adaptation of the process described in (Barrière 1997), with some ideas taken from (Nicolas, Mineau and Moulin 2002, Nicolas 2003), and some of our own ideas added in, in order to fit our particular problem and the particular challenges of the Hebrew language of the source text. In the following sections, we describe our method.

## 3 Ontology

Every CG knowledge base has to have an ontology (Sowa 2000a, p. 487). For our purposes, we could not find a suitable Hebrew ontology, and so we opted for constructing one ourselves. This

was done by matching a machine-readable Hebrew-English dictionary provided by the Werkgroep Informatica with WordNet (Fellbaum 1998).

WordNet contains nouns, verbs, adjectives, and adverbs. Nouns and verbs are organized into several hyperonymically ordered hierarchies, each with a unique beginner. Adjectives and adverbs are ordered by various relations, but not hyperonymically. Strictly speaking, it is not individual words which are related; instead, words are grouped into *synsets* (or "synonym sets", since the words in a synset are synonyms of each other), and the relations obtain between these synsets.

The result of our method is a datastructure laid out as a lattice of *entry clusters*. Inside of each entry cluster are zero or more *ontology entries*. An entry cluster corresponds to a WordNet synset. An ontology entry corresponds to a sense of a lexeme in the Hebrew-English dictionary.

Our method works as follows. For each lexeme in the source text, look it up in the dictionary. Construct an Abstract Syntax Tree (AST) from the definition. Traverse the AST in such as way that separate senses of a lexeme end up in different ontology entries. For each sense, use various heuristics to find the most specific synset in WordNet which matches the English glosses of the lexeme. Once such a synset is found, create an entry cluster for the synset, if not already in the ontology. Then add all of the supertypes of the synset as entry clusters if not already done. Finally, create an ontology entry for the sense of the lexeme, and add it to the original entry cluster.

Adjectives are placed underneath "attribute", whereas adverbs are placed underneath "manner". It is well known that adjectives are not always used as attributes, and that adverbs can have functions other than describing manner. However, these approximations served us well for our purposes.

Several other heuristics could be mentioned; here we will limit ourselves to only one. "Be X" verbs (such as "tov", mening "be good") end up in the ontology under "attribute" along with the adjectives, with a gloss of "X" ("good" for "tov"). This is utilized in the rules described below for transforming words to CGs.

When the method fails for whatever reason, we have simply emended the dictionary such that a suitable WordNet synset is arrived at. This has the advantage of not altering the method, only the input data.

Finally, the unique beginners from WordNet are placed into a top-level ontology which has been derived from (Sowa 1992) and (Martin 1995). It can be seen in Fig. 1.

## 4 Creating conceptual structures

### 4.1 Introduction

As mentioned in the Introduction, the method runs in four stages. First, an ontology is constructed for the lexemes in the text, using the method described in the previous section. Second, the WIVU syntax is transformed into more traditional syntax trees. The precise nature of this process is intricately bound up with the particulars of the WIVU database, and as such are not interesting enough to be pursued at length in this paper. Third, the resulting syntax trees are transformed to "intermediate graphs" using rules for how to treat each syntactic construction. Fourth, the resulting graphs are refined into "semantic graphs" using rule-driven transformations based on graph-matching algorithms. An overview of the process can be seen in Fig. .

In the following two subsections, we describe first the syntax-to-intermediate-CG process, then the intermediate-CG-to-semantic-CG process.

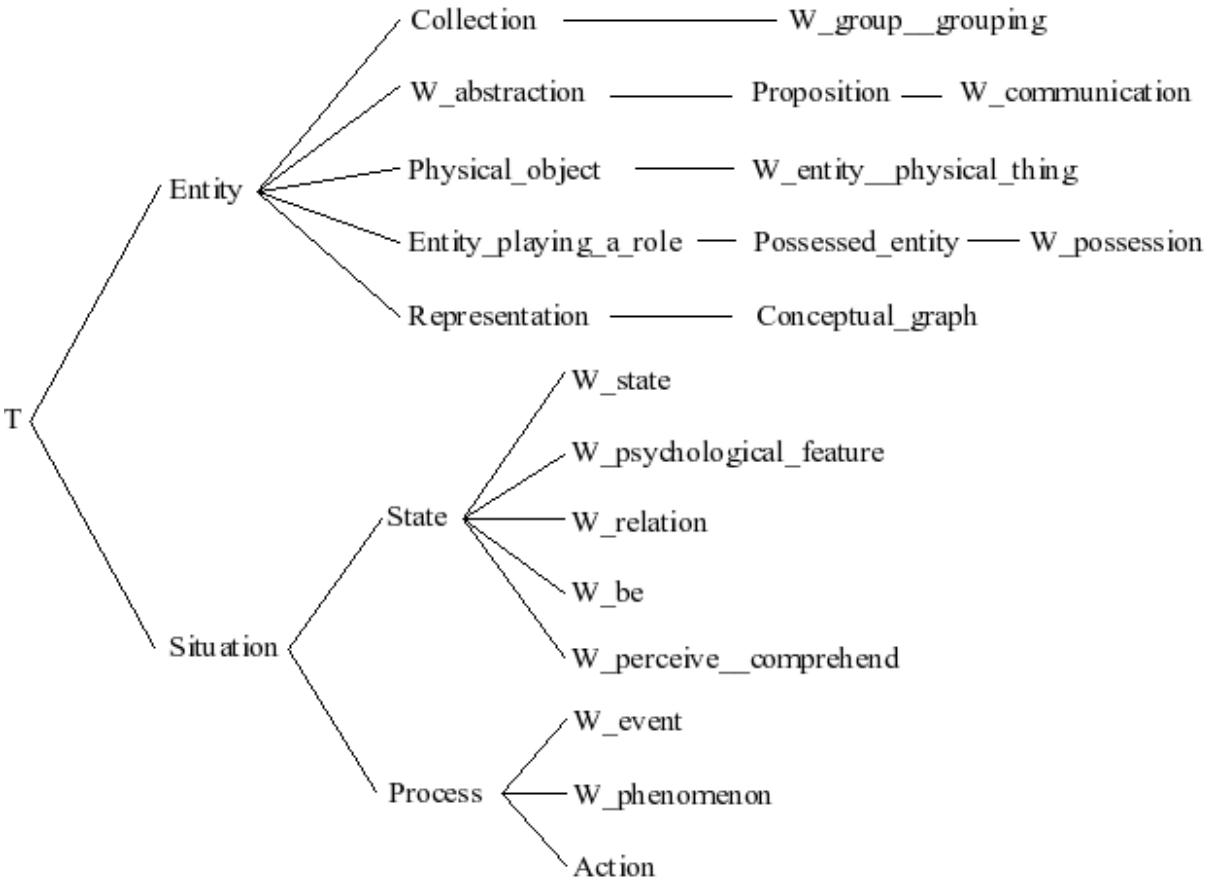


Figure 1: Our top-level ontology, derived from (Sowa 1992) and (Martin 1995). A "W_" prefix means that the concept type comes from WordNet.

### 4.2 From Syntax to Intermediate Graphs

For word-level and phrase-level, our method resembles that of (Barrière 1997), whereas for clause-level, we have taken cues from (Nicolas, Mineau and Moulin 2002) and added a few ideas of our own.

The input is a single clause, and the output is a list of CGs. Whenever there is ambiguity at some point in the transformation, the output list is duplicated as many times as the ambiguity necessitates, and the various possibilities are added in parallel.

The syntax tree is traversed in a depth-first manner. Thus the syntax tree is first traversed down to the bottom level, namely word-level, and is then traversed upwards again until the clause-node is encountered. For units up to and including the immediate phrasal constituents of the clause, the syntax-tree is used to decide the order in which to join the representations of the components. At clause-level, however, a different algorithm takes over, described below.

Each part of speech or phrase-type is used as input to a rule, which produces one or more (fragments of) CGs. In each CG, there is a specially privileged concept which is called the *attachment point*, and which is distinguished from all others by having a unary relation called "attach". This attachment point is the concept at which the CG is joined with other CGs at a

higher level.

For word-level, a number of rules are applied. For nouns, verbs, adjectives, and adverbs, the corresponding entry cluster or entry clusters are retrieved from the ontology, using the ontology entry or entries of the lexeme of the word as the mediating factor. Then various rules are applied, exemplified in Table . For example, plural nouns are simply translated to a concept with the type taken from the ontology, and the plural marker "{*}" added. Verbs are treated differently based on whether they are marked as "be X" verbs in the ontology or not. If not, the verb is treated simply as a concept with the right concept type. If it is a "be X" verb, it is treated as a "[state]" concept with an embedded CG which has the concept "[be]" attached to an "attr" relation again attached to a concept with the concept type of the verb. Recall that "be X" verbs end up under "attribute" in the ontology; hence this choice of relation is justified.

Prepositions and conjunctions become relations between concepts at higher levels. For these purposes, a small Hebrew-English lexicon is used which translates Hebrew prepositions and conjunctions to one or more English-language relations. For example, the Hebrew word "W:" (which is the connective conjunction) is translated to the relation "and". Likewise, the Hebrew preposition "<L" is translated to the relation "over".

For phrase-level, the input is a phrase structure rule (e.g., "NP --> noun"), and the output is one or more CGs. In order to know what phrase-structure rules might come up, and thus be able to write rules for how to treat them, we had to produce a grammar of the source text. This was not hard, in that the syntax-trees afforded easy reversal of the parsing process.

Examples of phrase-level rules are shown in Table , along with their resulting CG representations. When a concept of the resulting CG contains a right-hand-side part of the phrase structure rule in *italics*, it is understood that that concept is replaced by the CG which is the representation of that part. Moreover, it is really the "attachment point" concept of the underlying CG representation that replaces the concept, with all other concepts and relations of the underlying CG copied along, and any referent components in the resulting CG added to the referent of the attachment point. Finally, the "attach" relation from the underlying CG is removed, so as to leave only one "attach" relation in the resulting graph.

For example, the production "NP --> article noun" is represented by a concept whose concept type is that of the noun, but which gets the indexical "#" added to the referent of the noun-concept, in order to show that it is definite.

The "parallel construction" with the production rule "NP --> NP/PAR conjunction NP/par" deserves special mention. The "/PAR" and "/par" designations are *phrase-functions* assigned by the WIVU syntax. The former designates "first element of a parallel construction" whereas the latter designates "second element". First, the two NPs are transformed to CGs. Then the resulting CG representations are joined using the relation taken from the Hebrew-English lexicon translating prepositions and conjunctions to relations. Then, the resulting graph is embedded as a nested referent graph in a concept whose concept type is the minimum common supertype of the types of the two NPs. Finally, an "attach" relation is attached to the resulting concept. For example, the NP "[NP [NP/PAR Jacob] and [NP/par Esau]]"[2] would be transformed to the CG "[Person: [Person: Jacob]–and–>[Person: Esau]]".

2The [brackets] are here meant not as CG-brackets, but as linguistic brackets denoting phrase-boundaries.

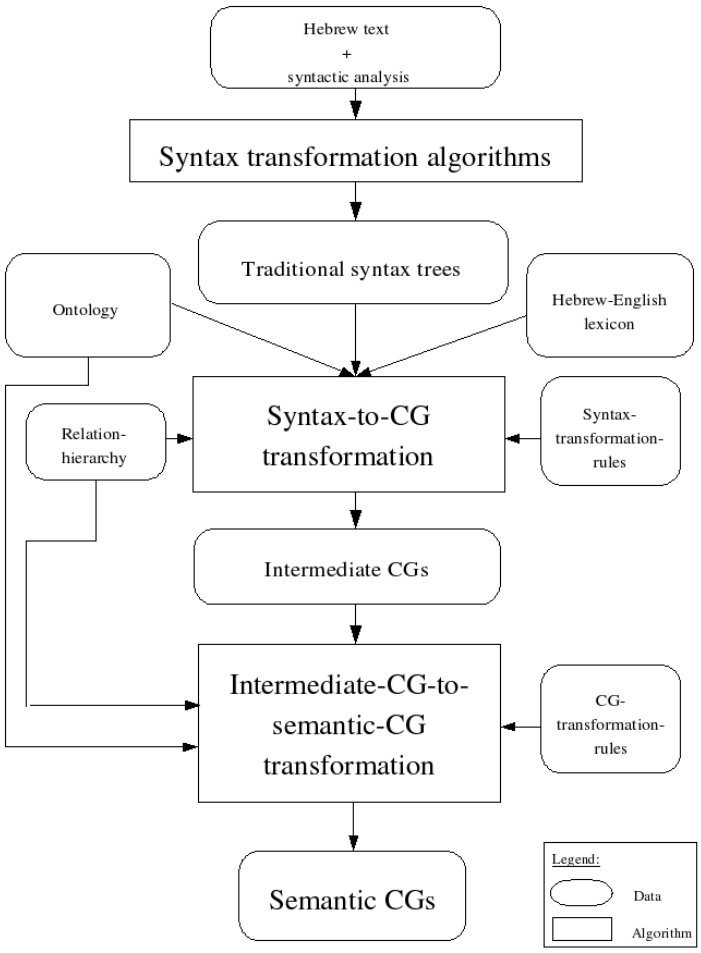


Figure 2: Overview of the process of the method.

Table 1: Examples of Word-level rules.

| input | output |
|---|---|
| noun, singular, no suffix | [*noun*]<–attach |
| noun, plural, no suffix | [*noun*: {*}]<–attach |
| verb, no suffix, is not “be_X” | [*verb*]<–attach |
| verb, no suffix, is “be_X” | attach–>[state: [be]–attr–>[*verb*]] |

Table 2: Examples of Phrase-level rules.

| input | output |
|---|---|
| NP --> noun | [*noun*]<–attach |
| NP --> article noun | [*noun*: #]<–attach |
| NP --> NP/PAR conjunction NP/par | [minComSuperType(NP/PAR,NP/par): [*NP/PAR*]–*conjunction*–> [*NP/par*]]<–attach |

This process continues, as mentioned before, up until the point where the immediate constituents of the clause have been converted the CGs. At that point, a different algorithm takes over. The immediate constituents of the clause are simply joined in a star, using the "most significant" constituent as the hub of the star, and using the "phrase functions" of the constituent phrases as the relations joining the representations of the other constituents to the representation of the "most significant" constituent. The "most significant" constituent is found by applying a "sliding scale" of importance for phrase-functions, and simply picking the one with the highest score. Generally, the following principle is used:

Predicates > Subjects > Objects > Complements > Adjuncts > All others

Predicates are always verbs of some sort in the WIVU database. Predicates are taken as central for two reasons. First, the verbal valency of a predicate verb in a sense determines what other phrases can cooccur with the verb, apart from peripheral elements such as adjuncts. Second, the predicate verb is central because many of the other constituents often incur relations with it. That is the case for subjects, objects, and complements. It should be noted, however, that in some linguistic theories, such as Role and Reference Grammar (Van Valin and LaPolla 1997), constituents such as fronted elements and time-references do not incur relations with the verb, nor would any linguistic theories claim that adjuncts incur relations with the verb, except when the adjuncts are PPs.

Finally, after the clause has been processed, the resulting graph or graphs are each encapsulated in a concept which either has type "proposition" or has type "Situation". The choice of type depends on the type of the clause as given by the WIVU database: If the clause type is "quotation", the type "proposition" is chosen; otherwise (e.g., for narrative clauses), the type "Situation" is chosen.

The process described above produces CGs which are "quite good" but not quite "good enough", in that they have traces of the Hebrew syntax left, especially at clause-level. The next subsection describes how we deal with this problem.

### 4.3 From Intermediate Graphs to Semantic Graphs

The final step of the method aims at removing the last traces of Hebrew syntax, thus yielding fully semantic CGs.[3] The input to the last step is the list of intermediate CGs produced in the previous step, and the output is again a list of CGs, hopefully with only one CG left.

```
[Rule:                                  Sample input:
   [Universal*a][Universal*b]           [Situation: [light*a]
   [Premise: [be_1?a: Pred]               [be_1*b: Pred](Subj?b?a)
       [Entity?b](Subj?a?b)]            ]
   [Conclusion: [Univeral?a]            Sample output:
       [Universal?b](Stat?b?a)]         [Situation: [be_1*a][light*b]
   ]]                                         (stat?b?a)]
```

Figure 3: Example of a rule along with sample input and output.

For refinement of CGs, both Nicolas et al. as well as Barrière use rules structured as a premise plus a conclusion. Whenever a premise matches a part of a CG, the corresponding matched

3As we argue later on, however, it is only "shallow semantics".

parts are replaced by the conclusion. After having applied all applicable rules, however, Barrière then copies whatever didn't get matched, whereas Nicolas et al. leave out whatever wasn't matched. Furthermore, Barrière keeps the premise and the conclusion separate, whereas Nicolas et al. gather them up into a single graph.

We follow the same course of action as Barrière with respect to copying whatever didn't get matched, but follow Nicolas et al. in their rule structure.

Our rules are made up of a [Rule] context whose referent is a conceptual graph. The nested CG contains three items: a) A premise (which is a context with type "Premise" and a nested CG expressing the premise), b) a conclusion (which has the same structure as the Premise, except its type is "Conclusion"), and c) A number of [Universal] concepts used as coreference links between the premise and the conclusion. This is precisely the structure employed by Nicolas et al.

An example rule can be seen in Fig. , along with sample input and output graphs.

### 4.4 Implementation

The method has been implemented in the Jython language[4] using the Notio library for providing CG operations (Southey and Linders 1999). The Emdros corpus query system is used for storage and retrieval of the WIVU database (Petersen 2004a, 2005, 2006a, 2006b)[5] . The sourcecode implementing the method can be downloaded from the URL mentioned at the front of the paper underneath the address.

```
In the beginning, God created the heavens and the earth.
[Situation:
  [God]<-agnt<-[create]-
    ->thme->[entity: [heavens: {*} #]->and->[earth: #]],
    ->in->[beginning]]

The earth was formless and void,       darkness was over the surface
                                       of the ocean,
[Situation:                            [Situation:
  [Universal:                            [darkness]->stat->[be_1]-
     [emptiness]->and->[void]               ->over->[surface_1: {*}]-
  ]<-stat-[earth: #]]                                  <-poss<-[ocean]]

and the Spirit of God was hovering over the waters.
[Situation:
  [hover]-
    ->agnt->[spirit]<-poss<-[God],
    ->over->[surface_1: {*}]<-poss<-[water: {*} #]]

And God said:          ``Let there be light''    And there was light.
[Situation:            [proposition:             [Situation:
  [say]->agnt->[God]     [be_1]<-stat<-[light]     [be_1]<-stat<-[light]
]                      ]                         ]
```

Figure 4: The CGs resulting from applying the method to Genesis 1:1-3.

4See http://www.jython.org/

5See http://emdros.org/

## 5 Results

The method just described has been applied to the first three verses of the first chapter of the book of Genesis in the Hebrew Bible. The verses, along with the resulting graphs, can be seen in Fig. 4.

## 6 Discussion

### 6.1 Compositionality

One of the foundational assumptions underlying the design of our method is that syntax and semantics are intricately interwoven.

The precise nature of this interwovenness is elusive, but we believe that they are interwoven in such a way that syntax, to an extent that is perhaps larger than what is usually assumed in some parts of the landscape of linguistic theories, contributes decisively towards the meaning of a given sentence. Another way of making the point is to say that we believe that syntax, at least for some languages, plays a large role in determining what a sentence means. This notion is not novel; rather, it has been assumed in Artificial Intelligence since the very beginning of the field, and is also acknowledged in many linguistic theories such as Role and Reference Grammar (Van Valin and LaPolla 1997).

The particular way in which syntax plays a role in determining the meaning of a sentence may vary from language-family to language-family. For a large class of languages, Hebrew included, it appears that syntax has a bearing on semantics because the semantics of a sentence in that language obeys the compositionality principle, and because syntax appears to be the factor guiding the order in which the semantics is is composed.

Fargues et al., make the point beautifully (Fargues, Landau, Dugourd and Catach 1986, p. 73):

> "A classical property of the formal models for natural language semantics used in AI is that they obey the compositionality principle. It is usually assumed that a representation of the semantics of an entire sentence can be built by combining the semantic representations associated with its components." (p. 73)

In our work, it has been found that Hebrew syntax exhibits the qualities which enable the compositionality principle to work for Hebrew, precisely when guided by Hebrew syntax. That is, the semantic representation of the lowest phrasal units can be composed from the semantic representation of their constituent words; the semantic representation of the higher phrasal units can be composed from the representation of their constituent words and/or phrases; and the semantic representation of clausal units can be composed from the representation of their constituent phrases. This is because the phrasal units expressed in Hebrew have strong internal coherence, and are not easily split apart by intervening material (van der Merwe, Naudé and Kroeze 1999).

### 6.2 Shallow or surface semantics

Velardi et al. (Velardi, Pazienza and De Giovanetti 1988, p. 252) draw a distinction between "deep" and "shallow" semantics, and affirm that their work falls within the latter category:

> "We believe that the ultimate goal of a language-understanding system is to produce a "deep" representation, but the methods by which this representation should be derived are unclear and not generally accepted in the present state of the art." (p. 252)

Our work falls squarely within the "shallow semantics/surface semantics" camp as well. For any given CG that is produced by our method, the "meaning" should be able to be extracted by means of three things: a) The ontology, giving meaning to the conceptual types, b) The relation hierarchy, giving meaning to the relations, and c) The syntax of the CG language (CGs are bipartite graphs; the direction of arcs determine the arguments of a relation; etc.) – the syntax of CGs helps us relate the semantics of the concept types with the semantics of the relation hierarchy to form a coherent whole.

However, if we look closely at each of these steps, it surfaces that the semantics are very shallow: First, our ontology has no canonical graphs underneath each entry, but merely a reference to WordNet's dictionary definition. Second, our relation hierarchy refers to external dictionaries. Third, the preceding two points imply that what we are really dealing with is not semantics, but a collection of *symbols* connected by a certain syntax.

This goes right back to the meaning triangle of Ogden and Richards, mentioned in (Sowa 2000b). The meaning triangle relates an object (such as a cat) with the symbol which stands for the object (such as "Garfield"), and these two in turn are related by the triangle to the "meaning" of the object, or the concept or neural excitation which appears in the mind of a person thinking about Garfield the cat.

Sowa calls this last neural excitation "elusive" (p. 60), and with good reason. Hoffmeyer (Hoffmeyer 1996) writes:

> "All computer programs are completely based on Peircean "secondness", i.e. syntactic operations, since applications of the rules governing the manipulation of the symbols does not depend upon what the symbols "mean" (their external semantics), only upon the symbol type. The problem is not only that the semantic dimension of the mental cannot be reduced to pure syntactics. ... The problem rather is that the semantic level itself is bound up in the unpredictable and creative power of the intentional, goal-oriented, embodied mind."

Thus Hoffmeyer would argue that it takes an "intentional, goal-oriented, embodied mind" to produce semantics; and since intentions are inherently based on Peircean Thirdness, and since all present-day computer programs are completely based on Peircean "Secondness", it follows that present-day computer programs cannot attain to "deep semantics" – all they can do is to manipulate symbols syntactically.

Thus all present-day computer programs must rely on syntactic operations on symbols, and cannot access any external semantics.[6] Applying this to the results of my method, we see that the semantics of the resulting CGs are by definition limited to "surface semantics" rather than "deep semantics". This is because the "deep semantics" could only be attained by accessing "external semantics", which cannot be codified in the computer except through more symbols which indexically point to that "external semantics". We are not arguing that symbols cannot account for semantics, nor that semantic concepts cannot be reduced to symbols; we are merely arguing that *if* it is possible, the *present state of the art* cannot produce the complex of symbols which would presumably be required to produce "deep semantics".

### 6.3 The non-centrality of Hebrew

How central to the working of our method is the choice of input language? And how central is the particulars of the WIVU syntax? Could the method be applied to other languages and other kinds of syntactic analysis?

6This is related to Searle's Chinese Room argument. We are not claiming that Searle is right; merely that the present state of the art has not proved him wrong.

The answer to these questions depends on the amount of modification that one is willing to accept to the method. Let us summarize the main points at which the method is dependent on Hebrew and the particulars of the WIVU database.

First, the ontology is Hebrew-based. However, any other ontology could be "plugged in" without much difficulty, as there is little Hebrew-specific about it. All that is required is that the ontology be able to relate lexemes of the input language to concept types.

Second, Hebrew plays a role in the Syntax-to-CG rules. Yet again, here Hebrew can be seen to be immaterial to the method itself, insofar as the rules could be rewritten for a different parsing strategy without changing the algorithm driving the transformation. All that is required is that we can account for all production rules. This can be achieved either by reverse-engineering a phrase-structure grammar from existing analyses (as we have done), or by simply taking the phrase-structure rules from the grammar that produced the synatctic analysis.

Third, the WIVU syntax plays a role in what the method does with immediate constituents of clauses. Recall that at that juncture, we produce a "star" with the most important phrase as the hub, and the other phrases related to the hub through their "phrase functions". Admittedly, phrase-functions are a particularity of the WIVU database, but they are also not that uncommon in syntactic analyses. Thus all that is required for the method to work is that the syntactic analyses at the level of the highest phrasal units exhibit some form of "phrase functions", and since this is not all that uncommon (e.g., the TIGER corpus (Brants, Skut and Uszkoreit 1999)), we can assume that this is not an onerous requirement.

Fourth, in the rules for transforming intermediate graphs to semantic graphs, the WIVU syntax plays a role via the phrase functions. However, the algorithm is so general that it could be applied to different rules with no changes to the algorithm of the method. Thus, for a different language with a different syntactic analysis, only the rules would have to be exchanged for others tailored to that language.

Thus it appears that our method is general enough that it could be applied to other languages than Hebrew.

## 7 Conclusion

We have presented a method for transforming Hebrew text to conceptual graphs, and have presented some results of the method. The method could aptly be dubbed "ontology-guided, syntax-driven, and rule-based joining and refinement of graphs", and is based on the methods described in (Barrière 1997, Nicolas, Mineau and Moulin 2002, Nicolas 2003).

We have argued that the semantics of the Hebrew language obeys the "compositionality principle", guided by syntax. We have argued that what the method produces is "shallow semantics" rather than "deep semantics". And finally, we have argued that the method could be applied to other languages with little modification to the algorithms.

Numerous points of critique could be advanced against our work. Here let us mention a few. First, the ontology should ideally be extended to include canonical graphs and schemas for each entry. This is so as to be able to put "more semantics" into the ontology. Second, the stage which transforms "intermediate graphs" to "semantic graphs" could be made better, e.g., by rejecting CGs for which no rules matched. As it is, if no rules matched, the input CG is copied verbatim without flagging a warning. Third, a lot of Hebrew language features are left unprocessed such as aspect, mood, verbal stems (binyanim), and inter-clausal relationships. These could all be taken into account.

Opportunities for further research abound. For example, the input data to the method could be extended to cover the full Hebrew Bible, including the Aramaic portions. Second, applications

could be made from the results of such a full-scale translation of the Hebrew Bible into conceptual graphs. Applications could include automatic Bible Translation and narratological analysis. Third, our method could be improved in various ways. For example: Anaphora resolution could be added, inter-clausal relations could be considered, along with aspect and mood, elliptic clauses, and clauses which play a role in other clauses. Finally, metaphors could profitably be handled in a better way.

**References**

Barrière, C.: 1997, *From a Children's First Dictionary to a Lexical Knowledge Base of Conceptual Graphs*, PhD thesis, Université Simon Fraser.

Brants, T., Skut, W. and Uszkoreit, H.: 1999, Syntactic annotation of a German newspaper corpus, *Proceedings of the ATALA Treebank Workshop*, Paris, France, pp. 69–76.

Fargues, J., Landau, M.-C., Dugourd, A. and Catach, L.: 1986, Conceptual graphs for semantics and knowledge processing, *IBM Journal of Research and Development* **30**(1), 70–79.

Fellbaum, C. (ed.): 1998, *WordNet: An Electronic Lexical Database*, MIT Press, London, England and Cambridge, Massachusetts.

Hoffmeyer, J.: 1996, Evolutionary intentionality, *in* E. Pessa, A. Montesanto and M. Penna (eds), *Proceedings from The Third European Conference on Systems Science, Rome 1.-4. Oct. 1996*, Edzioni Kappa, Rome, pp. 699–703.

Martin, P.: 1995, Using the WordNet concept catalog and a relation hierarchy for knowledge acquisition, *in* P. Eklund (ed.), *Proceedings of the Fourth International Workshop on PEIRCE*, Santa Cruz, USA, pp. 36–47.

Nicolas, S.: 2003, *Sesei: une filtre sémantique pour les moteurs de recherche conventionnels par comparaison de structures de connaisance extraites depuis des textes en langue naturel*, Master's thesis, Université Laval.

Nicolas, S., Mineau, G. W. and Moulin, B.: 2002, Extracting conceptual structures from English texts using a lexical ontology and a grammatical parser, *in* G. Angelova, D. Corbett and U. Priss (eds), *Supplemental proceedings of ICCS 2002*, pp. 15–28.

Nicolas, S., Moulin, B. and Mineau, G. W.: 2003, Sesei: A CG-based filter for internet search engines, *in* A. d. Moor, W. Lex and B. Ganter (eds), *Proceedings of ICCS 2003*, Vol. 2746 of *LNAI*, Springer Verlag, Berlin, pp. 362–377.

Petersen, U.: 2004a, Emdros — a text database engine for analyzed or annotated text, *Proceedings of COLING 2004*, pp. 1190–1193. http://emdros.org/petersen-emdros-COLING-2004.pdf.

Petersen, U.: 2004b, *Creation in Graphs: Extracting Conceptual Structures from Old Testament Texts*. MA thesis, University of Aalborg, Department of Communication. Published in: *Impact* -- an Electronic Journal on Formalisation in Media, Text and Language -- Impact Theses, http://www.impact.aau.dk/theses.html and http://ulrikp.org/MA/

Petersen, U.: 2005, *Evaluating corpus query systems on functionality and speed: TIGERSearch and Emdros*. In: Angelova, G., Bontcheva, K., Mitkov, R., Nicolov, N. and Nikolov, N. (eds): International Conference Recent Advances in Natural Language Processing 2005, Proceedings, Borovets, Bulgaria, 21-23 September 2005, pp. 387--391. http://www.hum.aau.dk/~ulrikp/pdf/Tiger-Emdros-2005-08-19-final.pdf

Petersen, U.: 2006a, *Querying both Parallel and Treebank Corpora: Evaluation of a Corpus Query System*. In: European Language Resource Association (ELRA) Proceedings

of LREC 2006, Language Resources and Evaluation Conference, Genoa, Italy, May 2006. http://www.hum.aau.dk/~ulrikp/pdf/LREC2006.pdf

Petersen, U.: 2006b, *Principles, Implementation Strategies, and Evaluation of a Corpus Query System*. In: Yli-Jyrä, Anssi; Karttunen, Lauri; Karhumäki, Juhani (eds), *Finite-State Methods and Natural Language Processing 5th International Workshop, FSMNLP 2005, Helsinki, Finland, September 1-2, 2005, Revised Papers*, Lecture Notes in Computer Science, Volume 4002/2006, Springer Verlag, Heidelberg, New York, pp. 215-226. DOI: 10.1007/11780885_21. http://dx.doi.org/10.1007/11780885_21

Schärfe, H. and Øhrstrøm, P.: 2000, Computer aided narrative analysis using conceptual graphs, *in* G. Stumme (ed.), *Working with Conceptual Structures: Contributions to ICCS 2000*, Shaker Verlag, Aachen, pp. 16–29.

Southey, F. and Linders, J. G.: 1999, Notio - a Java API for developing CG tools, *in* W. Tepfenhart and W. Cyre (eds), *Proceedings of ICCS 1999*, Vol. 1640 of *LNAI*, Springer Verlag, Berlin, pp. 262–271.

Sowa, J. F.: 1988, Using a lexicon of canonical graphs in a semantic interpreter, *in* M. Evens (ed.), *Relational Models of the Lexicon*, Cambridge University Press, Cambridge, UK, pp. 113–137.

Sowa, J. F.: 1992, Conceptual graphs summary, *in* T. E. Nagle, J. A. Nagle, L. L. Gerholz and P. W. Eklund (eds), *Conceptual Structures: Current Research and Practice*, Ellis Horwood, New York, pp. 3–51.

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

Sowa, J. F.: 2000b, Ontology, metadata, and semiotics, *in* B. Ganter and G. W. Mineau (eds), *Proceedings of ICCS 2000*, Vol. 1867 of *LNAI*, Springer Verlag, Berlin, pp. 55–81.

Sowa, J. F. and Way, E. C.: 1986, Implementing a semantic interpreter using conceptual graphs, *IBM Journal of Research and Development* **30**(1), 57–69.

Talstra, E. and Sikkel, C.: 2000, Genese und Kategorienentwicklung der WIVU-Datenbank, *in* C. Hardmeier, W.-D. Syring, J. D. Range and E. Talstra (eds), *Ad Fontes! Quellen erfassen - lesen - deuten. Was ist Computerphilologie?* , VU University Press, Amsterdam, pp. 33–68.

van der Merwe, C., Naudé, J. A. and Kroeze, J. H.: 1999, *A Biblical Hebrew Reference Grammar*, Sheffield Academic Press, Sheffield.

Van Valin, Jr., R. D. and LaPolla, R. J.: 1997, *Syntax – Structure, meaning, and function*, Cambridge University Press, Cambridge, U.K.

Velardi, P., Pazienza, M. T. and De Giovanetti, M.: 1988, Conceptual graphs for the analysis and generation of sentences, *IBM Journal of Research and Development* **32**(2), 251–267.