Automatic Lexicon-based Ontology-creation

A methodological study

Ulrik Petersen

January 6, 2003

Vejleder: Peter Øhrstrøm
Studium: Humanistisk datalogi
9. Semester
Aalborg Universitet

Abstract

Denne rapport behandler metodeovervejelser i forbindelse med automatisk ontologibygning ved hjælp af ordbøger. Ontologi er studiet af de kategorier, der findes i et givent domæne. En ontologi er produktet af et sådant studie. I denne rapport udvikler vi metoder til at tage en koncis hebraisk-engelsk ordbog og matche den med WordNet. WordNet kan ses som værende en ontologi over store dele af det Engelske vokabular. Ud af denne proces kommer en ontologi over de kategorier, som findes i den hebraiske ordbog.

I et baggrundsafsnit ser vi på den historiske udvikling af feltet ontologi siden Heraklit. Vi ser også på vores hebraisk-engelsk-ordbog, og på, i hvilken kontekst dette projekt laves, nemlig i kontekst af mine specialestudier. Slutteligen ser vi på, hvilken hebraisk tekst, der ligger til grund for den delmængde af hebraisk-ordbogen, som vi vil interessere os for.

I et afsnit om teori ser vi på det teoretiske fundament for vore anstrengelser. Et længere afsnit behandler formel ontologi, hvori vi ser på koncepter som ekstension og intension, ontotyper og individer, ontologiske relationer, gitre, med mere. Vi ser også på ontologiers top-kategorier og deres betydning for vores arbejde. Endvidere kigger vi nærmere på den valgte hebraiske ordbog, og afslutter med et nærmere kig på WordNet.

Hoveddelen af rapporten er et metodeafsnit, i hvilket vi udvikler og begrunder vores metode. Vi udvikler og anvender begreber som “ontology entry” og “entry cluster,” hvilket er centrale begreber i vores metode. Vi overvejer, hvad der bør være med i ontologien, og hvad der kan udelukkes. I et længere afsnit behandler vi de algoritmer, som er udviklet til at løse problemet. Selve algoritmerne kan beses i Appendix B, mens selve rapporten indeholder de metodiske overvejelser bag algoritmerne. Vi behandler også spørgsmålet om, hvad der kan gøres, når algoritmen fejler, samt hvorfor den fejler. Til slut giver vi lidt statistik, hvilket giver empirisk vægt bag påstanden, at vores metode virker.

Rapporten afsluttes med en konklusion og et udblik mod videre studier.[1]

[1] Dette resumé er ikke på engelsk som resten af rapporten, men på dansk, fordi det var et krav fra studienævnets side, hvis der skulle opnås tilladelse til at skrive på engelsk, at der forelå et resumé på dansk.

Contents

1 Introduction

This report is about methods for automatic ontology-creation based on dictionaries. The basic idea is to explore how one can go about building an ontology of Genesis 1-3 in the Old Testament by marrying a concise lexicon of Hebrew-English with WordNet (Fellbaum (1998b)). The end result is an ontology of chapters 1-3 of Genesis, cross-indexed to the Hebrew-English lexicon.

This research is a necessary preliminary step towards the goal which I will attempt to fulfill in my Master's Thesis, namely building a system for automatically translating a syntactic analysis of Genesis 1-3 in the Old Testament in Hebrew to the conceptual graphs of John Sowa (Sowa (2000)).

2 Background

2.1 Introduction

This section gives background information related to the project. The first section gives a historical overview of the study of ontology, while the second section discusses the Hebrew lexicon chosen as a basis for the matching. The third section discusses the intentionality behind the finished ontology, i.e., what it is to be used for, and why. The fourth section discusses the text whose lexemes we are treating, since are not attempting to deal with the full mass of Hebrew lexemes, but rather a subset belonging to a specific text. The final section discusses other related work.

2.2 Historical background

2.2.1 Introduction

In this section, we give a historical overview of the development of the field of ontology, from Heraclitus to Sowa. We shall do so based on a number of important names who have contributed to the field. We shall discuss the contributions of Heraclitus, Plato, Aristotle, Porphyry, Ramon Lull, Leibniz, Kant, Peirce, Whitehead, and Sowa. These names are by no means an exhaustive list of those who have contributed to the field, but rather a selection based in part on importance to the field in general, in part on importance for our purposes.

2.2.2 Heraclitus

The Greek philosopher Heraclitus, who lived in the sixth century B.C., invented one of the most basic distinctions in ontology which has influenced ontological designs ever since his day. This distinction is that between the physical and the abstract, although he did not originate these terms. Instead, Heraclitus maintained that "all" physical things "flow" (Greek πάντα ῥεί), that is, everything is in a state of flux. This was summed up in his famous statement, "you cannot step into the same river twice." But he also maintained that "all things come into being according to the *logos*." The intangible logos (Greek λόγος), which can be translated by such diverse notions as "word," "speech," "reason," "account," or "reckoning," came to underlie the later notion of "abstract." See Sowa (2000, pp. 55–56).

2.2.3 Plato

Plato, who lived a century after Heraclitus, adopted the latter's distinction, but developed a theory involving the *logos* in which the world was divided into two spheres: the world of *forms* or *ideas* on the one hand and the *physical world* on the other. The physical world, for Plato, was a mere shadow of the world of ideas. To him, the world of ideas was fundamental and most "real," whereas the physical world was merely illusory. See Sowa (2000, p. 56).

2.2.4 Aristotle

Plato also developed the subject of *epistemology* - the study of the nature of knowledge. Aristotle, who was Plato's student, shifted the focus of philosophy from the *nature* of knowledge to *how to represent*

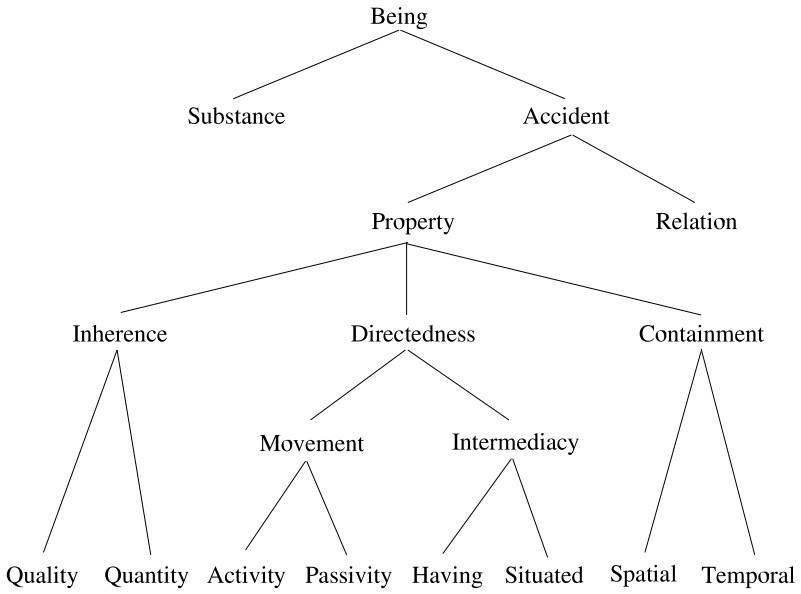

Figure 1: Aristotle's categories, from Brentano (1975[1862]).

knowledge. In doing so, he invented or at least established much of the technical vocabulary that is still used internationally in talking about the subjects that he described - vocabulary such as *category*, *metaphor*, *hypothesis*, *quantity*, *quality*, *genus*, *species*, *subject*, and *predicate*.

Among Aristotle's many achievements can be found the invention of formalized patterns of reasoning, called *syllogisms*. They are mentioned here because two of Aristotle's syllogisms are especially useful when dealing with the ontological property of inheritance between types, and inheritance between types and instances. The syllogism later called *Barbara* by the medieval scholastics supports inheritance between types, whereas the syllogism which the scholastics later named *Darii* supports inheritance between a type and its instances.

Aristotle invented the method of definition of categories now known as *definition by genus and differentiae*. The method involves taking a *genus* or supertype, e.g., **Mammal**, then defining a new ontotype, e.g., **Elephant** by saying, e.g., "an **Elephant** is a **Mammal** which has four legs, a trunk, and large ears." The differentiae show how the subtype differs from the supertype.

Aristotle also invented ten categories for talking about the world. The categories are: Substance, Quality, Quantity, Relation, Activity, Passivity, Having, Situatedness, Spatiality, and Temporality. The categories have later been categorized into a type-hierarchy by Brentano (1975[1862]), using other categories from Aristotle's work: Being, Accident, Property, Inherence, Directedness, Containment, Movement, and Intermediacy. They can be seen in Figure 1 in Brentano's categorization, taken from Sowa (2000, p. 57).

For sources of information in this section, see Sowa (2000, pp. 1–4, 56–57).

2.2.5 Porphyry

Porphyry, who lived in the 3rd century AD, wrote a commentary on Aristotle, "On Aristotle's Categories," see Porphyry (1992). In this commentary, Porphyry showed how some of Aristotle's categories could be arranged in a tree, ordered by genus, subtype, and differentiae. This was the first time anyone had arranged categories in a tree-like structure, a practice which is common with today's type-lattices. See Sowa (2000, p. 4).

QUANTITY	QUALITY	RELATION	MODALITY
Unity	Reality	Inherence	Possibility
Plurality	Negation	Causality	Existence
Totality	Limitation	Community	Necessity

Table 1: Kant's categories

2.2.6 Ramon Lull

The medieval philosopher and missionary Ramon Lull (Thirteenth Century AD) was the first to develop devices for automated reasoning. In his magnum opus, "Ars Magna," he invented devices using concentric disks with sectors that could align and form combinations of letters. The letters stood for various categories, some of which were Aristotle's categories. By rotating the discs systematically, all possible combinations could be generated. Lull's achievement is important in our context because it provided the inspiration for Leibniz's work on automated ontological reasoning some centuries later. See Sowa (2000, p. 4–6), Künzel and Bexte (1993, pp. 15–49).

2.2.7 Leibniz

Gottfried Wilhelm Leibniz, who lived in the Seventeenth Century, made many important discoveries in mathematics, logic, and philosophy. He invented the differential calculus independently of Newton, and discovered the predecessor to what today has become the many kinds of modal logic. He also invented a mechanical device for multiplication and division, an achievement that helped him in his endeavors to put Lull's combinatorial work on a more firm mathematical basis. Leibniz assigned prime numbers to each of Aristotle's categories, and stipulated that subtypes be formed by multiplication of these primes. Thus every category would have its own unique number, generated as a product of the numbers of its immediate supertypes. Because of the fundamental theorem of algebra, that every integer is a unique product of primes, and because Leibniz had assigned each of the fundamental categories a unique prime, all categories had their own, unique number. Inheritance was effectuated by multiplication, and could be checked by division. See Sowa (2000, p. 6–7), Künzel and Bexte (1993, pp. 135–154, 166–175).

2.2.8 Kant

Immanuel Kant (Eighteenth Century) was the first to challenge Aristotle's categories. In the "Critique of Pure Reason," Kant (1929[1781,1787]) organized his categories as in Table 1. The interesting thing to note is that Kant organized the categories in four sets of three. This was not a coincidence or for aesthetical purposes, but rather had a fundamental methodological purpose. Kant explained that the third in each triad "always arises from a combination (Verbindung) of the second category with the first." (1787:110, cited in Sowa (2000, p. 58)). This was an important insight, and its ramifications for the study of ontology in generations after Kant have been far-reaching. See Sowa (2000, p. 57–58).

2.2.9 Peirce

The great American philosopher Charles Sanders Peirce (1839-1914) contributed to numerous fields, but among his most basic achievements, which can be seen as influencing most of his contributions, stands the development of his categories of Firstness, Secondness, and Thirdness.

Peirce (1891) writes (cited in Sowa (2000, p. 60)):

> "First is the conception of being or existing independent of anything else. Second is the conception of being relative to, the conception of reacting with, something else. Third is the conception of mediation, whereby a first and a second are brought into relation."

Peirce's categories have had an enormous influence in a number of fields, including semiotics and ontology. Especially his methodological categories of Firstness, Secondness, and Thirdness have had a large influence, both in ontology and in semiotics. See Sowa (2000, p. 60–62).

2.2.10 Whitehead

Alfred North Whitehead is probably most famous for his collaboration with Bertrand Russell on the "Principia Mathematica" (Whitehead and Russell (1925)) which became the foundation for much of the work done in mathematics in the Twentieth Century. However, after working in mathematics, Whitehead turned to philosophy, and developed an ontology in the book "Process and Reality: An Essay in Cosmology" (Whitehead (1929)). In this book, Whitehead agreed with Heraclitus and Plato on the distinction between the physical and the abstract, but endorsed Aristotle's correction of Plato that the physical is what is real, not the forms.

Whitehead proposed eight categories, six of which constitute two Peircean triads, the remaining two being principles for generating more categories. On the Physical side of the Heraclitan dichotomy, Whitehead placed "actual entity" for Firstness, "prehension" for Secondness, and "nexus" for Thirdness. On the Abstract side of the Heraclitan dichotomy, Whitehead had "eternal objects" for Firstness, "propositions" for Secondness, and "subjective forms" for Thirdness. See Sowa (2000, pp. 63-66).

Whitehead also originated the distinction between *Continuant* and *Occurrent*, referring to the speed with which entities undergo change. A continuant is something that endures over a long period of time, whereas an occurrent is something that does not endure over time. This distinction is important in Sowa's top-level ontology. See Sowa (2000, p. 71).

2.2.11 Sowa

John F. Sowa has advocated a top-level ontology based on distinctions from Heraclitus, Peirce, and Whitehead. It can be seen in Figure 2. Sowa has expounded this top-level ontology in several places, including Sowa (2000, pp. 67–77) and Sowa (1995). The significance of the ontology is that it brings together three fundamental distinctions in ontology that have been developed in philosophy over a long period of time. As such, it stands as a viable tool for systematic thought about ontological categories.

2.2.12 Conclusion

We have looked at the history of the study of ontology since Heraclitus up until our present day. It turns out that some distinctions are fundamental when dealing with the nature of being. Among them are "physical" versus "abstract," from Heraclitus, Plato, and Aristotle. Another important distinction is Peirce's Firstness, Secondness, and Thirdness. Whitehead contributed the temporal distinction of "Continuant" and "Occurrent." Sowa pulled all three distinctions together into a top-level ontology that is very stringent and which is a viable tool for ontological thought.

2.3 Lexicon

The lexicon chosen is that produced by the Werkgroep Informatica of the Vrije Universiteit Amsterdam under Prof. Dr. Eep Talstra. It is a concise lexicon of Hebrew and Aramaic intended to be shipped with the commercial database "Quest 2" (Syring (1998)). The Quest 2 database, in turn, is based on the Hebrew database of the Old Testament produced by the Werkgroep Informatica, see Talstra and Sikkel (2000), Talstra (1989), Talstra and Postma (1989), Talstra and Van Wieringen (1992), Verheij (1994), and Hardmeier, Syring, Range and Talstra (2000). The reasons for choosing this lexicon include:

- It is already available in electronic form, thus there is no need to digitize anything, which can be a time-consuming and error-prone process,
- The lexemes in the lexicon match those used in the Werkgroep Informatica database, even down to the level of homographs. Thus there will be no need to do any additional matching between text-lexemes and lexicon-lexemes when building the graphs in the Master's Thesis work.

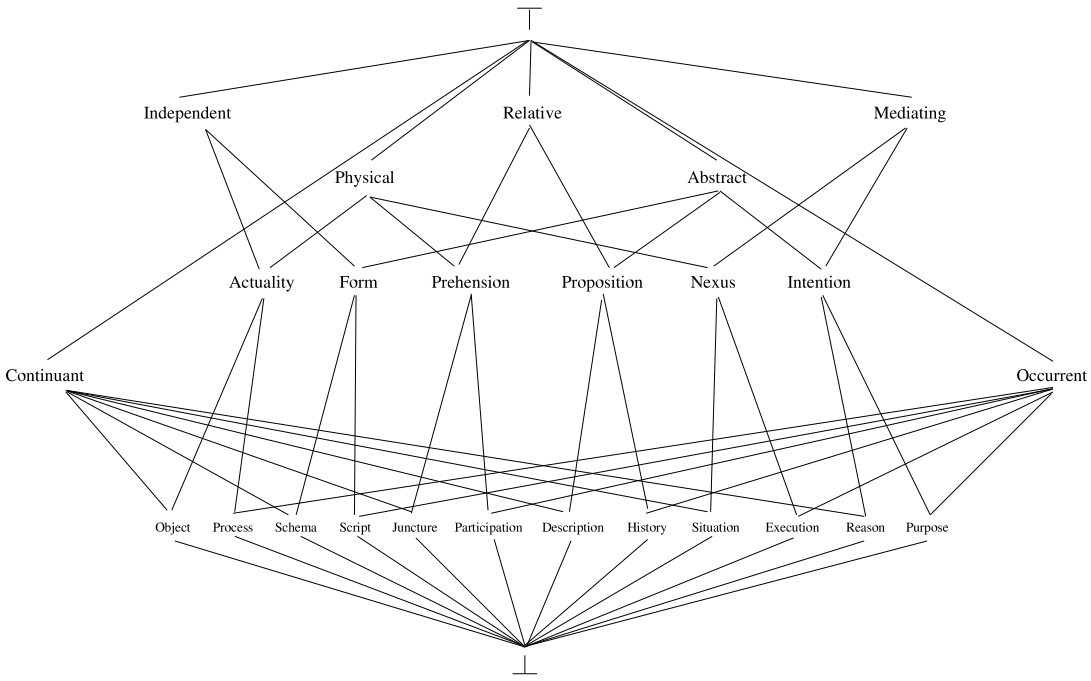

Figure 2: Sowa's top-level ontology

- It is concise, meaning that there are only a few glosses for each entry. This reduces the complexity of the matching between the lexicon and WordNet.

The lexicon was compiled by PhD-candidate Hendrik-Jan Bosman and Prof. Dr. Ferenç Postma.

2.4 Intentionality

Given that the work done in this report is to prepare the ground for the work to be done in my Master's Thesis, it is well to reflect on the intentionality of the automatically generated ontology. That is, what will it be used for, and why?

As already mentioned, in my Master's Thesis, I wish to explore methods for extracting meaning from Old Testament texts plus their syntactic analysis. The extracted meaning is to be represented in the form of conceptual graphs (Sowa (2000)). As stated in Nicolas, Mineau and Moulin (2002, p. 17), two main practices are described in the literature as regards extraction of conceptual structures from text:

1. The use of canonical conceptual graphs, or
2. The use of transformation rules.

The former method is represented by such works as Sowa and Way (1986), Sowa (1988), Hess and Cyre (1999). The latter method is represented by, e.g., Zhang and Yu (2001), Nicolas et al. (2002), Barrière (1997) and Tajarobi (1998).

I have already chosen the second method as a basis for my methodological explorations. The reason is that I do not believe it is feasible to generate an adequate set of canonical graphs for the entire Old Testament, and extracting graphs from the entire Old Testament is a long-term goal, even if it cannot be fulfilled in the course of the work for my Master's Thesis. Even an adequate set of canonical graphs for the limited vocabulary of the chosen text, Genesis 1-3, may prove intractable given the limited time available.

But in order for the process to work, one must still have an adequate ontology for giving the meaning and subtype placement in the type-hierarchy of each concept. WordNet promises to provide such an ontology.

Thus the intended use is as a type-hierarchy for extraction of meaning from text plus a syntactic analysis of this text. The reason we choose WordNet as a basis is that it is tractable and will save us time over the alternative method of constructing canonical graphs for each concept, which may not be tractable.

2.5 Text

I plan to use Genesis 1-3 as the textual basis or experimental testbed in my Master's Thesis. Accordingly, my ontology should reflect the lexemes present in this text. I have extracted the lexemes present in this text, which number 261 unique entries. Subsequently, I have extracted those 261 entries from the Hebrew lexicon. This mini-lexicon forms the basis of my methodological studies.

2.6 Other work

This report couldn't attempt completeness without at least mention of the doctoral work of Dr. Reinier de Blois (see de Blois (2000)). In his PhD thesis, Dr. de Blois laid the foundation for "a New Dictionary of Biblical Hebrew Based on Semantic Domains." The following is a short summary of his findings.

Dr. de Blois took the "Greek-English lexicon of the New Testament based on semantic domains" (Louw and Nida (1989)) as his point of departure. He took the four basic categories of that lexicon (Objects, Events, Attributes, and Relationals) and examined them, concluding that in Hebrew, Attributes resemble Events so much that they can be merged into a single category (p. 116). The category of Objects was further subdivided into eight subdomains (ibid.). The category of Events was further subdivided along cube with the dimensions four by three by three (p. 117), leading to thirty six subdomains within the category of Events. The category of Relationals was subdivided into three basic categories corresponding to Peirce's triad (Referents, Markers, and Relations). Each of these categories was then further subdivided by a dichotomy according to the semantic classes they replace, point to, or link, namely Objects or Events (p. 117).

The result is not an ontology, for the following reasons: First, there is no type-hierarchy based on the *is-a* relation. And second, the basic organizing category is not the concept, but the lexeme.

The work is interesting nonetheless, because it is an attempt at classifying Hebrew lexicalized meanings, which is also what we are trying to do in this report. Even if the work does not go all the way towards a full-scale ontology, it is an important step in the right direction.

3 Problem description

In this project, I wish to explore methods for automatically building an ontology of Genesis 1-3. The overall strategy is to take a concise Hebrew-English lexicon and marry it to WordNet. Which methods can be used in such a process? To what extent is it possible to use the proposed method, and which methods must be employed when it fails? What is left as residue after a matching of the Hebrew lexicon with WordNet, and why? When one constructs an ontology, it is always important to think of the intentionality behind the ontology. Which concerns must be addressed with regard to the intended use of the ontology in my Master's Thesis? Why are these concerns relevant?

These are some of the questions which we hope to answer in this report.

4 Theory

4.1 Introduction

Sowa (2000, p. 492) writes:

> "The subject of *ontology* is the study of the *categories* of things that exist or may exist in some domain. The product of such a study, called *an ontology*, is a catalog of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D."

Ontological studies are carried out somewhere on an axis going from purely metaphysical considerations to very formal considerations (formal ontology). Chisholm (1996) is an example of a work dealing mostly with metaphysical considerations. Works such as Woleński (1990) are to be found in the middle of the spectrum. At the formal end of the spectrum, Nilsson (2001) is found.

In this section, we describe the theoretical background necessary for our later methodological considerations. First, we give a survey of the most important concepts in formal ontology. We then discuss top-level categories, followed by a brief introduction to conceptual graphs. We then describe the Hebrew-English lexicon, followed by a description of WordNet.

4.2 Formal ontology

4.2.1 Introduction

In this section, we describe some ideas within formal ontology. The ideas are mostly taken from Sowa (2000) and Nilsson (2001).

4.2.2 Ontotypes

As stated earlier, an ontology is the product of the study of the *categories* that exist in some domain (Sowa (2000, p. 492)). These categories are variously called conceptual types, types, and ontotypes. From now on, we will adopt the terminology of calling them *ontotypes*.

An ontotype consists of three parts: A name denoting the ontotype, an extension, and an intension. The name identifies the ontotype when talking about it, and need not correspond to any word used in everyday speech, although we often choose names that do bear a resemblance to naturally occurring words.

4.2.3 Extension and intension

An ontotype has both an extension and an intension. The extension is the set of all individuals that are instances of the ontotypes. For example, the ontotype **cat** has as extension the set of all cats in the world.[2] The intension of an ontotype, on the other hand, is the set of properties that are common to all of the members of the extension.

It is necessary that an ontotype should have both an extension and an intension. If an ontotype had only an extension, there would be no way of distinguishing ontotypes that were distinct concepts but happened to have the same extension. As an example, let us take Mark Twain's famous line,

> "Reader, suppose you were an idiot. And suppose you were a member of Congress. But I repeat myself."

Taken at face value, it seems that Mr. Twain is making the ontological assumption that the ontotypes **idiot** and **member-of-congress** are co-extensional, i.e., have the same extension. But clearly he is making an intensional distinction. An **idiot**'s intension may include such properties as "is devoid of sense," "lacks ability to make prudent choices," etc., whereas the intension of a **member-of-congress** clearly includes such properties as "having been elected to the Congress."

The reason it is necessary to have intensions as a distinguishing factor for distinct ontotypes is that, according to set theory, two sets X and Y are equal if and only if $X \subseteq Y \wedge X \supseteq Y$. In other words, if two sets have the same members, they are the same set. Also, when two ontotypes, such as **Unicorn** and **Faun** have an empty extension, they would be the same concept if the only defining property of a concept were its extension. Therefore it is important to have more than the extension as a defining property of an ontotype. The intension provides that additional defining property.

It will be remembered that Aristotle invented the method of defining by genus and differentiae (see section 2.2.4). The intension is nothing but the accumulated set of differentiae of all of the supertypes of the ontotype in question. More on supertypes later.

[2] It is important to specify which world we are talking about. It could be the real world, but it may also be some possible world. Generally, extension is with reference to the domain D of which we are building an ontology, see Sowa (2000, p. 492).

4.2.4 Individuals

If a concept's extension is non-empty, it means that the ontotype has representatives or instances in the domain which the ontology describes. The instances are individuals of some sort. Note that there is a distinction between the *category*, which is the ontotype, and the *individuals belonging to* that category. The category, or ontotype, belongs in an ontology, whereas the individuals of a category belong in a catalog of individuals.

Ontotypes may, however, be higher-order types, not just first-order types. These higher-order types will then have lower-order types as instances. For example, **Color** may be a second-order type having such first-order types as **Red**, **Green**, and **Blue** as instances, see Sowa (2000, pp. 30–32). Another second-order type is **Act**, which can have any number of first-order types, such as **See**, **Run**, **Smell** as instances. Thus types may be instances of higher-order types. Both the first-order types and the second-order types would belong in an ontology, but the first-order types which were instances of second-order types would additionally belong in a catalog of individuals.

4.2.5 Formal ontologies

According to Nilsson (2001),

> "a formal ontology is a logical specification of essential concepts within a domain structured by relationships identified between the concepts." (p. 11).

Thus we have 'concepts' (also called 'types' or 'ontotypes') structured logically according to relationships that hold between the concepts.

4.2.6 Ontological relationships

It was mentioned that ontologies are structured by means of relationships that hold between ontotypes. The relation that is most often chosen as the structuring relation is that of 'inclusion' or '*is-a*'. (See Nilsson (2001, pp. 13–14).)

For example, if ontotype A *is-a* ontotype B, then ontotype A is said to be included in ontotype B. Ontotype A *inherits* all of the intensional properties of ontotype B, but adds further properties so that it gets a possibly smaller extension. Ontotype A is thus a *specialization* of ontotype B, and ontotype B is conversely a *generalization* of ontotype A. Ontotype A is said to be a *subtype* of B, and ontotype B is said to be a *supertype* of A.

When drawing the *is-a* relationship on paper, the established convention is to draw the more general ontotype *above* the more specific ontotype, with an upwards arrow or line indicating the *is-a* relationship.

4.2.7 Lattices

The *is-a* relationship can be seen as a partial ordering on ontotypes. As is well known within mathematics, partial orders give rise to the mathematical objects called lattices (see, e.g., Davey and Priestley (1990)).

Lattices have many properties that make them desirable as methodological tools when structuring ontologies. The two most important ones are that, by definition, for any non-empty set of ontotypes in a lattice ordered by the *is-a* partial ordering, it is always possible to find a unique least common supertype for all of them, and conversely, to find a unique greatest common subtype of them all.

4.2.8 Crux and meet

These two properties of lattices are formalized in the algebraic notions of 'crux' and 'meet' (Nilsson (2001, pp. 15–17)). For example, if we have the two ontotypes c_1 and c_2, we may form their least common supertype by the algebraic formula $c_1 + c_2$, and their greatest common subtype by the formula $c_1 \times c_2$. The $+$ operator is called the 'meet' operator, and the $\times$ operator is called the 'crux' operator.

4.2.9 Top and bottom

Two ontotypes are always present in any formal ontology. They are denoted as $\top$ and $\bot$, named "top" and "bottom" respectively. Other names include "**Universal**" or "**Entity**" for $\top$ and "**Absurdity**" for $\bot$ (Sowa (2000, p. 480, 499)). From now on, we will be calling them "**Entity**" and "**Absurdity**."

Entity is the most general type and predicates nothing of anything. It is a supertype of everything else in the ontology. **Absurdity** is the most specific type and predicates everything of everything. It is a subtype of everything else in the ontology

The reason we have these ontotypes in all ontologies is so as to be able to guarantee that the ontological lattice always has a least common supertype and greatest common subtype. They are, of course, **Entity** and **Absurdity** respectively.

4.2.10 Conclusion

In this section, we have given a survey of the most important concepts in the field of formal ontology. We have seen that a formal ontology consists of categories, or ontotypes, ordered by the is-a relation in a lattice. We have looked at the distinction between individuals and ontotypes, and seen that the latter belong in an ontology, whereas the former belong in a catalog of individuals. We have looked at the distinction between intension and extension, and seen that both are necessary as defining properties of ontotypes. We have looked at the crux and meet operators on ontotypes ordered by the is-a relation, and seen that they arise from the mathematical properties of lattices. Finally, we have defined the two ontotypes **Entity** and **Absurdity** and seen that they belong in all ontology-lattices.

4.3 Top-level categories

4.3.1 WordNet

WordNet's nouns and verbs are organized into a number of distinct hierarchies, each with a unique beginner (Miller (1998a, p. 28), Fellbaum (1998a, p. 71)). These unique beginners are not connected at the top, although Miller (1998a, p. 30) does suggest a way of organizing the unique beginners for nouns in a hierarchy with more levels.

4.3.2 Sowa

As has already been described, Sowa has brought together the top-level distinctions of Heraclitus, Peirce, and Whitehead into a single top-level ontology (see, e.g., Sowa (1995), Sowa (2000, p. 72)). This top-level ontology has the desirable property of forcing the builder of an ontology to think clearly about where any given ontotype belongs in the physical/abstract dichotomy (Heraclitus), the continuant/Occurrent dichotomy (Whitehead), and the Firstness/Secondness/Thirdness trichotomy (Peirce). This is important if one is to have an ontology that is both computationally efficient and ontologically powerful enough to say interesting things about the domain under consideration. The rather disorganized top-level ontology of Cyc (Lenat and Guha (1990)) stands in stark contrast to Sowa's very stringent top-level ontology (see Sowa (1995, pp. 671–673) for a comparison). Thus Sowa's work would have been nice to incorporate. However, this would require a careful assignment of each of the WordNet unique beginners to one of the categories in Sowa's top. This is a point for further research.

4.3.3 Martin

In an article on "Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition" (Martin (1995)), Dr. Philippe Martin took Sowa's top-level distinctions as he thought about them in Sowa (1992) and placed WordNet's unique beginners into this framework. Unfortunately, Sowa's 1992 work is not as complete as his later ontology described in section 2.2.11, thus we have chosen not to incorporate Martin's work.

4.4 Conceptual graphs

4.4.1 Introduction

Sowa's conceptual graphs (Sowa (1984), Sowa (2000)) will form part of the theoretical basis for my work in my Master's Thesis. We will also need to refer to them when we discuss methodological considerations later in this report. Therefore, we give a brief introduction to the subject here.

4.4.2 Conceptual graphs

A conceptual graph is a directed bipartite graph with two kinds of nodes: Concepts and Relations. Concepts may stand alone, but relations are always connected with at least one concept.

4.4.3 Concepts

A concept consists of a type and a referent. The type is either taken from an ontology, or it consists of a conceptual graph describing the type, or it is a primitive type (such as **Entity**). The referent describes the individual to which the concept refers, if any. The description can be as simple as an existential quantifier, in which case the referent is simply left blank as a short-hand. It can also be a so-called indexical, which is points to a specified referent by established conventions.

4.4.4 Relations

A relation is a node which modifies one or more concept nodes, and can bring two or more concept nodes into relation with each other. Typical relations include "AGNT" (agent), "PTNT" (patient), "RSLT" (result), and "SUCC" (successor).

A relation has a relation type which determines its signature. A relation's signature dictates which types the concepts which are attached to the relation can have.

4.4.5 Canonical graphs

A canonical graph for a word in a lexicon "represent[s] the default ways that concepts and relations are linked together in well-formed sentences." (Sowa and Way (1986, p. 57)). This is the basis on which the method of knowledge-extraction of Sowa and Way (1986) and their successors is based. A canonical graph restricts the number and type of concepts and relations that can be linked together to give rise to a conceptual graph that captures the meaning of the word in question.

4.4.6 Conclusion

Conceptual graphs are directed bipartite graphs consisting of concepts and relations. Concepts have a type and a referent, whereas relations only have a type. A relation type has an associated signature which determines the number and type of concepts that can be linked to the relation. Canonical graphs for words in a lexicon are conceptual graphs that represent the default ways of linking concepts and relations to give rise to a representation of the meaning of the word in question.

4.5 The lexicon

4.5.1 Introduction

The lexicon chosen is that of the Werkgroep Informatica, produced as a companion to the Quest 2 retrieval software (Syring (1998)). It has been handed to me in machine-readable form, and consist of one entry per line.

In the following, we describe various aspects of the lexicon which are necessary for the later methodological discussion. First, we list the parts of speech into which the lexicon classifies all lexemes. Second, we give a brief introduction to the lexical entries themselves. Third, we describe some of the kinds of information that a lexical entry holds. Fourth, we describe the structure of the glosses.

4.5.2 Parts of speech

The lexicon divides the lexemes into the following fourteen parts of speech:

0. Definite Article	5. Preposition	10. Interjection
1. Verb	6. Conjunction	11. Negative
2. Noun	7. Personal pronoun	12. Interrogative
3. Proper noun	8. Demonstrative pronoun	13. Adjective
4. Adverb	9. Interrogative pronoun	

4.5.3 Lexical entries

Each lexical entry is tied to a lexeme, and has a number of fields, some of which are present for all lexemes, and some of which are present only for certain parts of speech. Homographs are listed in separate entries.

4.5.4 Information in a lexical entry

The following fields are always present in a lexical entry:

1. Lexeme	3. English gloss
2. Part of speech	4. German gloss

For nouns, the following may be available:

1. gender	3. plural form
2. semantic set	4. dual form

For verbs, the following may be available:

1. verbal stem (binyan)
2. valency + prepositions

Other parts of speech may also have additional fields, but they are not considered important for our purposes.

4.5.5 Structure of glosses

Each gloss follows a pattern; in fact, the glosses follow a strict syntax. This syntax has been detailed in Appendix A. Here we note the following salient points:

1. Different *senses* of a lexeme have been separated with semicolons, whereas different glosses for the *same sense* have been separated with commas.

2. Sometimes, a gloss will have an elaboration either in <angle brackets> or in (parentheses). The angle-brackets are used for, e.g., classification according to kind (as in "<precious stone>") or classification according to grammatical function (as in "<interrogative>"). Parentheses are used for notional clarification, as in "(together) with", "be filled (with)", "belly (of serpents)".

4.5.6 Conclusion

The lexicon has fourteen parts of speech. Each lexical entry has some fields which are present for all entries, including part of speech and an English gloss. Different parts of speech have different additional fields. Each gloss is made up of at least one semicolon-separated *sense*, and each sense has one or more comma-separated *glosses*. Each gloss may have zero or more elaborations in parentheses or angle brackets.

4.6 WordNet

4.6.1 Introduction

In this subsection, we describe WordNet to the level of detail that is necessary for our purposes. First, we give an introduction to WordNet and its semantic relations. Then we describe some of the searches that can be performed with WordNet. Finally, we have a concluding section.

4.6.2 What is WordNet?

WordNet is an "electronic lexical database" (title of Fellbaum (1998b)). It contains almost two hundred thousand lexical items,[3] organized into so-called "synsets". A synset is a "synonym set" which contains a set of lexical items that are related by synonymy.[4]

Various semantic relations hold between the synsets. The synsets of nouns and verbs are related by, among other relations, hypernymy, which is what is important for our purposes.[5]

Adverbs and adjectives are not related by hypernymy. Instead, other relations such as antonymy, similarity, and gradation hold between adjectives.[6] Adverbs are organized mainly by synonymy, and so usually are just grouped in synsets with no relations to other synsets, although sometimes antonymy between synsets is recognized.[7]

4.6.3 Searches in WordNet

WordNet provides an API (Application Programming Interface) for programmers to make use of for querying and reading WordNet. This API makes available a number of search-types, of which the following are important for our purposes:

- Finding hypernyms recursively (up to the unique beginner).
- Finding synonyms (just synsets with no relations between synsets).

The search-functions return a data-structure which is basically a linked list of synsets. If the search is for recursive hypernyms, each synset also has a pointer upwards to a linked list of synsets which are immediate ancestors of the given synset, and so on recursively up to the unique beginners.[8]

4.6.4 Conclusion

WordNet organizes nouns, verbs, adjectives, and adverbs into so-called "synsets", which are sets of senses of lexical items which are related by synonymy. For the nouns and verbs, the synsets are organized hierarchically by hypernymy. Adjectives and adverbs are not ordered by hypernymy. Various searches can be performed, including a recursive search for hypernyms and a search for just synonyms. These functions return linked lists of synsets, which may also have pointers upwards in the hierarchy.

5 Method

5.1 Introduction

In the following, we describe methods which could be used to build an ontology automatically. In particular, we describe the method we have chosen, namely matching a concise lexicon with WordNet.

[3] See the `wnstats` Unix manual page in section 7 which comes with WordNet.
[4] See Miller (1998a, pp. 23–24).
[5] See Miller (1998a, pp. 25-28), Fellbaum (1998a, pp. 79–80) .
[6] See Miller (1998b, p. 48, 50, 52).
[7] See Miller (1998b, p. 61).
[8] See the `wnsearch` Unix man-page in section 3, available when you install WordNet.

We begin with some considerations on whether the enterprise is meaningful at all. We do so by discussing the processes which would be involved in constructing a type-hierarchy of canonical graphs automatically. We note some difficulties involved in this process, but conclude that some of the process could probably be automated. This bodes well for our present enterprise, which we view as a simpler task.

We then develop and motivate some concepts which we have found necessary for our methodology to work. Among them are entry clusters and ontology entries. We also specify what exactly an ontology is, from the point of view of this present study.

We then describe what should be in the ontology, based on the parts of speech in the lexicon. We argue that most parts of speech can be left out without causing us much trouble, from the point of view of our task in the Master's Thesis. We also discuss what to do about glosses which are not in WordNet, and propose a solution to this problem.

We then briefly describe the parts of speech that do go into the ontology, and explain what to do about them and why.

We then give a long section on the algorithms involved. The algorithms themselves are shown in Appendix B, whereas the space in this section is devoted to methodological concerns arising in the construction of the algorithms.

We then present two kinds of problems that arise from doing an actual match of the lexicon with WordNet. We also describe what we have done to overcome the problems.

We then describe some statistics related to the matching.

Finally, we conclude the section.

5.2 Canonical graphs or a mere typehierarchy

As already mentioned, there are generally speaking two approaches to extraction of meaning from text:

1. By means of unification of canonical graphs, and
2. By means of transformations of analyses of the text.

The former requires an ontology based on canonical graphs. The latter only requires a type-hierarchy with nodes placed in a lattice order by the subtype-relation.

For reasons already explained, we have chosen to go for the type-hierarchy without canonical graphs. Here we wish to reflect briefly on how an ontology with canonical graphs could have been constructed.

Firstly, the canonical graphs would have to be there for all concept- and relation-types (see Sowa and Way (1986, p. 59)). This amounts to a rather large amount of work for any reasonably sized ontology, since the canonical graphs serve two purposes, each of which requires careful consideration: One purpose is to provide the constraints holding for all lexical entries on issues like verb-valency, ontological selectional restrictions on arguments of verbs, and other pattens of allowable unification between concepts and relations. The other purpose is to place each concept into a type-hierarchy, which enables reasoning about constraints based on subsumption.

There are at least two ways of acquiring a suitable lattice of canonical graphs: Either the canonical graphs and their ordering lattice are written by hand, or they are (semi-)automatically created.

The hand-written approach will be the most accurate, if one is careful and meticulous in the process. It does, however, require a large amount of work, since ontological placement of categories is a difficult task, even for humans. Sowa (2000, p. 79) writes:

> "As Peirce and Whitehead noted, the way a physical entity is classified depends on the intention or subjective form of some perceiving agent."

This seems to suggest that ontological classification is a process that requires a perceiving agent who is capable of directing attention to certain characteristics of entities while discounting other characteristics. Sowa goes on to say (ibid.):

> "An intention . . . is the mental mediation or Thirdness that directs an agent's attention to some form that characterizes some entity."

So ontological classification is a process that requires a perceiving agent, capable of giving attention to characteristics of entities. At this point in time, with the current state of the art, unfortunately this entails the involvement of a human being, at least for processes as complex as ontological classification.

As already noted, ontological classification is only part of the process involved in constructing a lattice of canonical graphs, namely the ordering of the lattice itself. The other part involves creating the constraints on the ontotypes engraved in the form of the canonical graphs.

The creation of the canonical graphs involve operations that are more complex than the classification of ontotypes. The reason is that not only does the process involve some amount of ontological classification, since one would need to specify the ontotype of the defining concept, but it also involes determining which ontotypes are semantically co-selectional with the ontotype at hand, including which relations are semantically meaningful when relating the ontotypes in question. This involves not only knowledge of the domain for which the ontology is being constructed, but also knowledge of semantic restrictions within the domain. So we enter the realm of semantics when we attempt to construct ontologies based on canonical graphs.

According to Peirce,

> "A sign, or *Representamen*, is a First which stands in such a genuine triadic relation to a Second, called its *Object*, as to be capable of determining a Third, called its *Interpretant*, to assume the same triadic relation to its Object in which it stands itself to the same Object. The triadic relation is *genuine*, that is its three members are bound together by it in a way that does not consist in any complexus of dyadic relations. That is the reason why the Interpretant, or Third, cannot stand in a mere dyadic relation to the Object, but must stand in such a relation to it as the Representamen itself does." (Peirce: Collected Papers, 2.274)

For Peirce, a sign consists of an irreducible triad of:

1. an icon or symbol (the Representamen),
2. the object which the symbol represents, and
3. the meaning of the symbol, or its Interpretant, which arises in the mind because of the symbol.

Computers are not good at dealing with semantics. The reason is that it is either impossible, or we still do not know how, to create in the computer a mind that is capable of containing the meaning or Interpretant. All that is presently possible is to represent the Representamens.

Hoffmeyer (1996) writes:

> "All computer programs are completely based on Peircean "secondness", i.e. syntactic operations, since application 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. As Peter Cariani explains: there is no logical structure for the whole world so that the sign embedded in a logical "model" bears a definite logically-necessary relation to the world as model" (Cariani 1995). 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."[9]

Since meaning and intention are involved in the process of ontological classification and creation of canonical graphs, does this leave us without hope that ontologies can be constructed (semi-)automatically by means of a computer? Not quite. The computer can still be entrusted with parts of the process if it has the right data to work with. For example, an ontological classification of the lexical items involved may already be extant,[10] having been encoded previously, presumably by hand. Similarly, information about selectional restrictions may already be available, as for example the work of Joan Bresnan attempts to do,

[9] The "Cariani 1995" referenced here is: Cariani, Peter (1995). "*Towards an Evolutionary Semiotics: The Role of Symbols in Organisms and Adaptive Devices*". In Gertrudis Van de Vijver, Manela Delpos and Stanley Salthe (eds.) *Proceedings of The International Seminar on Evolutionary Systems ISES, Vienna*: Forthcoming.

[10] As in, for example, WordNet.

at least in part, within the framework of Lexical-Functional Grammar (e.g., Bresnan (1982)). Given the digitalization of these kinds of information, algorithms may be constructed which take away some of the burden of the processes of ontological classification and construction of canonical graphs. This premise is also the basis of the work being undertaken in this report.

5.3 Matching with WordNet

5.3.1 Introduction

In the following, we describe the method which we have developed for matching the given Hebrew lexicon with WordNet. We will develop the notions of ontology, ontology entry, and entry cluster. These will be central to our method later.

5.3.2 What is an ontology?

For our purposes, an ontology is a list of *entry clusters*, each of which contains zero or more *ontology entries*. The ontology entries are grouped in *entry clusters*, each with a unique WordNet synset. Since in WordNet, it is between synsets that the hypernym relation holds, the entry clusters are also ordered in a hypernymically ordered lattice.

5.3.3 What is an ontology entry?

An ontology entry represents one sense of a given lexeme. Thus it is lexeme- and sense-based. As explained in section 4.5.5 on page 16, each lexical entry may have more than one sense, separated by semicolons, and within each sense, several glosses may be given, separated by commas. Each ontology entry represents one sense, that is, it represents a merging of the (comma-separated) sense(s) of a given lexical entry, whereas n semicolon-separated senses would give rise n ontology-entries.

Each ontology entry has the following information:

1. An id that is unique within the ontology.
2. The lexeme on which it is based.
3. An English gloss.

What it does not have is a placement within the WordNet ontology. This is provided by the containing entry cluster.

5.3.4 What is an entry cluster?

An *entry cluster* represents a grouping of zero or more ontology entries under the heading of a single WordNet synset. It also has pointers to zero or more other entry clusters to which it stands in a hyponymic relationship. That is, it has pointers upwards in the ontology to the synsets above it. The only reason there are *zero* or more pointers upwards, and not *one* or more pointers is that the top-level category, **Entity**, does not have anything above it. All other categories have one or more pointers upwards.

Thus an entry cluster represents a single ontotype in the ontology.

Each entry cluster has the following information:

1. An id that is unique within the ontology.
2. The WordNet synset on which it is based.[11]
3. An English gloss.
4. A set of ids pointing to the entry clusters immediately above it in the hierarchy.

[11] Actually, it is a string-representation of the WordNet synset, coupled with an integer which distinguishes the synset from other synsets with the same string-representation.

5. A list of the ontology entries that it contains.

Point number 5 is worth noting. An ontology entry obtains its place in the lattice only indirectly, through the entry cluster of which it is a part.

Some entry clusters will have an empty list of ontology entries. This arises, e.g., in the situation when a synset is needed in the hierarchy but no lexical entry corresponds to it. It also arises in the case of the **Entity** ontotype, since no ontology entries can be placed in this most general concept.

5.4 What should be in the ontology?

5.4.1 Introduction

Not all parts of speech in the lexicon are represented in WordNet. This may be a problem for our pursuit, but then again, it may not. In the following subsections, we discuss this issue. In a section on the parts of speech, we discuss why this is not a big problem. In a section on proper nouns, we argue that proper nouns do not belong in the ontology, but rather in a catalog of individuals. In a section on glosses not found in WordNet, we discuss what to do about this phenomenon. We also discuss the inclusion of Sowa's top-level categories. Finally, we summarize this section in a conclusion.

5.4.2 Parts of speech

Of the parts of speech listed in section 5.4.2, only four are available in WordNet, viz. Noun, Verb, Adjective, and Adverb. The other parts of speech are not represented in WordNet.

This situation presents us with a choice: Either we include the other parts of speech that are not accounted for in WordNet, or we leave them out. I propose to leave most of them out, for the following reasons:

1. Noun, Verb, Adjective, and Adverb are the parts of speech that have the most semantic vs. grammatical content. The other parts of speech do also have meaning, but their contributions to the meaning of a sentence are more grammatical in nature than semantic.

2. The other parts of speech, except for Proper noun, are all "closed" classes of lexemes – that is, their representatives form classes of lexemes that linguists normally think of as "closed," meaning that a living language would not produce more lexemes of the given parts of speech (see Matthews (1997, p. 57)). This, coupled with the fact that each class contains a relatively low number of lexical entries, entails that it is feasible to deal with them separately from the ontology.

3. It is not evident that the other parts of speech belong in the ontology at all. For example, prepositions always occur together with noun phrases to form prepositional phrases, and the semantic contribution of the preposition is likely to give rise to a relation rather than a concept. This means that it is probably better to deal with prepositions based on the relations they induce rather than any concepts they would represent. This is what Jensen, Nilsson and Vikner (2001) and Madsen, Pedersen and Thomsen (2001) do concerning prepositions. Similarly, personal pronouns and demonstratives are probably better dealt with in rules than in the ontology, given that they are likely to induce indexicals such as #he, #his, #this, etc. rather than separate concepts. The conjunctions are likely to give rise to juxtaposition of graphs ("and"), disjunction of graphs ("or"), causal relationships between contexts ("because"), and similar constructions. Interrogatives could give rise to either free variables or monadic relations signaling the interrogative illocutionary force of the sentence. Negatives are likely to give rise to negation operators. Interjections might be flagged as propositions with special indexical referents. The point is that in all of these cases, the structural contribution to the resulting conceptual graph is not a concept but something else, be it an indexical, a relation, or a negation operator. Thus they probably do not belong in the ontology, since the ontology deals with the concept types necessary for creating and dealing with concepts.

5.4.3 Proper nouns

The only part of speech that is not covered in WordNet and which might be desirable in the ontology is Proper noun. The lexicon lists, for each proper noun, its so-called "semantic set," drawn from the following set of labels:

1. "pers" for a personal name	3. "topo" for a topographical name
2. "gens" for gentilic, or name of a people group	4. "mens" for the name of a month

This could suggest that each lexical item being a proper noun should be placed within the synset category (also called "entry cluster," see above) which best describes the semantic set to which the proper name belongs. This would be a mistake, however. The reason is the distinction between individuals and ontotypes that was mentioned in section 4.2.4. The proper nouns refer to individuals, not categories of things, and so they don't belong in the ontology, but rather in a catalog of individuals.

The suggested way of dealing with proper nouns in the Masters thesis-work is as follows: When encountering a proper noun, look up the synset entries in the ontology which best correspond to the semantic set(s) associated with the lexeme. If there is more than one, try to deduce the one that best fits the context. Then use that one as the concept type, and place the English gloss, available in the lexicon, into the referent of the concept.

5.4.4 Glosses not in WordNet

What should be done with lexical entries having glosses which are not in WordNet? I think the answer is that they should somehow be placed in the ontology. The following methods could be used:

1. Emend the lexicon so as to replace the gloss with one that corresponds to the meaning of the word and which is in WordNet.
2. Programmatically place the lexical entry in the ontology using hard-coded program code.

I believe that the former is a better solution, for two reasons:

1. It does not alter the method involved. This is clearly desirable, since simplicity of method is a goal of scientific pursuit.
2. It is far easier to do, since it only involves editing of a single line, rather than:
   (a) Looking the word up in WordNet to see which synset it should belong to.
   (b) Hand-editing the code that places it into the ontology, which could involve several lines of code.

Thus the former is what I have done, using Holladay (1988) as my reference lexicon.

5.4.5 Conclusion

Not all parts of speech present in the lexicon are represented in WordNet. We have argued that this is not a significant problem, and that the parts of speech not present in WordNet do not really belong in the ontology. Proper nouns, in particular, belong in a catalog of individuals rather than the ontology proper, and the other parts of speech find their natural expression in other ways than through ontotypes. Finally, we have also described a simple method of dealing with lexical entries whose glosses do not appear in WordNet.

5.5 Parts of speech

5.5.1 Introduction

In this section, we detail what we have decided to do with each part of speech that is to be included in the ontology. We treat noun and verbs first, followed by adjectives and finally adverbs. We end with a conclusion.

5.5.2 Nouns and verbs

Nouns and verbs are ordered hypernymically in WordNet, which allows us to do so in our ontology as well. We simply take over WordNet's type-hierarchy, adding all hypernyms up to the unique beginners, even if these extra hypernyms are not used for containing ontology-entries.

The reason why we add the extra hypernyms is that we wish the type-hierarchy to be complete. We could have gone to great lengths of only adding those hypernyms which were absolutely necessary, but this was not possible within our limited time, and is a topic for further research.

Another topic for further research is whether WordNet's type-hierarchy is at all adequate for describing Hebrew. WordNet is, after all, an ontology of English, not of Hebrew, and there is always some semantic skewing between concepts in different languages. There is a chance, however, that WordNet might be at least adequate, if not totally covering the Hebrew concepts. We shall see later (5.8) that there is empirical evidence that WordNet is adequate to some extent.

5.5.3 Adjectives

Adjectives are not ordered hypernymically in WordNet. Most of them will be an attribute of some sort, and so we have decided to place them all in entry clusters which have one ancestor, namely the entry cluster based on the synset "{ property, attribute, dimension }." This is a simplification, but we believe that it will suffice for our purposes. It will be a matter for further research whether it is sufficient.

5.5.4 Adverbs

Adverbs are more difficult to deal with than adjectives. They can have a number of functions, and so are not easy to classify. We have chosen to treat them analogically to adjectives, and place them in entry clusters beneath the entry cluster based on the synset "{ manner, mode, style, way, fashion }." Thus all adverbs are considered as a kind of "manner." This is obviously not adequate, but it is the best we can do given our limited time. It will be a matter for further research what can be done to remedy the situation.

5.5.5 Conclusion

Nouns and verbs will be ordered hypernymically according to WordNet's hierarchy. Adjectives will be placed in the synset corresponding to "attribute," and adverbs will be placed under the synset corresponding to "manner."

5.6 Algorithms

5.6.1 Introduction

In this section, we describe the methodological concerns behind the algorithms presented in Appendix B. We will describe the program in a top-down fashion, starting with the main program and gradually winding our way down the directed graph of algorithms that call each other. We will refer to Appendix B for the details and only note points that have significance for methodological concerns.

5.6.2 The main program

The main program can be seen in pseudo-code in Appendix B in section B.2. It simply reads in the lexicon, and initializes an ontology with the **Entity** ontotype, as well as the ontotypes corresponding to "attribute" and "manner" and their hypernyms. It then iterates over the lexicon entries, discarding entries that are not nouns, verbs, adjectives, or adverbs, and calls the top-level matching algorithm for entries of these kinds. Finally, it outputs the ontology in either XML format or tree format.

5.6.3 The top-level matching algorithm

The top-level matching algorithm `MatchLexeme` (see B.3) takes as input a lexical entry. It has the side-effect of adding zero or more ontology entries to the ontology, and zero or more entry clusters.

Zero ontology entries added would mean that the lexeme cannot be matched. In this case, we write a message on the screen to that effect, and simply discard the lexeme. We could instead have placed it inside of some top-level entry-cluster about which we were reasonably sure that it would at least not be wrong. However, this would lessen the usefulness of the ontology, so we choose to write a message instead so we can fix it by hand-editing the lexicon.

One ontology entry added would mean that there was only one sense in the lexeme, and it incurred a match.

More ontology entries added would mean that there was more than one sense in the lexeme, and two or more of these incurred a match.

One entry cluster added would mean that the WordNet synset which corresponded to the lexeme was not already present in the ontology.

Zero entry clusters added would mean the entry cluster(s) needed were already present, and they would just contain more than one ontology entry after the addition. There could be more than one entry cluster, because there could be more than one sense, and senses are the basis of ontology entries, and ontology entries might end up in different entry clusters.

5.6.4 Traversing the AST

When matching, the algorithm `MatchLexeme(L)` generates an Abstract Syntax Tree (AST[12]) of the gloss. This AST is traversed in the algorithm `TraverseAST` in such a way that the following hold:

1. Each semicolon-separated sense ends up in a separate ontology entry.
2. Each comma-separated gloss contributes to the placement of one ontology entry.

A note on step 2 in the algorithm (see B.4): When the definition is just an elaboration, there is really no gloss to match, since these are reserved for things like "<interrogative>" and "<object marker>". The pure elaboration-definitions are thus for parts of speech that are not in our target set (Verb, Noun, Adjective, Adverb), and so it is not a problem that we just return without doing further matching.

5.6.5 Matching a dterm

The `MatchDTerm` algorithm takes care of one sense of a lexeme. It takes a dterm node from the AST and a lexicon entry and calls two subroutines, `GetSet` and `MatchStringSet` for the grunt work. We refer the reader to section B.5 for the details.

5.6.6 Generating definition-string-sets

The `MatchDTerm` algorithm traverses the AST derived from the gloss by the grammar in Appendix A. It calls the present algorithm, `GetSet`, which generates a set of strings which may be candidates for matching with WordNet.

The main thrust of the algorithm is summarized in the following two points:

1. It puts each individual, comma-separated gloss into the set, which allows us to search on them individually.
2. It combines those glosses which have elaborations with the elaborations, thereby producing at least one extra string apart from the gloss itself, more if the elaboration is a comma-separated list. For example, "at (time, place)" would generate the following three strings: "at", "at time", "at place".

The details, which can be seen in section B.6, are largely uninteresting from a methodological standpoint, except that they carry out the above two operations.

[12] Abstract Syntax Trees constitute a well-known class of data structures in computer science for handling parsing of languages. See Appel (1997) for an introduction.

5.6.7 Matching a string-set

The `MatchDTerm` algorithm generates (by way of the `GetSet` algorithm) a set of strings which are possible definitions of the given sense of the given lexeme. The `MatchStringSet` algorithm (see B.7) takes these possible definitions and attempts to find at least one match with WordNet. If at least one match is found, an ontology entry is created and added.

1. If there is only one match, the following heuristic is used: We choose the first synset returned by WordNet. Otherwise, the following heuristics are used.

2. If two or more glosses match the same WordNet synset, then the WordNet synset which has the greatest number of common matches is chosen. This is exemplified by the sense "bad, evil, harmful", where "evil" and "harmful" belong to the same synset, whereas "bad" is only indirectly related to these two by way of a common ancestor. (See B.8 and B.9.)

3. Failing that, we choose the heuristic based on the part of speech of the lexeme:

   (a) If the part of speech is an adjective, we choose the first synset returned by the WordNet search algorithms. This will often be the right one, since it is often the most basic sense that is returned as the first one. We then add this synset as an entry cluster to the ontology if not already there, placing it directly beneath the synset corresponding to "attribute." We then add the ontology entry to the entry cluster. This is done in MatchAdjectiveOrAdverb (see B.11).

   (b) Similarly for adverbs, we choose the first synset, place it directly beneath the synset corresponding to "manner" if not there already, and place the ontology entry within the entry cluster (see B.11).

   (c) If the part of speech is noun or verb:

      i. We find the least common hypernym of all of the results. (See B.12, B.13, and B.14.)

      ii. If there is no such hypernym, or if it is a unique beginner, we take the last recourse available, and take the first synset of the first result.

The reason why we choose the least common hypernym is that the meet operator (which is what is simulated here) produces the most specific ontotype that is still a description of all of the matches. Thus we get the ontotype that is closest to all of the glosses while still containing them all.

5.7 When the algorithm doesn't work

5.7.1 Introduction

In two ways, the algorithm failed. Here we describe them both, and what we did about these problems. First, we describe lexemes that did not match at all, and second, we describe lexemes that did match but which were misplaced in the ontology.

5.7.2 Lexemes that did not match at all

Out of 261 lexemes, only 31 are not covered by the above algorithm. That is, the algorithm fails completely for only 31 of the lexemes, which amounts to 11.9%. That is not bad, and results in a manageable amount of lexemes to treat specially.

We proposed above (see section 5.4.4) to hand-edit the lexicon in the cases where the matching failed for all senses of the lexeme. Here we give a few examples of what kinds of decisions were made in hand-editing the lexicon.

- The lexeme "`<BWR`" had a gloss of "on account of, for the sake of; in order that". Looking it up in Holladay (1988), we find that it always occurs in combination with the preposition "be", meaning "in," which explains the presence of the prepositions and other particles in the definition. Thus we have replaced the gloss with "sake, interest; reason, ground". The first sense ("sake, interest")

corresponds to the synset with the gloss “(a reason for wanting something done; "for your sake"; "died for the sake of his country"; "in the interest of safety"; "in the common interest"),” while the second sense, “reason, ground” corresponds to the synset with the gloss “(a rational motive for a belief or action; "the reason that war was declared"; "the grounds for their declaration").” We feel that these synset correspond best to the senses of the word. And lo and behold, the lexeme ends up in the right entry clusters.

- The lexeme “<WP” had a gloss of “birds; flying insects.” We changed this to “bird; insect,” since WordNet does not distinguish between insects that fly and those that do not. Again the lexeme ends up in the right entry clusters.

- The lexeme “BDLX” had a gloss of “bdellium-gum.” WordNet does have “bdellium”, so we simply changed the gloss to that, and again the lexeme shows up in the right synset.

- The verb-lexeme “BDL” had a gloss of “be separated; separate oneself.” WordNet did not have these collocations, so it was not matched. Looking up the lexeme in Holladay (1988) yielded such definitions as “separate oneself,” “be excluded from,” “be singled out,” “separate, distinguish,” “separate, segregate.” The WordNet synsets that best describe all of these can be captured by the glosses “distinguish, separate, differentiate” and “separate, disunite, divide, part”. We changed the glosses accordingly, and again the lexemes show up in the right entry clusters.

- One problem occurred with the noun-lexeme “CRY”, which according to Holladay (1988) refers to “swarming things: tiny animals occurring in large numbers, in water . . . , in air . . . , on ground . . . .” The problem is, English does not have a word that covers all of these. However, the only place in our text where it occurs is Genesis 1:20, which says “Let the water teem with living creatures [CRY]”. So in our text, it refers to aquatic animals. We replaced the gloss with “marine animal,” which is what comes closest in WordNet, since there is no term for “teeming marine animals.” Again the lexeme ends up in the right entry cluster.

- Another problem occurred with “QDMH”, which has the gloss “in front of” in the lexicon. Holladay (1988) has “in front of, opposite.” Consulting our text, we see that it occurs in Genesis 2:14, where it is found in the clause “in front of Assyria” (American Standard Version). The word is a noun in Hebrew, standing in a “genitive” relationship to Assyria. So it is a special feature of Hebrew that this item is lexicalized as a noun and not a preposition. We have chosen to leave it as it is, without attempting a match. This is because the alternative would be relabeling it as a preposition, and have the graph-extraction algorithm in the Master’s thesis figure out what to do with it. But that decision is better left to a later time, when actually doing the Master’s thesis. In addition, since this is a preposition in English, there is no way of using WordNet to obtain a placement in the ontology, since WordNet does not have prepositions. So this is one instance where the matching algorithm did not work, and we couldn’t rectify the situation by hand-editing the lexicon, nor by modifying the algorithm. This was due to an idiosyncrasy of Hebrew, namely lexicalizing this concept as a noun.

- Another problem we couldn’t solve by hand-editing the lexicon occurs with the quite important verb “VWB” (tov), which means “be good.” This is not lexicalized in English as a verb, and so we can’t use WordNet to obtain a placement in the ontology. It will be up to our work in our Master’s thesis what to do about the lexeme.

Thus the cases where the algorithm did not work were solved by careful selection of the synset from WordNet that best matched the sense(s) of the word, and entering new glosses into the lexicon that would ensure that the right synsets were found. By hand-editing the lexicon, we have been able to bring the number of unmatched lexemes down from 31 to 2.

5.7.3 Lexemes that matched wrongly

A number of lexemes were matched to the wrong synset, at least from the point of view of the text. Here we give examples of these and how we overcame the problems.

- The lexeme “PNH” had a gloss of “face; surface; front.” This was far too wide for the text, which among other places ended up in the entry cluster with the synset “{ presence, front }.” This choice stemmed from our default policy of choosing the first synset in the list of results, when all else failed. In the text, however, only the meanings of “surface” and “face” were present, so we limited the definition to these glosses, and the terms showed up in the right entry clusters.[13]

- Likewise, the noun-lexeme “QDM” had a gloss of “front; east; aforetime; ancient time,” but is only present in the text in the sense “east.” So we reduced the gloss accordingly, and the ontology entry shows up in the right entry cluster.

- The noun-lexeme “DRK” had a gloss of “way.” In our text, it occurs only in Genesis 3:24, where we read that the Lord God posted cherubim and a flashing sword “to guard the way to the tree of life” (NIV).[14] Unfortunately, WordNet does not have this sense for “way” in a synset that is characterized by other than “way,” and it is not the first in the list of results returned. Therefore, our algorithm has no way of knowing which sense to pick, so it picks the first, which is not correct. Instead, we have given the lexeme another gloss, “road,” which makes it show up in the right entry cluster.

5.7.4 Conclusion

Two kinds of problems manifested themselves. First, some lexemes did not match at all. By hand-editing the lexicon, we were able to bring down the number of lexemes not matched from 31 to 2. Second, some lexemes were misplaced. We were also able to overcome these problems by hand-editing the lexicon to make the lexemes show up in the right entry clusters.

5.8 Statistics

Out of 261 lexemes in the lexicon, 2 were not successfully matched, while 219 lexemes were matched successfully. Thus 83.9% of the total number of lexemes were successfully matched, while 0.8% were not successfully matched. Of these 221 lexemes, 123 were nouns, 78 were verbs, 16 were adjectives, and 4 were adverbs. The rest of the lexicon, numbering 40 lexemes, consisted of the other parts of speech which we chose not to deal with.

Thus most of the lexicon was covered by the method, giving empirical weight to the claim that leaving out the parts of speech which were not Noun, Verb, Adjective, and Adverb did not constitute a significant drawback.

5.9 Conclusion

In this section, we have motivated and developed a method for algorithmically matching a concise Hebrew-English lexicon with WordNet, producing an ontology automatically. We have explained what an ontology is, from our point of view, and we have developed two concepts which have been the vehicle through which our ontology has been constructed, namely ontology entries and entry clusters. We have argued that only the parts of speech which are present in WordNet, viz. Noun, Verb, Adjective, and Adverb, need be considered for our purposes. We have then developed and motivated algorithms for doing the actual matching. Once the algorithms were in place, we were able to run actual matches. In this process, a number of problems surfaced, most of which we were able to handle by hand-editing the lexicon, without modifying the method. We also presented some statistics, which gave weight to the earlier claim that leaving out certain parts of speech did not consitute a significant problem.

[13] One way to overcome this would have been to take hints from the semicolon-separated senses as to which synset was right. This may not be a good idea in general, however. It is a matter for further research whether and how to do it.

[14] The letters NIV mean “New International Version” and refer to the Bible translation that is cited. Used by permission of Zondervan.

6 Conclusion

In this report, we have described methods for automatically constructing an ontology of the Hebrew text of Genesis chapters 1-3 in the Hebrew Bible. The overall strategy has been to match a concise Hebrew-English lexicon with WordNet.

We have given a historical overview of the field of ontology (2.2), and we have presented some earlier work (2.6). We have described the theory behind formal ontology (4.2), as well as WordNet (4.6) and the Hebrew-English lexicon on which we base our work (4.5). We have developed a method for constructing the ontology (5), and we have developed some concepts which are useful when using the method (5.3). We have described what to do when the method fails (5.7), and we have described some of the concerns with regard to the intended usage of the ontology (5.4.2 and 5.4.3).

Thus we have covered all of the questions in the problem description. The end result is an ontology of Genesis 1-3, a representation of which can be seen in Appendix C.

7 Further research

In our work, a number of questions have come up which we have not had time to answer. Among them are:

- Is it always adequate to place adjectives beneath the synset meaning “attribute”? If not, what can be done about it?
- It is probably not adequate to place adverbs beneath the synset meaning “manner.” What is the severity of the problem, and what can be done to remedy this deficiency?
- Which lexemes, apart from the ones already found, did not get placed correctly? Can such lexemes always be coerced into the right place, if only one hand-edits the definition?
- Which lexemes have ontology entries which are not used in the text? Such ontology entries could be left out, since they clutter the ontology and make the task in the Master’s Thesis more difficult, since there will be more ontology entries for the same lexeme to choose from.
- Sometimes, lexemes will have senses which are related. Could these related senses be taken as hints for placement of the ontology entries arising from the various senses? If so, how? Would it be a good thing to do in the first place?

References

Appel, A. W. (1997). *Modern Compiler Implementation in C: Basic Techniques*, Cambridge University Press, Cambridge, UK.

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

Brentano, F. (1975[1862]). *On the several senses of being in Aristotle*, University of California Press, Berkeley. Translated by Rolf George from Brentano (1862), *Von der mannigfachen Bedeutung des Seienden nach Aristoteles*.

Bresnan, J. (ed.) (1982). *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, Mass. and London.

Chisholm, R. M. (1996). *A Realistic Theory of Categories: An Essay on Ontology*, Cambridge University Press, Cambridge, New York, Melbourne.

Davey, B. A. and Priestley, H. A. (1990). *Introduction to Lattices and Order*, Cambridge University Press.

de Blois, R. (2000). *Towards a New Dictionary of Biblical Hebrew Based on Semantic Domains*, PhD thesis, Vrije Universiteit, Amsterdam.

Fellbaum, C. (1998a). A semantic network of English verbs, in *WordNet: An Electronic Lexical Database* (Fellbaum 1998b), pp. 69–104.

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

Hardmeier, C., Syring, W.-D., Range, J. D. and Talstra, E. (eds) (2000). *Ad Fontes! Quellen erfassen - lesen - deuten. Was ist Computerphilologie?*, Vol. 15 of *APPLICATIO*, VU University Press, Amsterdam.

Hess, J. and Cyre, W. R. (1999). A CG-based behavior extraction system, *in* W. Tepfenhart and W. Cyre (eds), *Conceptual Structures: 7th International Conference on Conceptual Structures, ICCS'99, Blacksburg, VA, USA, July 1999, Proceedings*, Vol. 1640 of *Lecture Notes in Artificial Intelligence (LNAI)*, Springer Verlag, Berlin, pp. 127–139.

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.

Holladay, W. L. (ed.) (1988). *A Concise Hebrew and Aramaic Lexicon of the Old Testament*, E.J. Brill, Leiden. ISBN 0-8028-3413-2.

Jensen, P. A. and Skadhauge, P. (eds) (2001). *Ontology-based Interpretation of Noun Phrases: Proceedings of the First International OntoQuery Workshop*, number 21/2001 in *Skriftserie - Syddansk Universitet, Institut for Fagsprog, Kommunikation og Informationsvidenskab*, Dept. of Business Communication and Information Science, University of Southern Denmark, Kolding.

Jensen, P. A., Nilsson, J. F. and Vikner, C. (2001). Towards an ontology-based interpretation of noun phrases, *in* Jensen and Skadhauge (2001), pp. 43–55.

Kant, I. (1929[1781,1787]). *Critique of Pure Reason*, St. Martin's Press, New York. Translated by Norman Kemp Smith from the German *Kritik der reinen Vernunft*.

Künzel, W. v. and Bexte, P. (1993). *Allwissen und Absturz – Der Ursprung des Computers*, Insel, Frankfurt am Main und Leipzig.

Lenat, D. B. and Guha, R. V. (1990). *Building Large Knowledge-Based Systems*, Addison-Wesley, Reading, MA.

Louw, J. P. and Nida, E. A. (eds) (1989). *Greek-English lexicon of the New Testament: based on semantic domains*, Vol. 1 & 2, second edn, United Bible Societies, New York. assisted by Rondal B. Smith, part-time editor and Karen A. Munson, associate editor.

Madsen, B. N., Pedersen, B. S. and Thomsen, H. E. (2001). Defining semantic relations for ontoquery, *in* Jensen and Skadhauge (2001), pp. 57–88.

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.

Matthews, P. H. (1997). *The Concise Oxford Dictionary of Linguistics*, Oxford University Press, Oxford.

Miller, G. A. (1998a). Nouns in wordnet, *in* Fellbaum (1998b), pp. 23–46.

Miller, K. J. (1998b). Modifiers in wordnet, *in* Fellbaum (1998b), pp. 47–67.

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), *Foundations and Applications of Conceptual Structures – Contributions to ICCS 2002*, Bulgarian Academy of Sciences, Sofia, Bulgaria, pp. 15–28.

Nilsson, J. F. (2001). A logico-algebraic framework for ontologies: Ontolog, *in* Jensen and Skadhauge (2001), pp. 11–35.

Peirce, C. S. (1891). Review of *Principles of Psychology* by William James, *The Nation* **53**: 32.

Porphyry (1992). *On Aristotle's Categories*, Cornell University Press, Ithaca, NY. Translated by S.K. Strange.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA.

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, pp. 113–137.

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

Sowa, J. F. (1995). Top-level ontological categories, *International Journal of Human-Computer Studies* **43**(5/6): 669–685.

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

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.

Syring, W.-D. (1998). Computergestützte Philologie und Exegese, *Zeitschrift für Althebraistik* **11**: 85–89.

Tajarobi, A. (1998). *La reconnaisance automatique des hyponymes*, Master's thesis, Département de langues et linguistique, Université Laval.

Talstra, E. and Postma, F. (1989). On texts and tools. a short history of the werkgroep informatica (1977-1987), *in* Talstra (1989), pp. 9–27.

Talstra, E. and Sikkel, C. (2000). Genese und Kategorientwicklung der WIVU-Datenbank, *in* Hardmeier et al. (2000), pp. 33–68.

Talstra, E. and Van Wieringen, A. (eds) (1992). *A Prophet on the Screen – Computerized Desccription and Literary Interpretation of Isaianic Texts*, Vol. 9 of *APPLICATIO*, VU University Press, Amsterdam.

Talstra, E. (ed.) (1989). *Computer Assisted Analysis of Biblical Texts*, Vol. 7 of *APPLICATIO*, VU University Press, Amsterdam.

Verheij, A. J. (1994). *Grammatica Digitalis I – The Morphological Code in the «Werkgroep Informatica» Computer Text of the Hebrew Bible*, Vol. 11 of *APPLICATIO*, VU University Press, Amsterdam.

Whitehead, A. N. (1929). *Process and Reality: An Essay in Cosmology*, Macmillan, New York.

Whitehead, A. N. and Russell, B. (1925). *Principia Mathematica*, 2nd edn, Cambridge University Press, Cambridge, UK. first edition (1910).

Woleński, J. (ed.) (1990). *Kotarbiński: Logic, Semantics and Ontology*, Vol. 40 of *Nijhoff International Philosophy Series*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Zhang, L. and Yu, Y. (2001). Learning to generate CGs from domain specific sentences, *in* H. Delugach and G. Stumme (eds), *Conceptual Structures: 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July/August 2001, Proceedings*, Vol. 2120 of *Lecture Notes in Artificial Intelligence (LNAI)*, Springer Verlag, Berlin, pp. 44–57.

A Grammar for glosses

The following is a context-free grammar for the glosses. It is written in the style of the standard Unix program 'yacc' (Yet Another Compiler Compiler). All code implementing the parser actions has been removed, so that only the grammar itself is left, along with the specification of the tokens, also called terminals.

```
%token T_WORD                     /* "[a-zA-Z]" */
%token T_KEY_SEMICOLON            /* ";" */
%token T_KEY_COMMA                /* "," */
%token T_KEY_OPEN_PARENTHESIS     /* "(" */
%token T_KEY_CLOSE_PARENTHESIS    /* ")" */
%token T_KEY_OPEN_ANGLE_BRACKET   /* "<" */
%token T_KEY_CLOSE_ANGLE_BRACKET  /* ">" */
%%
definition : elaboration
  | definition_item
;
definition_item : dterm
  | definition_item T_KEY_SEMICOLON dterm
;
dterm : dfactor
  | dterm T_KEY_COMMA dfactor
;
dfactor : word_list
  | word_list elaboration
  | elaboration word_list
;
word_list : word
  | word_list word
;
word : T_WORD
;
elaboration : T_KEY_OPEN_ANGLE_BRACKET
              elaboration_list
              T_KEY_CLOSE_ANGLE_BRACKET
  | T_KEY_OPEN_PARENTHESIS
    elaboration_list
    T_KEY_CLOSE_PARENTHESIS
;
elaboration_list : elaboration_item
  | elaboration_list T_KEY_COMMA elaboration_item
;
elaboration_item : word_list
;
```

B Algorithms

B.1 Introduction

This appendix shows pseudo-code for the algorithms described in section 5.3.

B.2 The main program

```
MAIN PROGRAM:
1. Read in the lexicon.
2. Initialize an empty ontology.
3. Populate the ontology with the top-level Entity
   entry-cluster
4. For each lexicon-entry:
   a) If its part-of-speech is Verb, Noun, Adjective,
      or Adverb, then call sub-routine MatchLexeme
      with the lexicon-entry as an argument.
   b) Otherwise, discard the entry.
5. Write an XML representation of the ontology on
   standard output.
```

B.3 MatchLexeme

```
MatchLexeme(L):
1. Parse the lexeme's gloss as per the grammar
   in Appendix A, constructing an Abstract Syntax
   Tree (AST).
2. Traverse AST using algorithm TraverseAST(AST,L).
```

Step (1) has been constructed in accordance with the principles in chapters 2-4 of Appel (1997).

B.4 TraverseAST

```
TraverseAST(AST,L):
1. Get current node in AST. Call it E.
2. What kind is E?
   definition:
      - If it is an elaboration, return.
      - If it is a definition_item, call ourselves
        with the definition_item as the AST parameter.
   definition_item:
      - If it is a dterm, call MatchDTerm(dterm,L).
      - If it is a definition_item SEMICOLON dterm,
        call MatchDTerm(dterm,L) and call ourselves
        with the definition_item as the AST
        parameter.
```

B.5 MatchDTerm

```
MatchDTerm(dterm,L):
1. Initialize a set of strings S to be empty.
2. Generate a set of definition-string candidates
   from the comma-separated list of glosses by
   calling GetSet(dterm,S).
3. Call MatchStringSet(S,L).
```

B.6 GetSet

```
GetSet(AST-node,S):
1. What kind is AST-node?
   dterm:
```

```
      - If it is a dfactor, call ourselves with
        dfactor and S.
      - If it is “dterm COMMA dfactor”, call
        ourselves first with the dfactor, then with
        the dterm, as well as S.
   dfactor:
      - If it is a word_list, add it to S.
      - If it is a “word_list elaboration”, add
        to S all instances of word_list concatenated
        with each comma-separated elaboration_item
        in the elaboration.
      - If it is an “elaboration word_list”, add
        the same, except for reversing the order of
        the concatenation.
```

B.7 MatchStringSet

```
MatchStringSet(S,L):
1. Initialize a list WRL of pointers to WordNet
   results to be empty.
2. For each of the strings in the set:
   a) Match it with WordNet, using L’s part of
      speech.  If L is a noun or an verb, return all
      the hypernyms.  If L is an adverb or adverb,
      return all synonyms
   b) If a match occurred, place it at the end of
      WRL.
3. If WRL is empty, there were no matches, so we
   return without adding anything to the ontology.
   We write a message to the screen saying that
   the match failed for this string-set and this
   lexeme.
4. If WRL holds only one result, choose the synset
   which is the first in this list of synset.  Call
   it A.  Go to 7.
5. Otherwise, Call GetMaxHits(WRL).  Call the result
   A.
6. Is A empty? This would mean that there were no
   entries in WRL that had the same synset.
   a) A is empty:
      i) Call MatchBackupPlan(WRL,L).
      ii) Return.
   b) A is not empty: Go to 7.
7. Call AddOntologyEntry(L,A).
```

B.8 GetMaxHits

```
GetMaxHits(WRL)
1. Initialize, to empty, an associative array that
   maps synset-representations to integers.
   This will hold the synsets and the number of hits
   which each incurred.  Call the array M.
2. Set H to WRL’s head.
3. While H is not the empty list (nil):
```

```
   a) Call CountHits(H,M).
   b) Set H to H's tail.
4. Traverse M, finding the synset that has the max
   number of hits. Call this synset A.
5. If A's hit-count is 1, it means that all synsets
   matched exactly once, and we have no basis for
   choosing any of them based on maximum number of
   hits.  Therefore, return the empty synset.
6. Otherwise, return A.
```

B.9 CountHits

```
CountHits(SS,M)
1. Set H to SS's head.
2. While H is not the empty list (nil):
   a) Set A to H's synset-representation.
   b) Is A in M? If so, increment A's associated
      hit-count by 1, and go to d).
   c) If A is not in M, add it with the hit-count 1.
   d) Set H to H's tail.
```

B.10 MatchBackupPlan

```
MatchBackupPlan(WRL,L)
1. What is L's part of speech?
   a) Adjective or adverb: Call
      MatchAdjectiveOrAdverb(WRL,L).
   b) Noun or verb: Call MatchNounOrVerb(WRL,L).
```

B.11 MatchAdjectiveOrAdverb

```
MatchAdjectiveOrAdverb(WRL,L):
1. Set H to WRL's head.
2. Take H's synset and call it A.
3. Call AddOntologyEntry(L,A).
```

B.12 MatchNounOrVerb

```
MatchNounOrVerb(WRL,L)
1. Call GetLeastCommonAncestor(WRL), and call the
   result A.
   a) If A is in WordNet, go to 2.
   b) If A is empty, set A to be the
      synset of the first item in WRL. Go to 2.
2. Call AddOntologyEntry(L,A).
```

B.13 GetLeastCommonAncestor(WRL)

```
GetLeastCommonAncestor(WRL):
1. Create an empty ontology.  Call it O.
2. For each of the items SS in WRL:
   a) Conduct a hyper-nym search on SS in WordNet.
      Call the result Hyp.
   b) Create an entry cluster EC based on SS.
```

```
   c) Add EC to O, if not already there.
   d) Call AddHypernyms(O,EC,SS).
3. For each of the items SS in WRL:
   a) Get the entry cluster from O based on SS.
   b) Call TraceAncestors(O,EC,SS,0).  This will add
      a fake ontology entry based on SS to each of
      SS's ancestors.  The English gloss of each
      ontology entry will hold the smallest distance
      from SS upwards in the hierarchy to the given
      entry cluster.
4. Get from O the set of entry clusters that have
   exactly as many ontology entries as there are
   entries in WRL.
5. If this set is empty, no single entry cluster was
   the common ancestor of them all.  Therefore,
   return the empty synset.
6. Get from the set the entry cluster EC with the
   smallest sum of "English glosses", i.e.,
   distance from the bottom-synsets.
7. Perform a WordNet search for EC and its hypernyms.
   Call the result SS.
8. If SS is a unique beginner, we don't want it, so
   return the empty set.
9. Otherwise, return SS.
```

B.14 TraceAncestors

```
TraceAncestors(O,EC,SS,Distance)
1. See if SS is already in EC.
   a) If it is:
      i) Call the ontology entry OE.
      ii) See if OE's "english gloss" is smaller
          than Distance. If it isn't, replace it with
          Distance.
   b) If it isn't, create an ontology entry OE with
      English gloss "distance" and Hebrew lexeme
      based on SS's synset-string, and ID based on
      SS's synset-string plus the distance.  Add OE
      to EC.
2. If it isn't, create an entry cluster based on SS,
   call it EC, create an ontology entry as described
   in 1b) above, add it to EC, add EC to O.
3. Perform a WordNet search for SS's hypernyms.  Call
   the result Hyp.
4. Set H to Hyp's head.
5. While H is not the empty list (nil):
   a) Create an entry cluster based on H's synset.
      Call it EC.
   b) See if EC is in the ontology O.  If it isn't,
      add it.
   c) Call TraceAncestors(O,EC,H,Distance+1).
   d) Set H = H's tail.
```

B.15 AddOntologyEntry

```
AddOntologyEntry(L,A):
1. What is L's part of speech?
   a) Adjective or adverb: Call
      AddOEAdjectiveAdverb(L,A).
   b) Noun or verb: Call AddOENounVerb(L,A).
```

B.16 AddOEAdjectiveAdverb

```
AddOEAdjectiveAdverb(L,A).
1. Create an ontology entry based on L.  Call it OE.
2. Find out whether A is represented by an entry
   cluster already:
   a) If it is, get the entry cluster and add OE to
      it.  Return.
   b) If it isn't, proceed with 3. below.
3. Find out which entry cluster to place it beneath:
   What is L's part of speech?
   a) Adjective: Find the entry cluster based on
      "attribute".  Call its id IDUp.
   b) Adverb: Find the entry cluster based on
      "manner".  Call its id IDUp.
4. Create an entry cluster based on A.  Call it EC.
5. Add IDUp to EC's list of up-ids.
6. Add OE to EC.
```

B.17 AddOENounVerb

```
AddOENounVerb(L,A).
1. Create an ontology entry based on L.  Call it OE.
2. Find out whether A is represented by an entry
   cluster already:
   a) If it is, get the entry cluster and add OE to
      it.  Return.
   b) If it isn't, proceed with 3. below.
3. Perform a WordNet search for all the hypernyms
   of A.  This will generate a synset which has a
   pointer upwards to a linked list of its immediate
   ancestors, which then again have pointers upwards
   to their immediate ancestors.  Call this synset
   SS.
3. Take SS, and add it to the ontology as an
   entry cluster.  Call it EC.  Add OE to it.
4. Call AddHypernyms(Ontology,EC,SS->hypernyms),
   where Ontology is the ontology we are building.
```

B.18 AddHypernyms

```
AddHypernyms(O,EC,SS)
1. Set H = SS's head.
2. While H is not the end of the list:
   a) Is H present as an entry cluster?
      i) If yes, go to b).
```

```
      ii) If no:
          1) Add an entry cluster based on H. Call it
             EC2.
          2) Add EC2's id to EC's set of up-id's.
          3) Call AddHypernyms recursively with EC2
             and H's hypernyms as the argumens.
   b) Set H = H's tail.
```

C The product

C.1 Introduction

Below is a representation of the ontology that is created by our method. It is printed in tree-form, with **Entity** at the top.

C.2 Ontology

```
EntryCluster: { 'Entity'
    EntryCluster: { '{ change }-84849'
       OE: { lexeme = 'HPK[' gloss = 'change' }
       EntryCluster: { '{ discolor, discolour, colour, color }-224327'
          EntryCluster: { '{ green }-417319'
             OE: { lexeme = 'DC>[' gloss = 'green' }
       EntryCluster: { '{ change_magnitude }-134465'
          EntryCluster: { '{ increase }-124017'
             EntryCluster: { '{ grow }-183370'
                EntryCluster: { '{ boom, prosper, thrive, get_ahead, flourish, expand }-248526'
                   OE: { lexeme = 'FKL[' gloss = 'prosper' }
                EntryCluster: { '{ develop }-200379'
                   EntryCluster: { '{ grow }-184258'
                      EntryCluster: { '{ shoot, spud, germinate, pullulate, bourgeon, burgeon_forth, sprout }-286419'
                         OE: { lexeme = 'YMX[' gloss = 'sprout' }
       EntryCluster: { '{ change_state, turn }-114704'
          EntryCluster: { '{ die, decease, perish, go, exit, pass_away, expire, pass }-287243'
             OE: { lexeme = 'MWT[' gloss = 'die' }
          EntryCluster: { '{ dress, clothe, enclothe, garb, raiment, tog, garment, habilitate, fit_out, apparel }-38422'
             OE: { lexeme = 'LBC[' gloss = 'clothe' }
       EntryCluster: { '{ change_shape, change_form, deform }-110525'
          EntryCluster: { '{ unfold, stretch, stretch_out, extend }-1617547'
             OE: { lexeme = 'CLX[' gloss = 'stretch_out' }
    EntryCluster: { '{ determine, find, find_out, ascertain }-726714'
       OE: { lexeme = 'CJT[' gloss = 'determine' }
    EntryCluster: { '{ consume, ingest, take_in, take, have }-913965'
       EntryCluster: { '{ eat }-923270'
          OE: { lexeme = '>KL[' gloss = 'eat' }
       EntryCluster: { '{ eat }-921744'
          EntryCluster: { '{ eat }-923270'
             OE: { lexeme = '>KL[' gloss = 'eat' }
    EntryCluster: { '{ psychological_feature }-16840'
       EntryCluster: { '{ cognition, knowledge, noesis }-17218'
          OE: { lexeme = 'D<T/' gloss = 'cognition' }
          EntryCluster: { '{ content, cognitive_content, mental_object }-4977584'
             EntryCluster: { '{ idea, thought }-4997818'
                EntryCluster: { '{ concept, conception, construct }-4999203'
                   EntryCluster: { '{ property, attribute, dimension }-5010335'
                      EntryCluster: { '{ third, 3rd, tertiary }-2137353'
                         OE: { lexeme = 'CLJCJ/' gloss = 'third' }
                      EntryCluster: { '{ sixth, 6th }-2137882'
                         OE: { lexeme = 'CCJ/' gloss = 'sixth' }
                      EntryCluster: { '{ seventh, 7th }-2138007'
                         OE: { lexeme = 'CBJ<J/' gloss = 'seventh' }
                      EntryCluster: { '{ bare, au_naturel(p), naked, nude }-432444'
                         OE: { lexeme = '<JRM/' gloss = 'naked' }
                         OE: { lexeme = '<RWM/' gloss = 'naked' }
                      EntryCluster: { '{ male }-1432909'
                         OE: { lexeme = 'ZKR=/' gloss = 'male' }
                      EntryCluster: { '{ great }-1347355'
                         OE: { lexeme = 'GDWL/' gloss = 'great' }
                      EntryCluster: { '{ astute, sharp, shrewd }-412961'
                         OE: { lexeme = '<RWM=/' gloss = 'shrewd' }
                      EntryCluster: { '{ young, immature }-1603567'
                         OE: { lexeme = 'QVN/' gloss = 'young' }
                      EntryCluster: { '{ second, 2nd, 2d }-2137210'
                         OE: { lexeme = 'CNJ/' gloss = 'second' }
                      EntryCluster: { '{ evil, harmful, injurious }-226511'
                         OE: { lexeme = 'R</' gloss = 'bad' }
                      EntryCluster: { '{ fourth, 4th, quaternary }-2137489'
                         OE: { lexeme = 'RBJ<J/' gloss = 'fourth' }
                      EntryCluster: { '{ alive(p) }-98396'
                         OE: { lexeme = 'XJ/' gloss = 'alive' }
                      EntryCluster: { '{ fifth, 5th }-2137758'
                         OE: { lexeme = 'XMJCJ/' gloss = 'fifth' }
                      EntryCluster: { '{ female }-1433765'
                         OE: { lexeme = 'NQBH/' gloss = 'female' }
                      EntryCluster: { '{ insignificant, trivial }-599116'
                         OE: { lexeme = 'QVN/' gloss = 'insignificant' }
                      EntryCluster: { '{ good }-1088300'
                         OE: { lexeme = 'VWB/' gloss = 'good' }
                   EntryCluster: { '{ whole }-5024209'
                      OE: { lexeme = 'KL/' gloss = 'whole' }
```

```
                  EntryCluster: { '{ category }-5001856'
                     EntryCluster: { '{ kind, sort, form, variety }-5002061'
                        OE: { lexeme = 'MJN/' gloss = 'kind' }
               EntryCluster: { '{ belief }-5079811'
                  EntryCluster: { '{ spiritual_being, supernatural_being }-7800273'
                     EntryCluster: { '{ God, Supreme_Being }-7828973'
                        OE: { lexeme = '>LHJM/' gloss = 'god' }
                        OE: { lexeme = '>LHJM/' gloss = 'God' }
               EntryCluster: { '{ representation, mental_representation, internal_representation }-5067209'
                  EntryCluster: { '{ image, mental_image }-5068399'
                     OE: { lexeme = 'YLM/' gloss = 'image' }
            EntryCluster: { '{ information }-4982819'
               EntryCluster: { '{ evidence, grounds }-4989126'
                  EntryCluster: { '{ symptom }-12072666'
                     EntryCluster: { '{ pain, hurting }-12093415'
                        OE: { lexeme = '<YBWN/' gloss = 'pain' }
            EntryCluster: { '{ equivalent }-4887737'
               EntryCluster: { '{ counterpart, opposite_number, vis-a-vis }-4887971'
                  OE: { lexeme = 'NGD/' gloss = 'counterpart' }
            EntryCluster: { '{ ability, power }-4830304'
               EntryCluster: { '{ faculty, mental_faculty, module }-4851828'
                  EntryCluster: { '{ sense, sensation, sentience, sentiency, sensory_faculty }-4853239'
                     EntryCluster: { '{ modality, sense_modality, sensory_system }-4853520'
                        EntryCluster: { '{ sight, vision, visual_sense, visual_modality }-4855059'
                           OE: { lexeme = 'MR>H/' gloss = 'appearance' }
                     EntryCluster: { '{ sensitivity, sensitiveness, sensibility }-4854014'
                        EntryCluster: { '{ exteroception }-4854581'
                           EntryCluster: { '{ sight, vision, visual_sense, visual_modality }-4855059'
                              OE: { lexeme = 'MR>H/' gloss = 'appearance' }
         EntryCluster: { '{ motivation, motive, need }-17684'
            EntryCluster: { '{ rational_motive }-7542709'
               EntryCluster: { '{ reason, ground }-7542887'
                  OE: { lexeme = '<BWR/' gloss = 'ground' }
         EntryCluster: { '{ feeling }-18103'
            EntryCluster: { '{ desire }-6222835'
               OE: { lexeme = 'T>WH/' gloss = 'desire' }
               OE: { lexeme = 'TCWQH/' gloss = 'desire' }
            EntryCluster: { '{ emotion }-6219674'
               EntryCluster: { '{ anger, choler, ire }-6248644'
                  OE: { lexeme = '>P/' gloss = 'anger' }
      EntryCluster: { '{ abstraction }-16993'
         EntryCluster: { '{ attribute }-23392'
            EntryCluster: { '{ property }-4239634'
               EntryCluster: { '{ manner, mode, style, way, fashion }-4249815'
                  EntryCluster: { '{ now, at_present }-49038'
                     OE: { lexeme = '<TH' gloss = 'now' }
                  EntryCluster: { '{ besides, too, also, likewise, as_well }-47251'
                     OE: { lexeme = 'GM' gloss = 'also' }
                  EntryCluster: { '{ then, so, and_so, and_then }-117038'
                     OE: { lexeme = 'CM' gloss = 'then' }
                  EntryCluster: { '{ thus, thusly, so }-120452'
                     OE: { lexeme = 'KN' gloss = 'so' }
               EntryCluster: { '{ magnitude }-4372088'
                  EntryCluster: { '{ amount }-4384342'
                     EntryCluster: { '{ quantity }-4384565'
                        EntryCluster: { '{ abundance, copiousness, teemingness }-4388714'
                           EntryCluster: { '{ profusion, profuseness, richness, cornucopia }-4389280'
                              EntryCluster: { '{ greenness, verdancy, verdure }-4389776'
                                 OE: { lexeme = 'JRQ/' gloss = 'greenness' }
               EntryCluster: { '{ spatial_property, spatiality }-4350089'
                  EntryCluster: { '{ shape, form, configuration, contour, conformation }-4350978'
                     OE: { lexeme = 'DMWT/' gloss = 'shape' }
            EntryCluster: { '{ quality }-4099891'
               EntryCluster: { '{ good, goodness }-4409046'
                  EntryCluster: { '{ benefit, welfare }-4409413'
                     EntryCluster: { '{ sake, interest }-4409774'
                        OE: { lexeme = '<BWR/' gloss = 'interest' }
               EntryCluster: { '{ appearance, visual_aspect }-4065243'
                  EntryCluster: { '{ countenance, visage }-4070044'
                     EntryCluster: { '{ expression, look, aspect, facial_expression, face }-4070233'
                        OE: { lexeme = '<JN/' gloss = 'look' }
               EntryCluster: { '{ sameness }-4112810'
                  EntryCluster: { '{ similarity }-4113610'
                     EntryCluster: { '{ likeness, alikeness, similitude }-4114547'
                        OE: { lexeme = 'DMWT/' gloss = 'likeness' }
            EntryCluster: { '{ shape, form }-19434'
               EntryCluster: { '{ round_shape }-11699933'
                  EntryCluster: { '{ loop }-11708602'
                     EntryCluster: { '{ belt }-2460164'
                        OE: { lexeme = 'XGWRH/' gloss = 'belt' }
         EntryCluster: { '{ measure, quantity, amount, quantum }-24936'
            EntryCluster: { '{ fundamental_quantity, fundamental_measure }-11450237'
               EntryCluster: { '{ time_period, period_of_time, period }-12786206'
                  EntryCluster: { '{ season, time_of_year }-12896721'
                     EntryCluster: { '{ spring, springtime }-12897329'
                        OE: { lexeme = '<JN/' gloss = 'spring' }
                  EntryCluster: { '{ day, daytime, daylight }-12832221'
                     EntryCluster: { '{ evening, eve, eventide }-12833627'
                        OE: { lexeme = '<RB/' gloss = 'evening' }
                        OE: { lexeme = '<RB/' gloss = 'evening' }
                  EntryCluster: { '{ year, twelvemonth, yr }-12867473'
                     OE: { lexeme = 'CNH/' gloss = 'year' }
                  EntryCluster: { '{ morning, morn, morning_time, forenoon }-12832553'
                     OE: { lexeme = 'BQR=/' gloss = 'morning' }
                  EntryCluster: { '{ night, nighttime, dark }-12834176'
                     OE: { lexeme = 'LJLH/' gloss = 'night' }
                  EntryCluster: { '{ life, lifetime, lifespan }-12810635'
                     OE: { lexeme = 'XJJM/' gloss = 'life' }
            EntryCluster: { '{ definite_quantity }-11450469'
               EntryCluster: { '{ number }-11455258'
                  EntryCluster: { '{ integer, whole_number }-11584525'
                     EntryCluster: { '{ digit, figure }-11594993'
                        EntryCluster: { '{ four, 4, IV, tetrad, quatern, quaternion, quaternary, quaternity, quartet, quadru-
plet, foursome, Little_Joe }-11597497'
                           OE: { lexeme = '>RB</' gloss = 'four' }
                        EntryCluster: { '{ two, 2, II, deuce }-11596692'
                           OE: { lexeme = 'CNJM/' gloss = 'two' }
```

```
                  EntryCluster: { '{ one, 1, I, ace, single, unity }-11596183'
                     OE: { lexeme = '>XD/' gloss = 'one' }
         EntryCluster: { '{ time_unit, unit_of_time }-12823670'
            EntryCluster: { '{ day, twenty-four_hours, solar_day, mean_solar_day }-12824116'
               OE: { lexeme = 'JWM/' gloss = 'day' }
         EntryCluster: { '{ linear_measure, long_measure }-11470439'
            EntryCluster: { '{ linear_unit }-11473734'
               EntryCluster: { '{ foot, ft }-11513651'
                  OE: { lexeme = 'P<M/' gloss = 'foot' }
      EntryCluster: { '{ relation }-23704'
         EntryCluster: { '{ social_relation }-24288'
            EntryCluster: { '{ communication }-24503'
               EntryCluster: { '{ indication, indicant }-5681175'
                  EntryCluster: { '{ evidence }-5560202'
                     EntryCluster: { '{ clue, clew, cue }-5560449'
                        EntryCluster: { '{ footprint, footmark, step }-5561033'
                           OE: { lexeme = '<QB=/' gloss = 'footprint' }
                        EntryCluster: { '{ sign, mark }-5561275'
                           OE: { lexeme = '>WT/' gloss = 'sign' }
                  EntryCluster: { '{ mark }-5682502'
                     EntryCluster: { '{ footprint, footmark, step }-5561033'
                        OE: { lexeme = '<QB=/' gloss = 'footprint' }
               EntryCluster: { '{ visual_communication }-5739850'
                  EntryCluster: { '{ artwork, art, graphics, nontextual_matter }-5851743'
                     EntryCluster: { '{ illustration }-5852022'
                        EntryCluster: { '{ figure, fig }-5852382'
                           OE: { lexeme = 'T>NH/' gloss = 'fig' }
         EntryCluster: { '{ position, spatial_relation }-4359534'
            EntryCluster: { '{ placement, arrangement }-4360344'
               EntryCluster: { '{ spacing, spatial_arrangement }-4366815'
                  EntryCluster: { '{ distance }-4367296'
                     OE: { lexeme = 'BJN/' gloss = 'distance' }
            EntryCluster: { '{ direction }-11667742'
               EntryCluster: { '{ compass_point, point }-11670233'
                  EntryCluster: { '{ cardinal_compass_point }-11670928'
                     EntryCluster: { '{ east, due_east, E }-11672255'
                        OE: { lexeme = 'QDM/' gloss = 'east' }
         EntryCluster: { '{ part, portion, component_part, component }-11652310'
            EntryCluster: { '{ language_unit, linguistic_unit }-5295988'
               EntryCluster: { '{ name }-5332857'
                  OE: { lexeme = 'CM/' gloss = 'name' }
   EntryCluster: { '{ time }-19843'
      EntryCluster: { '{ eternity, infinity, forever }-12903988'
         OE: { lexeme = '<WLM/' gloss = 'eternity' }
         OE: { lexeme = '<WLM/' gloss = 'forever' }
      EntryCluster: { '{ past, past_times, yesteryear, yore }-12792698'
         EntryCluster: { '{ history }-12793186'
            OE: { lexeme = 'TWLDWT/' gloss = 'history' }
EntryCluster: { '{ control, command }-1919122'
   EntryCluster: { '{ govern, rule }-2032771'
      OE: { lexeme = 'MCL[' gloss = 'govern' }
      OE: { lexeme = 'RDH[' gloss = 'rule' }
EntryCluster: { '{ work }-1896035'
   EntryCluster: { '{ putter, mess_around, potter, tinker, monkey, monkey_around, muck_about, muck_around }-1164977'
      OE: { lexeme = 'JYR[' gloss = 'potter' }
EntryCluster: { '{ group, grouping }-22634'
   EntryCluster: { '{ social_group }-6643140'
      EntryCluster: { '{ organization, organisation }-6683510'
         EntryCluster: { '{ institution, establishment }-6689622'
            EntryCluster: { '{ company }-6693875'
               OE: { lexeme = '<MD/' gloss = 'company' }
         EntryCluster: { '{ unit, social_unit }-6787448'
            EntryCluster: { '{ military_unit, military_force, military_group, force }-6791040'
               OE: { lexeme = 'M>D/' gloss = 'force' }
               EntryCluster: { '{ army_unit }-6788397'
                  EntryCluster: { '{ division }-6796169'
                     OE: { lexeme = 'R>C/' gloss = 'division' }
            EntryCluster: { '{ administrative_unit, administrative_body }-6710350'
               EntryCluster: { '{ committee, commission }-6882991'
                  EntryCluster: { '{ board }-6881462'
                     OE: { lexeme = 'YL</' gloss = 'board' }
      EntryCluster: { '{ gathering, assemblage }-6661700'
         EntryCluster: { '{ meeting }-6875317'
            OE: { lexeme = 'MW<D/' gloss = 'meeting' }
   EntryCluster: { '{ world, human_race, humanity, humankind, human_beings, humans, mankind, man }-6637467'
      OE: { lexeme = '>DMH/' gloss = 'world' }
   EntryCluster: { '{ community, biotic_community }-6637063'
      EntryCluster: { '{ biome }-6637260'
         EntryCluster: { '{ desert }-7003112'
            OE: { lexeme = 'THW/' gloss = 'desert' }
   EntryCluster: { '{ collection, aggregation, accumulation, assemblage }-6643594'
      OE: { lexeme = 'MQWH/' gloss = 'accumulation' }
EntryCluster: { '{ perceive, comprehend }-1659601'
   EntryCluster: { '{ hear }-1710482'
      OE: { lexeme = 'CM<[' gloss = 'hear' }
   EntryCluster: { '{ see }-1677760'
      OE: { lexeme = 'R>H[' gloss = 'see' }
EntryCluster: { '{ change, alter }-97545'
   EntryCluster: { '{ shape, form }-111569'
      OE: { lexeme = 'JYR[' gloss = 'form' }
   EntryCluster: { '{ better, improve, amend, ameliorate, meliorate }-163159'
      EntryCluster: { '{ repair, mend, fix, bushel, doctor, furbish_up, restore, touch_on }-207536'
         OE: { lexeme = '<FH[' gloss = 'fix' }
   EntryCluster: { '{ end, terminate }-282818'
      EntryCluster: { '{ complete, finish }-386289'
         OE: { lexeme = 'KLH[' gloss = 'complete' }
   EntryCluster: { '{ translate, transform }-307159'
      EntryCluster: { '{ prepare }-168805'
         EntryCluster: { '{ preserve, keep }-168152'
            OE: { lexeme = 'CMR[' gloss = 'guard' }
EntryCluster: { '{ worry }-1388281'
   EntryCluster: { '{ fear }-1399136'
      OE: { lexeme = 'JR>[' gloss = 'be_afraid' }
EntryCluster: { '{ discontinue, stop, cease, give_up, quit, lay_off }-2110398'
   OE: { lexeme = 'CBT[' gloss = 'cease' }
EntryCluster: { '{ entity, physical_thing }-1742'
   EntryCluster: { '{ location }-19046'
```

```
         EntryCluster: { ’{ region, part }-7110412’
            EntryCluster: { ’{ extremity }-7053550’
               EntryCluster: { ’{ boundary, bound, bounds }-7008686’
                  EntryCluster: { ’{ surface }-7134986’
                     EntryCluster: { ’{ celestial_sphere, sphere, empyrean, firmament, heavens, vault_of_heaven, welkin }-
7015404’
                        OE: { lexeme = ’RQJ</’ gloss = ’firmament’ }
                     EntryCluster: { ’{ side, face }-7007293’
                        EntryCluster: { ’{ bottom, underside, undersurface }-7007795’
                           EntryCluster: { ’{ heel }-3060482’
                              OE: { lexeme = ’<QB=/’ gloss = ’heel’ }
                  EntryCluster: { ’{ end }-7051434’
                     OE: { lexeme = ’KNP/’ gloss = ’end’ }
               EntryCluster: { ’{ inside, interior }-7072180’
                  EntryCluster: { ’{ midst, thick }-7072567’
                     OE: { lexeme = ’TWK/’ gloss = ’midst’ }
         EntryCluster: { ’{ region }-7111224’
            EntryCluster: { ’{ district, territory }-7040158’
               OE: { lexeme = ’>DMH/’ gloss = ’territory’ }
               EntryCluster: { ’{ administrative_district, administrative_division, territorial_division }-6992023’
                  EntryCluster: { ’{ country, state, land }-7034213’
                     OE: { lexeme = ’>RY/’ gloss = ’country’ }
            EntryCluster: { ’{ West, occident }-7152976’
               OE: { lexeme = ’JM/’ gloss = ’west’ }
            EntryCluster: { ’{ geographical_area, geographic_area, geographical_region, geographic_region }-7058546’
               EntryCluster: { ’{ tract, piece_of_land, piece_of_ground, parcel_of_land, parcel }-7144938’
                  EntryCluster: { ’{ plot, plot_of_ground, patch }-7146192’
                     EntryCluster: { ’{ garden }-2976476’
                        OE: { lexeme = ’GN/’ gloss = ’garden’ }
                  EntryCluster: { ’{ field }-7054636’
                     OE: { lexeme = ’FDH/’ gloss = ’field’ }
               EntryCluster: { ’{ desert }-7003112’
                  OE: { lexeme = ’THW/’ gloss = ’desert’ }
         EntryCluster: { ’{ point }-7101501’
            EntryCluster: { ’{ topographic_point, place, spot }-7137864’
               OE: { lexeme = ’MQWM/’ gloss = ’place’ }
      EntryCluster: { ’{ imaginary_place }-7070716’
         EntryCluster: { ’{ Heaven }-7064858’
            OE: { lexeme = ’CMJM/’ gloss = ’heaven’ }
      EntryCluster: { ’{ object, physical_object }-13067’
         EntryCluster: { ’{ natural_object }-13738’
            EntryCluster: { ’{ plant_part }-11022946’
               EntryCluster: { ’{ plant_organ }-11023465’
                  EntryCluster: { ’{ leaf, leafage, foliage }-11083106’
                     OE: { lexeme = ’<LH=/’ gloss = ’leafage’ }
                  EntryCluster: { ’{ reproductive_structure }-9623850’
                     EntryCluster: { ’{ fruit }-11066234’
                        OE: { lexeme = ’PRJ/’ gloss = ’fruit’ }
            EntryCluster: { ’{ rock, stone }-7732649’
               OE: { lexeme = ’>BN/’ gloss = ’stone’ }
            EntryCluster: { ’{ covering, natural_covering, cover }-7607016’
               EntryCluster: { ’{ body_covering }-4483296’
                  EntryCluster: { ’{ skin, tegument, cutis }-4483805’
                     OE: { lexeme = ’<WR=/’ gloss = ’skin’ }
            EntryCluster: { ’{ body, organic_structure, physical_structure }-4464337’
               OE: { lexeme = ’BFR/’ gloss = ’body’ }
            EntryCluster: { ’{ celestial_body, heavenly_body }-7593134’
               EntryCluster: { ’{ star }-7754660’
                  OE: { lexeme = ’KWKB/’ gloss = ’star’ }
         EntryCluster: { ’{ land, dry_land, earth, ground, solid_ground, terra_firma }-7667106’
            OE: { lexeme = ’JBCH/’ gloss = ’dry_land’ }
         EntryCluster: { ’{ living_thing, animate_thing }-2956’
            EntryCluster: { ’{ organism, being }-3135’
               EntryCluster: { ’{ animal, animate_being, beast, brute, creature, fauna }-11413’
                  OE: { lexeme = ’XJH/’ gloss = ’beast’ }
                  EntryCluster: { ’{ marine_animal, sea_animal }-1013516’
                     OE: { lexeme = ’CRY/’ gloss = ’marine_animal’ }
                  EntryCluster: { ’{ chordate }-1145284’
                     EntryCluster: { ’{ vertebrate, craniate }-1150487’
                        EntryCluster: { ’{ bird }-1181338’
                           OE: { lexeme = ’<WP/’ gloss = ’bird’ }
                        EntryCluster: { ’{ mammal }-1534425’
                           EntryCluster: { ’{ placental, placental_mammal, eutherian, eutherian_mammal }-1558951’
                              EntryCluster: { ’{ primate }-2128707’
                                 EntryCluster: { ’{ hominid }-2130483’
                                    EntryCluster: { ’{ homo, man, human_being, human }-2130996’
                                       OE: { lexeme = ’>DM/’ gloss = ’human_being’ }
                                       OE: { lexeme = ’>JC/’ gloss = ’human_being’ }
                              EntryCluster: { ’{ ungulate, hoofed_mammal }-2033070’
                                 EntryCluster: { ’{ even-toed_ungulate, artiodactyl, artiodactyl_mammal }-2055122’
                                    EntryCluster: { ’{ ruminant }-2059415’
                                       EntryCluster: { ’{ bovid }-2061354’
                                          EntryCluster: { ’{ bovine }-2062333’
                                             EntryCluster: { ’{ cattle, cows, kine, oxen, Bos_taurus }-2062748’
                                                OE: { lexeme = ’BHMH/’ gloss = ’cattle’ }
                        EntryCluster: { ’{ aquatic_vertebrate }-1152246’
                           EntryCluster: { ’{ fish }-2169634’
                              OE: { lexeme = ’DGH/’ gloss = ’fish’ }
                        EntryCluster: { ’{ reptile, reptilian }-1337209’
                           EntryCluster: { ’{ diapsid, diapsid_reptile }-1337909’
                              EntryCluster: { ’{ snake, serpent, ophidian }-1402291’
                                 OE: { lexeme = ’NXC/’ gloss = ’serpent’ }
                     EntryCluster: { ’{ invertebrate }-1576672’
                        EntryCluster: { ’{ arthropod }-1442445’
                           EntryCluster: { ’{ insect }-1826146’
                              OE: { lexeme = ’<WP/’ gloss = ’insect’ }
                        EntryCluster: { ’{ creepy-crawly }-1010503’
                           OE: { lexeme = ’RMF/’ gloss = ’creepy-crawly’ }
                  EntryCluster: { ’{ plant, flora, plant_life }-12420’
                     EntryCluster: { ’{ vascular_plant, tracheophyte }-11019945’
                        EntryCluster: { ’{ weed }-11021436’
                           EntryCluster: { ’{ thistle }-9886927’
                              OE: { lexeme = ’DRDR/’ gloss = ’thistle’ }
                        EntryCluster: { ’{ woody_plant, ligneous_plant }-11037159’
                           EntryCluster: { ’{ tree }-11037900’
                              OE: { lexeme = ’<Y/’ gloss = ’tree’ }
                           EntryCluster: { ’{ shrub, bush }-11045834’
```

```
                  OE: { lexeme = 'FJX/' gloss = 'shrub' }
               EntryCluster: { '{ herb, herbaceous_plant }-10146714'
                  OE: { lexeme = '<FB/' gloss = 'herb' }
                  EntryCluster: { '{ gramineous_plant, graminaceous_plant }-10044245'
                     EntryCluster: { '{ grass }-10044515'
                        OE: { lexeme = 'DC>/' gloss = 'grass' }
         EntryCluster: { '{ person, individual, someone, somebody, mortal, human, soul }-5303'
            OE: { lexeme = '>JC/' gloss = 'somebody' }
            EntryCluster: { '{ worker }-7911996'
               EntryCluster: { '{ assistant, helper, help, supporter }-8062152'
                  OE: { lexeme = '<ZR/' gloss = 'help' }
            EntryCluster: { '{ female, female_person }-7901005'
               EntryCluster: { '{ woman, adult_female }-8828291'
                  OE: { lexeme = '>CH/' gloss = 'woman' }
            EntryCluster: { '{ relative, relation }-8394508'
               EntryCluster: { '{ ancestor, ascendant, ascendent, antecedent, root }-8045071'
                  EntryCluster: { '{ progenitor, primogenitor }-8309449'
                     EntryCluster: { '{ genitor }-8309351'
                        EntryCluster: { '{ parent }-8522773'
                           EntryCluster: { '{ father, male_parent, begetter }-8272054'
                              OE: { lexeme = '>B/' gloss = 'father' }
                           EntryCluster: { '{ mother, female_parent }-8471061'
                              OE: { lexeme = '>M/' gloss = 'mother' }
               EntryCluster: { '{ offspring, progeny, issue }-8501994'
                  OE: { lexeme = 'TWLDWT/' gloss = 'offspring' }
                  OE: { lexeme = 'ZR</' gloss = 'offspring' }
                  EntryCluster: { '{ child, kid }-8144624'
                     EntryCluster: { '{ baby, babe, infant }-8071651'
                        EntryCluster: { '{ cherub }-8142520'
                           OE: { lexeme = 'KRWB/' gloss = 'cherub' }
                     EntryCluster: { '{ male_offspring, man-child }-8435310'
                        EntryCluster: { '{ son, boy }-8701001'
                           OE: { lexeme = 'BN/' gloss = 'son' }
               EntryCluster: { '{ male, male_person }-7904937'
                  EntryCluster: { '{ man, adult_male }-8436072'
                     OE: { lexeme = '>JC/' gloss = 'man' }
                  EntryCluster: { '{ male_offspring, man-child }-8435310'
                     EntryCluster: { '{ son, boy }-8701001'
                        OE: { lexeme = 'BN/' gloss = 'son' }
               EntryCluster: { '{ leader }-7904081'
                  EntryCluster: { '{ head, chief, top_dog }-8337045'
                     OE: { lexeme = 'R>C/' gloss = 'chief' }
               EntryCluster: { '{ adult, grownup }-7889306'
                  EntryCluster: { '{ host }-8356757'
                     OE: { lexeme = 'YB>/' gloss = 'host' }
         EntryCluster: { '{ mutant, mutation, variation, sport }-8478006'
            EntryCluster: { '{ freak, monster, monstrosity, lusus_naturae }-8296452'
               EntryCluster: { '{ leviathan }-8410944'
                  OE: { lexeme = 'TNJN/' gloss = 'leviathan' }
EntryCluster: { '{ whole, whole_thing, unit }-2664'
   EntryCluster: { '{ artifact, artefact }-15787'
      EntryCluster: { '{ way }-3976508'
         EntryCluster: { '{ road, route }-3570522'
            OE: { lexeme = 'DRK/' gloss = 'road' }
      EntryCluster: { '{ instrumentality, instrumentation }-3115433'
         EntryCluster: { '{ device }-2771586'
            EntryCluster: { '{ source_of_illumination }-3715642'
               EntryCluster: { '{ lamp }-3165939'
                  OE: { lexeme = 'M>WR/' gloss = 'lamp' }
               EntryCluster: { '{ instrument }-3115045'
                  EntryCluster: { '{ weapon, arm, weapon_system }-3977167'
                     EntryCluster: { '{ knife }-3156455'
                        EntryCluster: { '{ dagger, sticker }-2750404'
                           OE: { lexeme = 'XRB/' gloss = 'dagger' }
               EntryCluster: { '{ support }-3799113'
                  EntryCluster: { '{ rib }-3562342'
                     OE: { lexeme = 'YL</' gloss = 'rib' }
         EntryCluster: { '{ means }-3252432'
            EntryCluster: { '{ medium }-5272870'
               EntryCluster: { '{ telecommunication }-5286604'
                  EntryCluster: { '{ telephone, telephony }-5286825'
                     EntryCluster: { '{ call, phone_call, telephone_call }-5286980'
                        OE: { lexeme = 'QWL/' gloss = 'call' }
      EntryCluster: { '{ surface }-3801477'
         OE: { lexeme = 'PNH/' gloss = 'surface' }
      EntryCluster: { '{ block }-2480558'
         EntryCluster: { '{ anvil }-2372043'
            OE: { lexeme = 'P<M/' gloss = 'anvil' }
EntryCluster: { '{ part, portion }-3390629'
   OE: { lexeme = 'BD/' gloss = 'part' }
EntryCluster: { '{ artifact, artefact }-15787'
   EntryCluster: { '{ way }-3976508'
      EntryCluster: { '{ road, route }-3570522'
         OE: { lexeme = 'DRK/' gloss = 'road' }
   EntryCluster: { '{ instrumentality, instrumentation }-3115433'
      EntryCluster: { '{ device }-2771586'
         EntryCluster: { '{ source_of_illumination }-3715642'
            EntryCluster: { '{ lamp }-3165939'
               OE: { lexeme = 'M>WR/' gloss = 'lamp' }
         EntryCluster: { '{ instrument }-3115045'
            EntryCluster: { '{ weapon, arm, weapon_system }-3977167'
               EntryCluster: { '{ knife }-3156455'
                  EntryCluster: { '{ dagger, sticker }-2750404'
                     OE: { lexeme = 'XRB/' gloss = 'dagger' }
         EntryCluster: { '{ support }-3799113'
            EntryCluster: { '{ rib }-3562342'
               OE: { lexeme = 'YL</' gloss = 'rib' }
      EntryCluster: { '{ means }-3252432'
         EntryCluster: { '{ medium }-5272870'
            EntryCluster: { '{ telecommunication }-5286604'
               EntryCluster: { '{ telephone, telephony }-5286825'
                  EntryCluster: { '{ call, phone_call, telephone_call }-5286980'
                     OE: { lexeme = 'QWL/' gloss = 'call' }
   EntryCluster: { '{ surface }-3801477'
      OE: { lexeme = 'PNH/' gloss = 'surface' }
   EntryCluster: { '{ block }-2480558'
      EntryCluster: { '{ anvil }-2372043'
```

```
            OE: { lexeme = 'P<M/' gloss = 'anvil' }
EntryCluster: { '{ substance, matter }-14223'
   EntryCluster: { '{ food, nutrient }-15442'
      OE: { lexeme = '>KLH/' gloss = 'food' }
      OE: { lexeme = 'M>KL/' gloss = 'food' }
   EntryCluster: { '{ material, stuff }-12307888'
      EntryCluster: { '{ plant_material }-12676584'
         EntryCluster: { '{ wood }-12772693'
            OE: { lexeme = '<Y/' gloss = 'piece_of_wood' }
         EntryCluster: { '{ plant_product }-12676745'
            EntryCluster: { '{ natural_resin }-12588682'
               EntryCluster: { '{ gum_resin }-12591965'
                  EntryCluster: { '{ bdellium }-12592505'
                     OE: { lexeme = 'BDLX/' gloss = 'bdellium' }
      EntryCluster: { '{ animal_material }-12464697'
         EntryCluster: { '{ animal_product }-12415112'
            EntryCluster: { '{ animal_skin }-12467555'
               EntryCluster: { '{ leather }-12468557'
                  OE: { lexeme = '<WR=/' gloss = 'leather' }
      EntryCluster: { '{ waste, waste_material, waste_matter, waste_product }-12555950'
         EntryCluster: { '{ rubbish, trash }-12557112'
            OE: { lexeme = '<PR/' gloss = 'rubbish' }
      EntryCluster: { '{ earth, ground }-12544735'
         EntryCluster: { '{ soil, dirt }-12546373'
            EntryCluster: { '{ dust }-12541883'
               OE: { lexeme = '<PR/' gloss = 'dry_earth' }
      EntryCluster: { '{ mineral }-12379360'
         EntryCluster: { '{ quartz }-12408582'
            EntryCluster: { '{ chalcedony, calcedony }-12511918'
               EntryCluster: { '{ carnelian, cornelian }-12507384'
                  OE: { lexeme = 'CHM/' gloss = 'carnelian' }
   EntryCluster: { '{ compound, chemical_compound }-12522505'
      EntryCluster: { '{ organic_compound }-12439147'
         EntryCluster: { '{ resin, rosin }-12588413'
            EntryCluster: { '{ natural_resin }-12588682'
               EntryCluster: { '{ gum_resin }-12591965'
                  EntryCluster: { '{ bdellium }-12592505'
                     OE: { lexeme = 'BDLX/' gloss = 'bdellium' }
      EntryCluster: { '{ binary_compound }-12339673'
         EntryCluster: { '{ water, H2O }-12547246'
            OE: { lexeme = 'MJM/' gloss = 'water' }
   EntryCluster: { '{ solid }-12726340'
      EntryCluster: { '{ crystal }-12578607'
         EntryCluster: { '{ gem, gemstone, stone }-12414207'
            EntryCluster: { '{ transparent_gem }-12753595'
               EntryCluster: { '{ chalcedony, calcedony }-12511918'
                  EntryCluster: { '{ carnelian, cornelian }-12507384'
                     OE: { lexeme = 'CHM/' gloss = 'carnelian' }
      EntryCluster: { '{ food }-6278924'
         EntryCluster: { '{ baked_good }-6337373'
            EntryCluster: { '{ bread, breadstuff, staff_of_life }-6391241'
               OE: { lexeme = 'LXM/' gloss = 'bread' }
   EntryCluster: { '{ fluid }-12630223'
      EntryCluster: { '{ liquid }-12630736'
         EntryCluster: { '{ water, H2O }-12547246'
            OE: { lexeme = 'MJM/' gloss = 'water' }
   EntryCluster: { '{ body_substance }-4506531'
      EntryCluster: { '{ liquid_body_substance, bodily_fluid, body_fluid, humor, humour }-4632978'
         EntryCluster: { '{ secretion }-4638611'
            EntryCluster: { '{ perspiration, sweat, sudor }-4639616'
               OE: { lexeme = 'Z<H/' gloss = 'sweat' }
EntryCluster: { '{ body_of_water, water }-7582157'
   EntryCluster: { '{ stream, watercourse }-7758560'
      OE: { lexeme = '>D/' gloss = 'stream' }
      EntryCluster: { '{ river }-7729110'
         OE: { lexeme = 'JM/' gloss = 'river' }
         OE: { lexeme = 'NHR/' gloss = 'river' }
   EntryCluster: { '{ sea }-7740659'
      OE: { lexeme = 'JM/' gloss = 'sea' }
   EntryCluster: { '{ ocean }-7700297'
      OE: { lexeme = 'THWM/' gloss = 'ocean' }
EntryCluster: { '{ part, piece }-7708371'
   EntryCluster: { '{ body_part }-4467893'
      EntryCluster: { '{ organ }-4537584'
         EntryCluster: { '{ sense_organ, sensory_receptor, receptor }-4539203'
            EntryCluster: { '{ chemoreceptor }-4540238'
               EntryCluster: { '{ nose, olfactory_organ }-4813798'
                  OE: { lexeme = '>P/' gloss = 'nose' }
               EntryCluster: { '{ eye, oculus, optic, peeper }-4549962'
                  OE: { lexeme = '<JN/' gloss = 'eye' }
      EntryCluster: { '{ tissue }-4509873'
         EntryCluster: { '{ animal_tissue }-4510096'
            EntryCluster: { '{ connective_tissue }-4527862'
               EntryCluster: { '{ bone, os }-4512190'
                  OE: { lexeme = '<YM/' gloss = 'bone' }
               EntryCluster: { '{ skin, tegument, cutis }-4483805'
                  OE: { lexeme = '<WR=/' gloss = 'skin' }
            EntryCluster: { '{ flesh }-4510642'
               OE: { lexeme = 'BFR/' gloss = 'flesh' }
            EntryCluster: { '{ membrane, tissue_layer }-4657971'
               EntryCluster: { '{ tunic, tunica, adventitia }-4821650'
                  OE: { lexeme = 'KTNT/' gloss = 'tunic' }
      EntryCluster: { '{ abdomen, venter, stomach, belly }-4775636'
         OE: { lexeme = 'GXWN/' gloss = 'abdomen' }
      EntryCluster: { '{ external_body_part }-4471672'
         EntryCluster: { '{ extremity }-4784320'
            EntryCluster: { '{ hand, manus, mitt, paw }-4782405'
               OE: { lexeme = 'JD/' gloss = 'hand' }
         EntryCluster: { '{ head, caput }-4758769'
            OE: { lexeme = 'R>C/' gloss = 'head' }
      EntryCluster: { '{ process, outgrowth, appendage }-4696710'
         EntryCluster: { '{ plant_process, enation }-11023882'
            EntryCluster: { '{ aculeus }-11024946'
               EntryCluster: { '{ spine, thorn, prickle, pricker, sticker }-11025122'
                  OE: { lexeme = 'QWY/' gloss = 'pricker' }
      EntryCluster: { '{ underpart }-1571215'
         OE: { lexeme = 'TXT/' gloss = 'under_part' }
```

```
      EntryCluster: { ’{ causal_agent, cause, causal_agency }-4911’
         EntryCluster: { ’{ person, individual, someone, somebody, mortal, human, soul }-5303’
            OE: { lexeme = ’>JC/’ gloss = ’somebody’ }
            EntryCluster: { ’{ worker }-7911996’
               EntryCluster: { ’{ assistant, helper, help, supporter }-8062152’
                  OE: { lexeme = ’<ZR/’ gloss = ’help’ }
            EntryCluster: { ’{ female, female_person }-7901005’
               EntryCluster: { ’{ woman, adult_female }-8828291’
                  OE: { lexeme = ’>CH/’ gloss = ’woman’ }
            EntryCluster: { ’{ relative, relation }-8394508’
               EntryCluster: { ’{ ancestor, ascendant, ascendent, antecedent, root }-8045071’
                  EntryCluster: { ’{ progenitor, primogenitor }-8309449’
                     EntryCluster: { ’{ genitor }-8309351’
                        EntryCluster: { ’{ parent }-8522773’
                           EntryCluster: { ’{ father, male_parent, begetter }-8272054’
                              OE: { lexeme = ’>B/’ gloss = ’father’ }
                           EntryCluster: { ’{ mother, female_parent }-8471061’
                              OE: { lexeme = ’>M/’ gloss = ’mother’ }
               EntryCluster: { ’{ offspring, progeny, issue }-8501994’
                  OE: { lexeme = ’TWLDWT/’ gloss = ’offspring’ }
                  OE: { lexeme = ’ZR</’ gloss = ’offspring’ }
                  EntryCluster: { ’{ child, kid }-8144624’
                     EntryCluster: { ’{ baby, babe, infant }-8071651’
                        EntryCluster: { ’{ cherub }-8142520’
                           OE: { lexeme = ’KRWB/’ gloss = ’cherub’ }
                     EntryCluster: { ’{ male_offspring, man-child }-8435310’
                        EntryCluster: { ’{ son, boy }-8701001’
                           OE: { lexeme = ’BN/’ gloss = ’son’ }
            EntryCluster: { ’{ male, male_person }-7904937’
               EntryCluster: { ’{ man, adult_male }-8436072’
                  OE: { lexeme = ’>JC/’ gloss = ’man’ }
               EntryCluster: { ’{ male_offspring, man-child }-8435310’
                  EntryCluster: { ’{ son, boy }-8701001’
                     OE: { lexeme = ’BN/’ gloss = ’son’ }
            EntryCluster: { ’{ leader }-7904081’
               EntryCluster: { ’{ head, chief, top_dog }-8337045’
                  OE: { lexeme = ’R>C/’ gloss = ’chief’ }
            EntryCluster: { ’{ adult, grownup }-7889306’
               EntryCluster: { ’{ host }-8356757’
                  OE: { lexeme = ’YB>/’ gloss = ’host’ }
         EntryCluster: { ’{ vital_principle, life_principle }-8806165’
            EntryCluster: { ’{ spirit }-8711097’
               EntryCluster: { ’{ soul, psyche }-8703453’
                  OE: { lexeme = ’NPC/’ gloss = ’living_being’ }
   EntryCluster: { ’{ utter, emit, let_out, let_loose }-777192’
      OE: { lexeme = ’CLX[’ gloss = ’let_loose’ }
      EntryCluster: { ’{ shout, shout_out, cry, call, yell, scream, holler, hollo, squall }-722351’
         OE: { lexeme = ’QR>[’ gloss = ’call’ }
   EntryCluster: { ’{ state }-20595’
      EntryCluster: { ’{ being, beingness, existence }-11771798’
         OE: { lexeme = ’<YM/’ gloss = ’being’ }
      EntryCluster: { ’{ physiological_state, physiological_condition }-11834815’
         EntryCluster: { ’{ pregnancy, gestation }-11844862’
            OE: { lexeme = ’HRWN/’ gloss = ’pregnancy’ }
         EntryCluster: { ’{ sleep, slumber }-11826795’
            OE: { lexeme = ’TRDMH/’ gloss = ’sleep’ }
      EntryCluster: { ’{ nonbeing }-11775621’
         EntryCluster: { ’{ nonexistence, nonentity }-11775734’
            OE: { lexeme = ’>JN/’ gloss = ’non-existence’ }
      EntryCluster: { ’{ condition, status }-11745254’
         EntryCluster: { ’{ pathological_state }-11849251’
            EntryCluster: { ’{ ill_health, unhealthiness, health_problem }-11849364’
               EntryCluster: { ’{ injury, hurt, harm, trauma }-12060683’
                  OE: { lexeme = ’<YB=/’ gloss = ’hurt’ }
         EntryCluster: { ’{ emptiness }-12207503’
            OE: { lexeme = ’BHW/’ gloss = ’emptiness’ }
            OE: { lexeme = ’THW/’ gloss = ’emptiness’ }
         EntryCluster: { ’{ sanitary_condition }-12240110’
            EntryCluster: { ’{ dirtiness, uncleanness }-12242724’
               EntryCluster: { ’{ dirt, filth, grime, soil, stain, grease, grunge }-12242949’
                  OE: { lexeme = ’>DMH/’ gloss = ’soil’ }
         EntryCluster: { ’{ need, demand }-12202441’
            EntryCluster: { ’{ lack, deficiency, want }-12202684’
               EntryCluster: { ’{ absence }-11776507’
                  OE: { lexeme = ’BLTJ/’ gloss = ’absence’ }
         EntryCluster: { ’{ dominance, ascendance, ascendence, ascendancy, ascendency, control }-12196662’
            EntryCluster: { ’{ dominion, rule }-12197487’
               OE: { lexeme = ’MMCLH/’ gloss = ’dominion’ }
         EntryCluster: { ’{ hostility, enmity, antagonism }-11791405’
            OE: { lexeme = ’>JBH/’ gloss = ’enmity’ }
         EntryCluster: { ’{ illumination }-11793413’
            EntryCluster: { ’{ dark, darkness }-11793746’
               OE: { lexeme = ’XCK/’ gloss = ’darkness’ }
   EntryCluster: { ’{ assail, assault, set_on, attack }-884292’
      EntryCluster: { ’{ rape, ravish, violate, dishonor, dishonour, outrage }-2018846’
         OE: { lexeme = ’KBC[’ gloss = ’rape’ }
   EntryCluster: { ’{ get_rid_of, remove }-1752337’
      EntryCluster: { ’{ discard, fling, toss, toss_out, toss_away, chuck_out, cast_aside, dis-
pose, throw_out, cast_out, throw_away, cast_away, put_away }-1750922’
         EntryCluster: { ’{ abandon }-1755212’
            OE: { lexeme = ’<ZB[’ gloss = ’abandon’ }
   EntryCluster: { ’{ enter, come_in, get_into, get_in, go_into, go_in, move_into }-1586576’
      OE: { lexeme = ’BW>[’ gloss = ’enter’ }
   EntryCluster: { ’{ take, get_hold_of }-960049’
      EntryCluster: { ’{ seize, prehend, clutch }-958483’
         EntryCluster: { ’{ grip }-968244’
            EntryCluster: { ’{ bite, seize_with_teeth }-1144117’
               OE: { lexeme = ’CWP=[’ gloss = ’bite’ }
   EntryCluster: { ’{ oppress, suppress, crush }-1905919’
      OE: { lexeme = ’CWP[’ gloss = ’crush’ }
      EntryCluster: { ’{ repress, quash, keep_down, subdue, subjugate, reduce }-1905096’
         OE: { lexeme = ’KBC[’ gloss = ’subdue’ }
   EntryCluster: { ’{ equal, be }-2097262’
      EntryCluster: { ’{ match, fit, correspond, check, jibe, gibe, tally, agree }-2091846’
         EntryCluster: { ’{ meet, fit, conform_to }-2100053’
            EntryCluster: { ’{ satisfy, fulfill, fulfil, live_up_to }-2103002’
               EntryCluster: { ’{ suffice, do, answer, serve }-2101102’
```

```
            EntryCluster: { ’{ serve, function }-2102112’
               OE: { lexeme = ’<BD[’ gloss = ’serve’ }
EntryCluster: { ’{ travel, go, move, locomote }-1441983’
   OE: { lexeme = ’HLK[’ gloss = ’go’ }
   EntryCluster: { ’{ rise, lift, arise, move_up, go_up, come_up, uprise }-1545370’
      OE: { lexeme = ’<LH[’ gloss = ’go_up’ }
   EntryCluster: { ’{ fly, wing }-1524438’
      OE: { lexeme = ’<WP[’ gloss = ’fly’ }
   EntryCluster: { ’{ return, go_back, get_back, come_back }-1576099’
      OE: { lexeme = ’CWB[’ gloss = ’return’ }
   EntryCluster: { ’{ fall }-1548160’
      OE: { lexeme = ’NPL[’ gloss = ’fall’ }
   EntryCluster: { ’{ crawl, creep }-1483611’
      OE: { lexeme = ’RMF[’ gloss = ’creep’ }
   EntryCluster: { ’{ step, tread }-1648033’
      OE: { lexeme = ’RDH[’ gloss = ’tread’ }
EntryCluster: { ’{ make, create }-1278671’
   OE: { lexeme = ’BR>[’ gloss = ’create’ }
   EntryCluster: { ’{ construct, build, make }-1308020’
      OE: { lexeme = ’BNH[’ gloss = ’build’ }
   EntryCluster: { ’{ create, make }-1296780’
      EntryCluster: { ’{ do, make }-1280313’
         OE: { lexeme = ’<FH[’ gloss = ’do’ }
   EntryCluster: { ’{ beget, get, engender, father, mother, sire, generate, bring_forth }-43527’
      OE: { lexeme = ’JLD[’ gloss = ’beget’ }
   EntryCluster: { ’{ reproduce, procreate, multiply }-43825’
      EntryCluster: { ’{ breed, multiply }-44735’
         OE: { lexeme = ’RBH[’ gloss = ’breed’ }
   EntryCluster: { ’{ bear, turn_out }-1306064’
      OE: { lexeme = ’PRH[’ gloss = ’bear’ }
   EntryCluster: { ’{ arouse, elicit, enkindle, kindle, evoke, fire, raise, provoke }-1382318’
      EntryCluster: { ’{ invite, ask_for }-1382794’
         OE: { lexeme = ’QR>[’ gloss = ’invite’ }
EntryCluster: { ’{ designate, denominate }-813050’
   EntryCluster: { ’{ label }-812858’
      EntryCluster: { ’{ name, call }-811896’
         EntryCluster: { ’{ entitle, title }-812560’
            EntryCluster: { ’{ proclaim }-773164’
               OE: { lexeme = ’QR>[’ gloss = ‘proclaim’ }
EntryCluster: { ’{ arrive, get, come }-1577035’
   OE: { lexeme = ’BW>[’ gloss = ’arrive’ }
EntryCluster: { ’{ bless }-686842’
   OE: { lexeme = ’BRK[’ gloss = ’bless’ }
EntryCluster: { ’{ possession }-23182’
   EntryCluster: { ’{ property, belongings, holding, material_possession }-11171797’
      EntryCluster: { ’{ real_property, real_estate, realty }-11173744’
         EntryCluster: { ’{ land }-11176958’
            OE: { lexeme = ’>RY/’ gloss = ’land’ }
   EntryCluster: { ’{ assets }-11246282’
      EntryCluster: { ’{ material_resource }-11265318’
         EntryCluster: { ’{ wealth, riches }-11265442’
            EntryCluster: { ’{ treasure, hoarded_wealth }-11279942’
               EntryCluster: { ’{ valuable }-11280303’
                  EntryCluster: { ’{ precious_metal }-11280736’
                     EntryCluster: { ’{ gold }-11281047’
                        OE: { lexeme = ’ZHB/’ gloss = ’gold’ }
      EntryCluster: { ’{ sum, sum_of_money, amount, amount_of_money }-11247551’
         OE: { lexeme = ’R>C/’ gloss = ’sum’ }
EntryCluster: { ’{ gather, garner, collect, pull_together }-1093089’
   OE: { lexeme = ’QWH=[’ gloss = ’collect’ }
EntryCluster: { ’{ be }-2090076’
   EntryCluster: { ’{ dwell, shack, reside, live, inhabit, people, populate, domicile, domiciliate }-2086331’
      OE: { lexeme = ’CKN[’ gloss = ’inhabit’ }
      OE: { lexeme = ’XJH[’ gloss = ’be_alive’ }
   EntryCluster: { ’{ reach, extend_to, touch }-2114355’
      OE: { lexeme = ’NG<[’ gloss = ’reach’ }
EntryCluster: { ’{ be }-2047807’
   OE: { lexeme = ’>WR[’ gloss = ’be’ }
   OE: { lexeme = ’HJH[’ gloss = ’be’ }
   EntryCluster: { ’{ stay, remain, rest }-91056’
      OE: { lexeme = ’NWX[’ gloss = ’remain’ }
EntryCluster: { ’{ express, verbalize, verbalise, utter, give_tongue_to }-744133’
   EntryCluster: { ’{ curse, cuss, blaspheme, swear, imprecate }-686274’
      OE: { lexeme = ’>RR[’ gloss = ’curse’ }
   EntryCluster: { ’{ state, say, tell }-796572’
      OE: { lexeme = ’>MR[’ gloss = ’say’ }
EntryCluster: { ’{ touch, adjoin, meet, contact }-952943’
   EntryCluster: { ’{ cling, cleave, adhere, stick, cohere }-965528’
      OE: { lexeme = ’DBQ[’ gloss = ’cleave_to’ }
EntryCluster: { ’{ move, displace }-1454310’
   EntryCluster: { ’{ separate, disunite, divide, part }-1233907’
      OE: { lexeme = ’BDL[’ gloss = ’disunite’ }
   EntryCluster: { ’{ expel, throw_out, kick_out }-1966464’
      OE: { lexeme = ’GRC[’ gloss = ’expel’ }
   EntryCluster: { ’{ send, direct }-1531892’
      OE: { lexeme = ’CLX[’ gloss = ’send’ }
   EntryCluster: { ’{ put, set, place, pose, position, lay }-1182384’
      EntryCluster: { ’{ plant, set }-1241292’
         OE: { lexeme = ’NV<[’ gloss = ’plant’ }
      EntryCluster: { ’{ settle, settle_down }-1562045’
         OE: { lexeme = ’CKN[’ gloss = ’settle_down’ }
      EntryCluster: { ’{ sow, sough, seed }-1187282’
         OE: { lexeme = ’ZR<[’ gloss = ’sow’ }
   EntryCluster: { ’{ transfer }-1758708’
      EntryCluster: { ’{ convey, transmit, communicate }-1758405’
         EntryCluster: { ’{ communicate, pass_on, pass, put_across }-591089’
            EntryCluster: { ’{ request, bespeak, call_for, quest }-599379’
               EntryCluster: { ’{ ask }-599108’
                  EntryCluster: { ’{ request }-600025’
                     EntryCluster: { ’{ order, tell, enjoin, say }-594545’
                        OE: { lexeme = ’CJT[’ gloss = ’order’ }
                        OE: { lexeme = ’YWH[’ gloss = ’order’ }
EntryCluster: { ’{ dishonor, disgrace, dishonour, attaint, shame }-2003295’
   OE: { lexeme = ’BWC[’ gloss = ’disgrace’ }
EntryCluster: { ’{ determine, set }-556358’
   EntryCluster: { ’{ identify, place }-493411’
      EntryCluster: { ’{ distinguish, separate, differentiate, secern, secernate, severalize, sever-
```

```
alise, tell, tell_apart }-517479’
            OE: { lexeme = ’BDL[’ gloss = ’differentiate’ }
    EntryCluster: { ’{ precipitate, come_down, fall }-2175124’
       EntryCluster: { ’{ rain, rain_down }-2174897’
          OE: { lexeme = ’MVR[’ gloss = ’rain’ }
    EntryCluster: { ’{ move }-1438226’
       EntryCluster: { ’{ gather, congregate, collect }-1592409’
          EntryCluster: { ’{ crowd, crowd_together }-1595540’
             EntryCluster: { ’{ pour, swarm, stream, teem, pullulate }-1596222’
                OE: { lexeme = ’CRY[’ gloss = ’pour’ }
       EntryCluster: { ’{ shake, agitate }-1486638’
          OE: { lexeme = ’RXP[’ gloss = ’shake’ }
    EntryClus-
ter: { ’{ go_to_bed, turn_in, crawl_in, kip_down, hit_the_hay, hit_the_sack, get_into_bed, sack_out, go_to_sleep, retire }-
14444’
       OE: { lexeme = ’JCN=[’ gloss = ’go_to_sleep’ }
    EntryCluster: { ’{ phenomenon }-25413’
       EntryCluster: { ’{ process }-11422319’
          EntryCluster: { ’{ natural_process, natural_action, action, activity }-11403985’
             EntryCluster: { ’{ chemical_process, chemical_change, chemical_action }-11345803’
                EntryCluster: { ’{ chemical_reaction, reaction }-11346773’
                   EntryCluster: { ’{ oxidation, oxidization, oxidisation }-11414048’
                      EntryCluster: { ’{ combustion, burning }-11349264’
                         EntryCluster: { ’{ fire, flame, flaming }-11373370’
                            EntryCluster: { ’{ blaze, blazing }-11339665’
                               OE: { lexeme = ’LHV/’ gloss = ’blaze’ }
             EntryCluster: { ’{ organic_process, biological_process }-11410462’
                EntryCluster: { ’{ bodily_process, body_process, bodily_function, activity }-11340514’
                   EntryCluster: { ’{ breath }-629165’
                      OE: { lexeme = ’NCMH/’ gloss = ’breath’ }
                      OE: { lexeme = ’RWX/’ gloss = ’breath’ }
       EntryCluster: { ’{ natural_phenomenon }-9377075’
          EntryCluster: { ’{ physical_phenomenon }-9385918’
             EntryCluster: { ’{ energy }-9414774’
                EntryCluster: { ’{ radiation }-9456093’
                   EntryCluster: { ’{ electromagnetic_radiation, electromagnetic_wave, nonparticulate_radiation }-9413564’
                      EntryCluster: { ’{ actinic_radiation, actinic_ray }-9387759’
                         EntryCluster: { ’{ light, visible_light, visible_radiation }-9433880’
                            OE: { lexeme = ’>WR/’ gloss = ’light’ }
       EntryCluster: { ’{ consequence, effect, outcome, result, event, issue, upshot }-9378924’
          OE: { lexeme = ’PRJ/’ gloss = ’result’ }
    EntryCluster: { ’{ transfer }-1749745’
       EntryCluster: { ’{ give }-1732957’
          EntryCluster: { ’{ provide, supply, ply, cater }-934551’
             EntryCluster: { ’{ serve, help }-933410’
                OE: { lexeme = ’CQH[’ gloss = ’help’ }
    EntryCluster: { ’{ know, cognize, cognise }-474590’
       OE: { lexeme = ’JD<[’ gloss = ’know’ }
    EntryCluster: { ’{ be_full }-939186’
       OE: { lexeme = ’ML>[’ gloss = ’be_filled_with’ }
    EntryCluster: { ’{ have, feature }-2069912’
       EntryCluster: { ’{ bear }-2070367’
          OE: { lexeme = ’JLD[’ gloss = ’bear’ }
    EntryCluster: { ’{ leave, go_forth, go_away }-1579985’
       OE: { lexeme = ’JY>[’ gloss = ’go_forth’ }
    EntryCluster: { ’{ think, cogitate, cerebrate }-501870’
       EntryCluster: { ’{ chew_over, think_over, meditate, ponder, excogotate, contemplate, muse, re-
flect, mull, mull_over, ruminate, speculate }-503363’
          EntryCluster: { ’{ brood, worry, dwell }-560103’
             OE: { lexeme = ’CKN[’ gloss = ’dwell’ }
       EntryCluster: { ’{ reason }-504962’
          EntryCluster: { ’{ calculate, cipher, cypher, compute, work_out, reckon, figure }-508429’
             EntryCluster: { ’{ estimate, gauge, approximate, guess, judge }-534611’
                EntryCluster: { ’{ place, put, set }-535422’
                   OE: { lexeme = ’CJT[’ gloss = ’put’ }
                   OE: { lexeme = ’FJM[’ gloss = ’put’ }
                   OE: { lexeme = ’NTN[’ gloss = ’give’ }
       EntryCluster: { ’{ think_about }-584658’
          EntryCluster: { ’{ consider, take, deal, look_at }-584405’
             OE: { lexeme = ’R>H[’ gloss = ’look_at’ }
    EntryCluster: { ’{ find, happen, chance, bump, encounter }-1771305’
       OE: { lexeme = ’MY>[’ gloss = ’attain_to’ }
    EntryCluster: { ’{ decide, make_up_one’s_mind, determine }-554546’
       EntryCluster: { ’{ choose, take, select, pick_out }-536437’
          OE: { lexeme = ’R>H[’ gloss = ’choose’ }
          EntryCluster: { ’{ assign, specify, set_apart }-538617’
             OE: { lexeme = ’QDC[’ gloss = ’be_holy’ }
    EntryCluster: { ’{ act, move }-1860072’
       EntryCluster: { ’{ interact }-1868026’
          EntryCluster: { ’{ communicate, intercommunicate }-589741’
             EntryCluster: { ’{ inform }-660824’
                EntryCluster: { ’{ misinform, mislead }-662840’
                   EntryCluster: { ’{ deceive, betray, lead_astray }-678230’
                      OE: { lexeme = ’NC>=[’ gloss = ’deceive’ }
                EntryCluster: { ’{ report, describe, account }-762866’
                   OE: { lexeme = ’NGD[’ gloss = ’make_known’ }
    EntryCluster: { ’{ hold, take_hold }-961954’
       EntryCluster: { ’{ grasp, grip, hold_on }-961490’
          OE: { lexeme = ’LQX[’ gloss = ’grasp’ }
    EntryCluster: { ’{ act, human_action, human_activity }-22113’
       EntryCluster: { ’{ activity }-310023’
          EntryCluster: { ’{ work }-431993’
             EntryCluster: { ’{ service }-433614’
                OE: { lexeme = ’YB>/’ gloss = ’service’ }
          EntryCluster: { ’{ attempt, effort, endeavor, endeavour, try }-591212’
             EntryCluster: { ’{ best }-97738’
                OE: { lexeme = ’R>C/’ gloss = ’best’ }
                OE: { lexeme = ’R>CJT/’ gloss = ’best’ }
          EntryCluster: { ’{ occupation, business, job, line_of_work, line }-436669’
             OE: { lexeme = ’ML>KH/’ gloss = ’occupation’ }
    EntryCluster: { ’{ hide, conceal }-1690723’
       OE: { lexeme = ’XB>[’ gloss = ’conceal’ }
    EntryCluster: { ’{ breathe, take_a_breath, respire, suspire }-1742’
       EntryCluster: { ’{ exhale, expire, breathe_out }-3768’
          EntryCluster: { ’{ blow }-5886’
             OE: { lexeme = ’NPX[’ gloss = ’blow’ }
    EntryCluster: { ’{ connect, link, tie, link_up }-1072599’
```

```
   EntryCluster: { ’{ attach }-1025314’
      EntryCluster: { ’{ fasten, fix, secure }-1060284’
         EntryCluster: { ’{ sew, run_up, sew_together, stitch }-1050914’
            OE: { lexeme = ’TPR[’ gloss = ’sew_together’ }
EntryCluster: { ’{ open, open_up }-1065001’
   OE: { lexeme = ’PQX[’ gloss = ’open’ }
EntryCluster: { ’{ spread, distribute }-1091241’
   EntryCluster: { ’{ diffuse, spread, spread_out, fan_out }-1622062’
      OE: { lexeme = ’PRD[’ gloss = ’spread_out’ }
EntryCluster: { ’{ event }-21905’
   EntryCluster: { ’{ happening, occurrence, natural_event }-6067926’
      EntryCluster: { ’{ case, instance, example }-6088089’
         EntryCluster: { ’{ time, clip }-6088709’
            OE: { lexeme = ’P<M/’ gloss = ’time’ }
      EntryCluster: { ’{ beginning }-6073380’
         OE: { lexeme = ’R>C/’ gloss = ’beginning’ }
         OE: { lexeme = ’R>CJT/’ gloss = ’beginning’ }
         OE: { lexeme = ’VRM/’ gloss = ’beginning’ }
EntryCluster: { ’{ close, shut }-1064431’
   OE: { lexeme = ’SGR[’ gloss = ’close’ }
EntryCluster: { ’{ include }-2072148’
   EntryCluster: { ’{ hold, bear, carry, contain }-2126474’
      EntryCluster: { ’{ surround, environ, encircle, circle, round, ring }-2133098’
         OE: { lexeme = ’SBB[’ gloss = ’surround’ }
EntryCluster: { ’{ desire, want }-1432869’
   OE: { lexeme = ’XMD[’ gloss = ’desire’ }
```