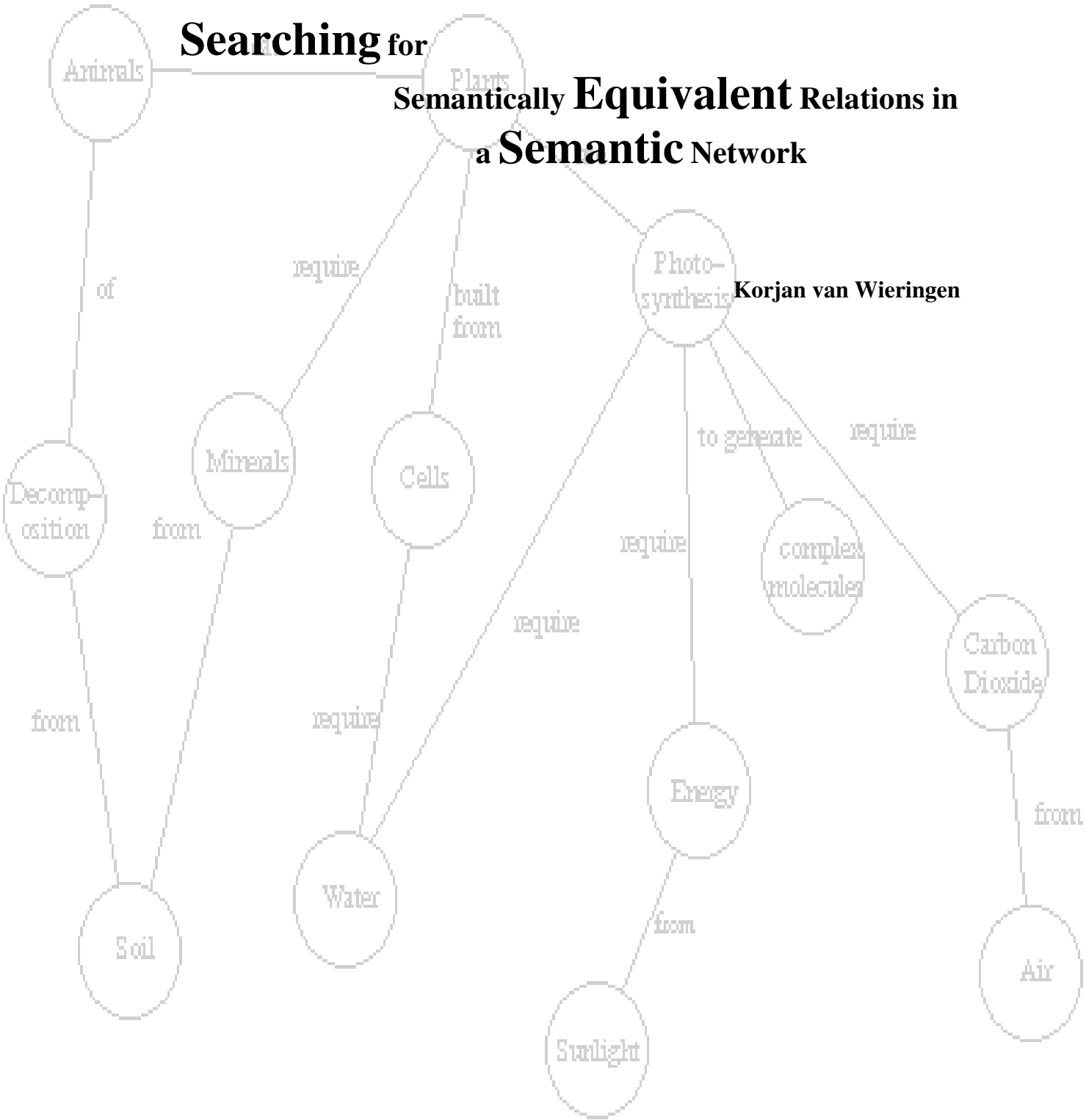
Searching for Semantically Equivalent Relations in a Semantic Network



Korjan van Wieringen

March 2003

Erasmus University Rotterdam
Rotterdam School of Economics
Master Thesis
Informatics & Economics
(Informatica & Economie)

Supervision by
dr. ir. Jan van den Berg
dr. ir. Uzay Kaymak

Supported by
TNO

Searching for Semantically Equivalent Relations in a Semantic Network

---

Korjan van Wieringen
<korjanvanwieringen@zonnet.nl>

The following acknowledgements will be in Dutch. This enables everybody who is mentioned in it to read it.

Acknowledgements

Inmiddels iets meer dan een jaar geleden ben ik begonnen aan iets wat tot nu toe de grootste uitdaging van mijn leven is gebleken; het schrijven van een scriptie. Welke uitdagingen er in de toekomst nog komen kan ik alleen maar raden. Ik weet echter wel dat het resultaat wat nu voor u ligt nooit tot stand had kunnen komen zonder de hulp van verschillende mensen.

Een paar van deze mensen wil ik graag bij naam noemen. Ik wil mijn ouders bedanken voor het bieden van de mogelijkheid om te studeren, de nimmer aflatende steun en het vertrouwen in mij. Ook Ellen wil ik bedanken voor het vertrouwen in mij en voor het feit dat ze me gedurende een jaar toch steeds kon motiveren om door te werken.

Verder wil ik natuurlijk mijn begeleiders Jan van den Berg en Uzay Kaymak bedanken voor hun steun, inzicht en inzet gedurende het jaar. Tijdens de rit hebben ze mij constant van oprechte feedback voorzien, maar zijn gelukkig ook wel mild geweest in de periodes waarin ik minder produktief was.

Natuurlijk wil ik ook mijn collega's bij TNO bedanken voor de gezellige tijd. Hierbij bedank ik in het bijzonder Ronald Poell en Jan Kleberg.

Contents

1. Introduction

"Onder onjuiste aannames is alles bewijsbaar"
-- dr. ir. Jan van den Berg

This chapter is the introduction of this thesis. It starts with the background of the thesis (1.1). Hereafter the research question will be mentioned (1.2), followed by the related research (1.3). The methodology used is given (1.4) followed by the chapter outline for the thesis (1.5).

1.1 Background

Nowadays most people in western civilisations are used to working with computers. Many people even use them on a daily basis. When using computers this often, one would expect it to be easy to communicate with them. We would expect to be able to talk to the computer, and expect it to talk back. Besides easy communication, we would expect it to be able to check for consistency in our writing, and automatically correct typing errors. In short, we want it to behave intelligently.

However, when we look for example at the search for information on the Internet, we see computer behaviour can hardly be called intelligent. Suppose we are watching television and there is a commercial for the musical 'Cats'. When we want to search the Internet for information on the location and ticket availability for the show, current technology forces us at best to use one of the Internet's search engines. Here, one should enter one or more words connected with the subject and press the search button. The result will be a number of sites in which one or more of these words occur. In contrast with the current situation, we could imagine a situation where we could ask the computer something like "Do you know if Cats will be in town?". To which it reacts with "One moment", searches the Internet and says, "I believe they will be here Saturday the 5th, shall I order tickets for you?". But, as we've seen, this is not the case. Why is it so hard for a computer to process and/or generate natural language?

In this thesis we will see that the difficulty computers have when dealing with natural language is one caused by the difficulty to deal with meaning. This thesis will be about computers and meaning. The difficulty when dealing with meaning can briefly be illustrated by continuing the example of searching information on the Internet. On the Internet, most documents contain natural language. Most search engines on the Internet use symbol matching to search documents about a certain subject. When using symbol matching, a search engine compares a search string (or parts of it) with the words in documents. It is possible for a word to refer to different objects in the world (homonyms) or for a certain object to be referred to by different words (synonyms).

Suppose one is looking for information on 'the musical Cats'. In one of the Internets' search engines, the search string 'Cats' can be entered. The result on a search for 'the musical Cats' will probably be a list of documents on the subject. It probably also contains documents about the animal 'cat' which can also be referred to with word 'Cats'. Furthermore it is possible that the search engine does not find certain documents about the musical 'Cats', because these documents do not contain the word 'Cats'. Here the musical 'Cats' is referred to by the words 'new musical by Andrew Loyd Webber'. Keep in mind, we already made the concession to search for Internet sites about 'the musical Cats' instead of typing the question "Can I book

tickets for Cats?" or "Will Cats be in my town in the near future?". Determining the subject of a document looks hard enough as is.

If symbol matching yields such poor results, why aren't search engines based on a different technique that takes meaning into account when searching? While progress is being made to develop search techniques based on something else than symbol matching, the meaning of natural language is hard to determine for a computer. In the example there was attention for the ambiguity of single words ('Cats', the musical and 'Cats', the animals). Determining meaning can even get harder when dealing with whole sentences instead of single words. The meaning of a sentence can change when words change position. For example with the words in the last sentence, we can make another sentence with a different meaning, namely: "Change position when the meaning of words can change a sentence". The meaning of a sentence can also change when there is irony or sarcasm in it. When someone loses a whole day work by a computer crash and calls out: "Well, that is just great!", he usually does not really mean that he thinks this is great.

Although natural language can have a lot of ambiguity in it, humans can work with it. It is likely the knowledge humans have about the world and the ability to deal with the context it is used in helps to work with natural language. At the moment, computers have difficulty working with natural language. A reason for this could be their lack of knowledge of the world and inability to interpret the context in which it is used. To overcome this, we could let a computer use a representation of knowledge. This means knowledge must somehow be represented in a form suitable for use by computers.

Some computer applications today already use a representation of knowledge. This knowledge is almost always domain specific; examples include domains like animals, medicine and geography. This knowledge is usually used to do a specialised task in a certain domain. However when we look at search engines on the Internet, the knowledge a search engine needs, can not be called domain specific, but rather an 'all domain overview'. The search engine does not have to be able to perform surgery to determine that an Internet site provides information about it. To determine what an arbitrary document is about, little knowledge about many domains is needed. Is it possible to merge existing domain specific knowledge bases into one, whereby it the possibility creates to use knowledge from all domains and to reuse knowledge for different applications in that domain?

When it is attempted to create one knowledge base out of many, one need to keep in mind the different domain specific knowledge bases will probably have different authors. Each author wants to represent his knowledge as detailed as is necessary for his application(s). Each representation namely needs to be accurate enough to enable a computer application to reach its goals. This means that a method of representation needs to be able to represent everything other methods of representation can also represent. Hereby it remains possible for all authors to represent the knowledge they need.

At TNO (a Dutch organization for applied science) an attempt is being made to create a knowledge base that is able to store knowledge which is represented by different authors. This creates the advantage of having knowledge about different domains stored in the same way and also creates the possibility to "re-use" this knowledge for other applications.

This thesis is written after an internship at TNO. TNO is using a semantic network (see chapter 2 for further information) to represent knowledge in a computer. Semantic networks in computer science exist since the early '60's, but are here only for use by one author. TNO has

made an extension on the semantic network that enables the representation of knowledge by many authors.

1.2 Research Question

We have seen why it is interesting to create one knowledge base for knowledge used by many different applications. Let us now look at where the challenges lay. Using one knowledge base to store the represented knowledge of different authors, allows two authors to independently represent the same knowledge. When considering two representations of knowledge, where the first representation represents a part of the knowledge that is also represented in the second, the first is called *semantically equivalent* to the second.

Semantical equivalence differs from the related term 'synonymy' because synonymy implicates a symmetrical relation. When 'bank' is synonymous with 'financial institution' it is also the other way around. With semantic equivalence this is not the case, representation *A* could represent the same knowledge as representation *B*, but not the other way around. Representation *B* could namely represent more than *A*.

When there is hidden semantic equivalence in the network, it will lead to 'underperforming' applications. An 'underperforming' application is an application which can access a knowledge base, but is unable to act like it knows it. This will be illustrated by the following example. Consider two applications, *A* and *B*. The knowledge for application *A* is represented as "Shakespeare has written Othello". The knowledge for application *B* is represented as "Shakespeare has created Macbeth". When the applications are not programmed to work with the representation of the other, both only behave like they know one play (or piece, or book), by the hand of Shakespeare. This way, these applications perform less than possible considering the represented knowledge they both can access. If we could make both applications work with writing and creating, both applications can behave like they know "Macbeth" and "Othello" were creations of Shakespeare.

Discovering semantic equivalence in the network is therefore an interesting challenge. Discovering semantic equivalence however has a few pitfalls. For example, while 'writing' does always mean 'creating', this does not hold the other way around. One is tempted to say the creator of a book will also have written it, so under the boundaries that the knowledge is about a person and a book, 'creating' is equivalent to 'writing', but how about the publisher of a book? We can also say that the publisher of a book created the book.

While a large knowledge representation will probably contain thousands and thousands of elements, it becomes merely impossible for humans to discover hidden semantic equivalence without some initial shifting between expected semantically un-equivalent situations and semantically equivalent representations. While we have seen meaning is a hard subject, final judgement about the semantic equivalence of representations is currently done by humans. The question arises whether this judgement can also be performed by a computer. A computer which can judge semantic equivalence just as well as humans will be also be able to find all semantic equivalence in a semantic network.

A semantic network has three main elements that can be used to represent knowledge. These elements are: concepts, relations and attributes. This thesis will focus on one of these three namely relations. This leads to the following research question this thesis will focus on:

To which extent is it possible to automate the finding of collections of semantically equivalent relations in a semantic network?

At a more pragmatic level, we will develop a computer algorithm that is capable of selecting semantically equivalent relations from the current semantic network at TNO.

1.3 Related Research

The semantic network, as used at TNO [7], is different from the usual semantic networks found in literature [11]. Therefore there is little publicized research on this extension of the semantic network. This thesis will be the first on semantic equivalence. Another thesis [8] however has also focused on meaning in a semantic network.

Important research on the meaning of words [6] has been very useful for this thesis, as well as a paper on the question "What is Knowledge Representation?" [2].

One has tried to define the semantics of a 'usual' semantic network [1] and used a semantic network for information extraction [3].

As far as we are concerned with the similarity in meaning, research has been done to discover the semantic distance in WordNet [16]. One has also looked at the usefulness of Taxonomies and Ontologies to represent knowledge and search for the similarity in it [15][17].

1.4 Methodology

A literature survey was the start of this thesis. Related research as well as most other references comes from the Internet. Unfortunately I still had to search it myself instead of leaving the search to a computer. After the literature survey and an introduction on semantic networks at TNO, the research question was still open.

It was however already clear that enabling many authors to represent knowledge in one knowledge base would create the challenge of dealing with the meaning of what is represented. A lot of time hereafter was spent trying to crystallize the problem at hand.

While trying to come to the problem, approaches to a solution were created and rejected. In the end however three main approaches to the problem kept standing. These approaches were not yet rejected, but acceptance was only reasonable with a more pragmatic view. An implementation of one of the theories was made and validated based on the results on a test-set.

1.5 Chapter Outline

After this introduction to the thesis, chapter 2 will contain a more elaborate explanation of semantic networks. Here we will see the extension TNO has made to enable multiple authors to use the same semantic network to represent knowledge. This usage by many authors leads to possible semantic equivalence in a semantic network. The issues of meaning in the network and semantic equivalence are discussed in chapter 3.

In chapter 4 we will focus on conceptual solutions to search for semantic equivalence. One of the suggested solutions is implemented and tested on the existing semantic network at TNO. Chapter 5 will be about the implementation, chapter 6 about its results on the test-set.

Chapter 7 holds the conclusions of this thesis, together with suggestions for further research.

2. Semantic Networks

"Wat betekent eigenlijk eigenlijk?"
-- Wim T. Schippers

Before starting a search for semantic equivalence in a semantic network, it has to be clear what a semantic network is and what it is meant by semantic equivalence. This chapter will focus on semantic networks. Semantic networks can be used to represent knowledge. Therefore we will first go into knowledge representation in general and discuss a theory on the way humans deal with meaning (2.1).

Next, the theory on semantic networks is given (2.2). This is split up in a part about elements that can be used to build a semantic network, which is called its syntax, and a part on the meaning in a semantic network, which is called its semantics.

The semantic network used in this thesis is being developed at TNO. This semantic network is an extension of the syntax of a 'general' semantic network. The extension of the syntax will first be explained, followed by the consequences this has for the semantics of the semantic network (2.3).

This chapter ends with a discussion of the advantages and disadvantages of the extension of the semantic network (2.4).

2.1 Knowledge representation

Humans are in constant interaction with their environment. The environment is also called the world or reality. Humans are not only interacting with their environment, they are also part of it. In the world there are millions of objects. Whenever people come across one of these objects, it causes a reaction in the brain by being seen, heard, felt, smelled or tasted. The way someone thinks an object looks, sounds, tastes, smells or feels like is called a concept of that object in the mind of the thinker. Besides concepts of physical objects, it is possible to have abstract concepts. Abstract concepts can stand for a group of objects, like the concept 'cats'.

Figure 2.1 A man with a concept of how the world looks like

Abstract concepts can also be things like math, geography, truth or time. Concepts can be anything humans can think of as being a coherent whole or sharing some common property. Generally speaking, concepts can be anything in the world, including the ideas and concepts of humans in the world. Figure 2.1 shows someone with a concept of how the world looks like.

People tend to bring structure to everything they encounter. We could say a concept like ‘a cat’, is created by encountering multiple cats and structuring the properties they have in common to ‘conclude’ these will probably be the characteristics of everything that is ‘a cat’. This process sounds very similar to the scientific method called induction.

The tendency to associate concepts can also be seen as the will to structure. People are able to associate different concepts and seem to be very willing to do so. Some of these associations sound ‘logical’ like associating the concept ‘rain’ with the concept ‘getting wet’ or

associating ‘fire’ with ‘hot’. Other associations seem to be less ‘logical’ some sportsmen tend to associate the concept ‘wearing particular underwear’ with the concept ‘winning the game’ others associate ‘black cat’ with ‘bad luck’.

While there are millions of objects, one human can impossibly create concepts of everything in the world. Often, person A has concepts that person B has not and vice versa. One way humans use to inform another of their concepts, and the associations they have on them, is language. When person A knows there is a lion coming and he thinks person B does not, person A can tell B that “there is a lion coming” or he can tell him to “get out of here”. Person B can take action on hearing this and decide to get out of there.

To do so sounds very simple, but how does person B know the meaning of the things person A says? To understand the way humans attach meaning to the symbols in language, we will take a look at a sample theory on the meaning of words, the so-called ‘meaning triangle’ [6].

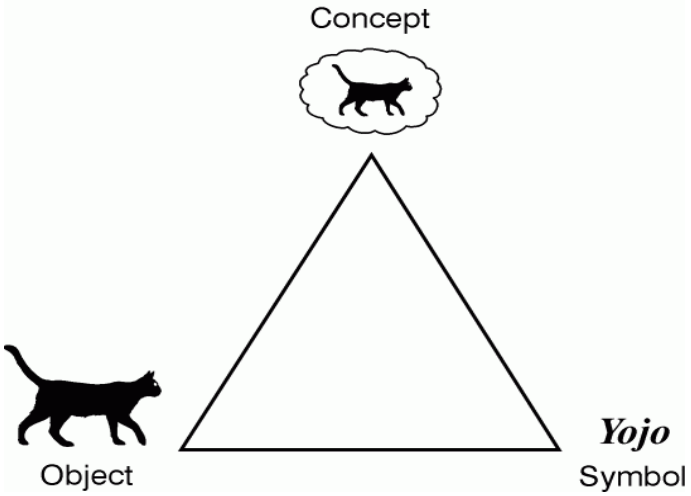


Figure 2.2 The 'meaning triangle' taken from [Sowa]

According to the meaning triangle, the symbols in a language acquire meaning, not by directly referencing objects in the world, but by referring to concepts of these objects which are in the mind of a cognitive agent. Note that the word symbol (or symbols) in this thesis is used in its broadest sense. It can stand for one character or a whole word, even a group of words, both the written as well as spoken. It can also stand for a picture or a sign.

On the left, figure 2.2 shows an object in the world, a cat called “Yojo”. On top stands the idea or concept one has in mind when thinking of the cat called “Yojo”. On the right is the

symbol or sign one can use to refer to the cat called “Yojo”. When people talk or read about “Yojo”, they actually are referring to the concept of the cat they have in mind.

Is it necessary for person A and B to have exactly the same concept? We believe this to be unnecessary. This will be illustrated by the following example. Assume two people are looking at a cat. This cat is half black, half white. From one point of view, only its black side could be seen, so while most people have occasionally seen a cat, the concept of “Yojo” it creates is probably one of a totally black cat. The same goes for the other person with a different point of view, who would only see its white side. The concept “Yojo” in their mind is probably totally white. So, while the object is not changed, it can create two different concepts in one’s mind (see figure 2.3). Keep in mind that neither concept, that of a totally white or totally black cat, is right. Still when people want to talk about the cat, it is possible.

Figure 2.3 Two man looking at “Yojo”

A concept is often as detailed or accurate as needed to help accomplish a goal. Suppose a disagreement occurs about the colour of the cat. Someone states “Yojo” is a black cat, while another states it is a white cat. They can take a closer look at the cat to refine their concept. By attaching meaning to the symbols in natural language, humans are capable of communication. This is possible even though the concepts that correspond to the symbols are not equal for sender and receiver. As long as it is equal from the context they need it for. For example, when one needs information about the weight of the cat to determine its diet, the one that is asked does not necessary have to have the same idea on the colour of the cat. This means person A and B do not have to have exactly the same concept in their mind as long as it refers to the same object.

From the meaning triangle we can derive at least three properties of a human involved in the process of communication:

- The ability to perceive objects in the real world, for which most people have the availability over five senses.
- The ability to store the concepts, related to the objects, in the mind.
- The ability to communicate about these concepts with one another by referring to it with symbols.

To successfully communicate with each other using language, humans need to share at least a minimum set of symbols to which they attach in essence the same meaning. Someone who only speaks French will have great difficulty to communicate using language with someone who only speaks English.

Simplified we could say humans use natural language to exchange information with each other. The meaning of the symbols in language is based on a common understanding between people. The meaning of a symbol is the concept to which it refers in the mind of a person. The meaning of the concept in the mind of a person is the object in the real world related to the concept.

Humans can of course communicate in many different ways besides natural language. People can for example make gestures at each other to inform the other of their intentions. They can also look at each other in a certain way. Sometimes it is even enough to say absolutely nothing to communicate your ideas about something. The simplified presentation above is however sufficient for explaining the reason why and the way how humans work with language. Before discussing a model of how computers can communicate, we could question if computers also 'need' to communicate. Here the term 'communication' is not used for the technical use of communication between computers. We all agree the World Wide Web and email are excellent ways of communication, but it is a communication between humans, technically enabled by computers. With communication is meant the exchange of knowledge. Do computers need to be able to exchange knowledge? To answer this question, we will first look at the term 'knowledge'.

There are many definitions of knowledge. The classic definition used in philosophy is that knowledge is *true, justified belief*. However, it is being questioned if this is always true [4]. Another definition of knowledge defines it along side with two terms on things that can be stored in a computer. Table 2.1 derived from [10] shows these definitions.

Table 2-1 Definitions of data, information and knowledge

	Definition	Example
Data	Uninterpreted signals or symbols sensible in the world or stored in a computer	…---…
Information	Data to which meaning is attached	Morse-code: S.O.S
Knowledge	The whole of data and information on which humans can understand each other and act in creating new information or taking action	Humans in need, start rescue operation

Although people can have knowledge about something, can we also say a computer has knowledge about something? This is a difficult question, because is it not so that we see

knowledge only existing in living creatures, preferably human? Is it not so a computer is an eccentric calculator that outperforms men only in computation? Don't we see the hand of God in everything we do not yet understand and the hand of men in everything we do understand? Does the acceptance of a computer holding knowledge means a crack in the superiority of men? We see how one question can trigger others, but fortunately we do not need to answer these questions here. It is not necessary to answer this question, while a computer uses a representation of knowledge. A representation of knowledge is not the knowledge itself. So there is no need to discuss whether computers can actually know something as long as they can successfully represent it. Gladly we can leave discussions about the meaning of 'to know' to philosophers. Here we state a representation of knowledge need a) to be in a format understandable for humans, and b) must cause the system to act *as if it knows it*. [12]

This also answers the question if computers need to communicate. This is the same reason as goes for humans; the exchange of knowledge. It is unlikely one human or computer could hold all the knowledge needed to achieve its goals at a given time. When one computer application behaves as if it knows something and we want another program to behave the same,

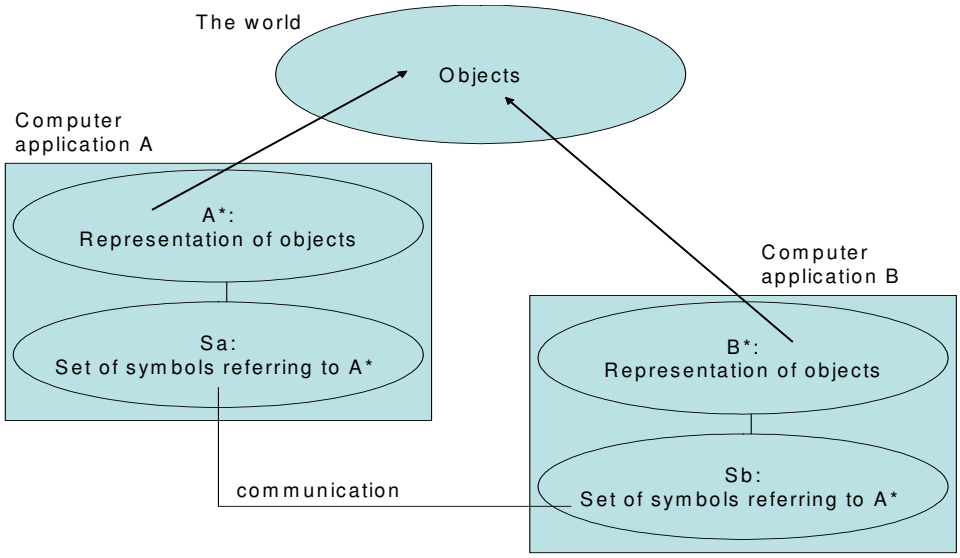


Figure 2.4 Model of communicating computer applications

one approach is to let the first application communicate the representation of the knowledge to the second so the second application can also behave as if it knows it.

Figure 2.4 shows a model of how computer applications could communicate. This model is inspired by the sample theory on the way people communicate. Here we also see a representation of objects in the world, as well as a set of symbols that can refer to the representation. We started this chapter by asking how humans deal with meaning. The question arises now how computer applications deal with meaning.

The knowledge one needs depends on the goals one has. One could wonder which part of the model represents the represented knowledge. This depends however on the behavior one expects from the computer application. One could expect it to be able to behave as if it knows which symbol corresponds to which representation. This knowledge is not represented in the model. Most of the time however, one expects the computer application to behave as if it knows

something that exists in the world. The knowledge to do so can partially be found in the model, namely as the representation of objects in the world.

A representation of objects in the world will need to hold information on the possible actions that are possible for an object. A computer application for example needs a representation of the knowledge that 'a cat can be stroked' and 'one can sit on a chair' in order to act as if it knows it.

Knowledge can be represented using a symbolic approach. The challenge of a symbolic approach is that it forces one to split some knowledge into several parts. For most knowledge however it is possible to choose different splits. This division is mostly based on the viewpoint of a person and depends on the goal one has in mind. A good example of possible ways to divide knowledge is seen when we search for information on 'Knowledge Representation". Different authors have different goals and therefore divide the concept differently. There is a division in symbolic and connectionistic [9]. Others divide Knowledge Representation into four main features [12] or in five different roles [2].

Here we will focus on the symbolic approach called a 'semantic network'.

2.2 Theory of a Semantic Network

According to one of the first publications on semantic networks, a semantic network is defined as: *A graph structure in which nodes (or vertices) represent concepts, while the arcs between these nodes represent relations among concepts.* [14] The concepts and relations used in a semantic network follow the here for presented theory on how humans deal with meaning. What we have also seen is that one can say humans tend to structure the world in concepts and associations (or relations). Next we will present the constructions that are allowed in a semantic network.

2.2.1 Syntax of a Semantic Network

A semantic network is a graph representing concepts and relations representing knowledge. So what exactly is a concept? There are a lot of different objects in the world, there can be even more different concepts. A concept can reach from something physical, like "Yojo" the cat, to something abstract like an idea. It could be a collection, like "all students at Erasmus", or a single element from a collection, like "a student at Erasmus". As we see a concept in a semantic network closely follows the concepts humans can have according to the 'meaning triangle'.

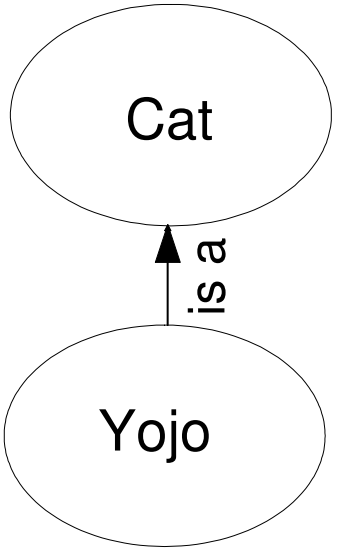


Figure 2.5 Semantic network representing 'Yojo

Between concepts, there are relations, this also follows the way people structure the world. The concept "Yojo" has the relation "is a" with the concept "cat". The concept "cat" has the relation "has a" with the concept "tail". Two concepts have a relation when they can be associated in the world a certain way. A relation represents the way a person believes two concepts are connected. Using concepts and relations, the knowledge that "Yojo is a cat" can be represented (see figure 2.5). This minimal network can be extended with, for example, the knowledge that "a cat has a tail" (see figure 2.6). These networks have a small set of relations, (is a, has a, is part of) which are the only relations used between concepts. From now on, unless explicitly stated otherwise, a concept refers to a concept in a semantic network.

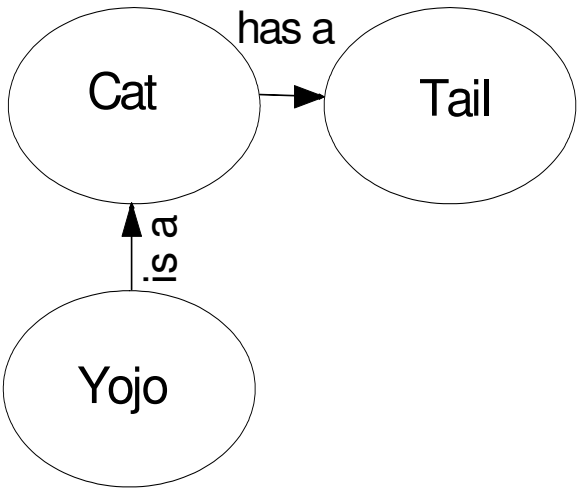


Figure 2.6 A small semantic network

So the allowed syntax for a semantic network exists of a small set of relations that can be used between an unrestricted number of concepts. Next, we look at the meaning these elements have.

2.2.2 Semantics of a Semantic Network

What are the semantics of a semantic network? It becomes hard to pinpoint the exact meaning of the node labeled “Yojo”. It could be the object “Yojo” or the concept in a human brain. It would simplify the matter if we could state the concept “Yojo” in the semantic network refers to the object in the world. This way, there is only one object represented and we avoid the question to which mind is being referred to. This however points at another difficulty. When a node in the semantic network refers to an abject in the world, does something like ‘math’ exist in a world without the humans that created it? Perhaps a more pragmatic approach is appropriate

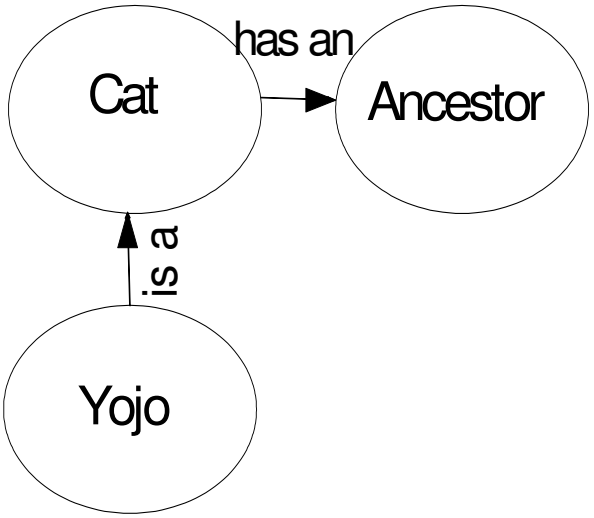


Figure 2.7 Another small semantic network

here. A node refers to the concept and behavior that follows from the concept (considering its context) on which most people would agree. Note this leaves the meaning of a concept fuzzy.

A semantic network however, is created by a person (or sometimes a group of persons) giving his (their) view on the objects and how they are related. This means different semantic networks describing knowledge in the same domain will probably contain different concepts.

So a semantic network is created according to a (or more) person(s) view on the world. While persons could have a different view, they look at the same world. The meaning of the knowledge in their mind and therefore of the representation in the network should therefore be the same.

To further explore the meaning of a semantic network, let us take another look at the semantic network representing “Yojo is a cat” and “a cat has a tail”. From this network, a computer could deduct that “Yojo has a tail”. But what is the meaning of this deducted “knowledge”. Does it mean every cat has the same tail, Yojo included? Or does every cat have its’ own tail. Most people would agree of course that ‘Yojo’ does not have the same tail as any other cat. This is our knowledge, but not necessarily what is represented in the network. Suppose the following was represented “”Yojo is a cat” and “a cat has an ancestor” (see figure 2.7).

In contrast with a tail, which is a part of cat and which is unique for every cat, an ancestor is not part of the cat and it is possible for cats to share ancestors. This is a property of an ancestor. It lies hidden in the meaning of the concept ancestor. To “have a tail” is something else then

"having an ancestor". This is a recognized issue when working with semantic networks and therefore it should be made very explicit which inferences can be made from certain relations.

Finally the 'Oedipus' example will be used to point out some subtleties. For people who do not know the story about Oedipus, here follows a very brief summary of the story. It points out how subtle dealing with meaning can be. This is the story of Oedipus by Sophocles:

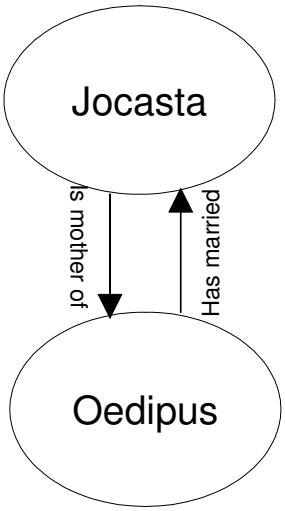


Figure 2.8 Oedipus has married Jocasta

"Laius and Jocasta were king and queen of Thebes, a town in Greece. One day, they had a baby boy. An oracle prophesied that the boy would grow up and kill his father and marry his mother. To thwart the prophecy, Laius and Jocasta decided to kill their baby. In those days, it was usual to leave an unwanted or defective baby in the wilderness. Laius and Jocasta did this.

A kindly shepherd found the baby. He gave the baby to a friend, who took it to Corinth, another town. The king and queen of Corinth couldn't have a baby of their own. So they adopted the foundling.

Nobody ever told little Oedipus that his mother was never pregnant. One day, after he had grown up, a drunk mentioned his being adopted. Oedipus questioned his parents, but they denied it. Oedipus visited various oracles to find out whether he was really adopted. All the oracles told him instead that he would kill his father and marry his mother.

To thwart the oracles, Oedipus left Corinth permanently. Travelling the roads, Oedipus got into a traffic squabble and killed a stranger who (unknown to him) was King Laius.

Soon Oedipus's smarts saved the town of Thebes, and he was made king. Oedipus married Laius's widow, Queen Jocasta. He ruled well, and they had four children. Eventually, Oedipus and Jocasta found out what had really happened. Jocasta committed suicide, and Oedipus blinded himself and became a wandering beggar."[1]

We could in short make a semantic network (figure 2.8) from the story of Oedipus. From this network we could conclude Oedipus married his mother, which is in fact true. Everyone that knows the story will however say the statement "Oedipus married his mother" is too rigorous and prefers the milder "Oedipus married Jocasta". At the time they got married they namely did not know they were mother and son. It depends on the context whether "Oedipus married his mother" is true. All people at the wedding would at the time have believed that "Oedipus married his

[1] Summary by E. Friedlander, http://www.pathguy.com/oedipus.htm

mother" was false. This points out that it is not always as easy to draw conclusions from a network as it initially seems. It also points out that what is true can depend on context.

When we go back to the two demands of knowledge representation, we see computer applications using this kind of semantics are complying with criterion a), i.e. the knowledge is in a human understandable format. But to comply with criteria b), acting like the application "knows" it, is only possible when it does not need to know a lot. For example, when a computer application only has to answer questions like "does a cat have a tail?" and "Is Yojo a cat?" an interpreter for these questions can be written as also an interpreter for the semantic network. These interpreters will also be a representation of knowledge, namely a procedural representation (see 2.1). This way the questions can be answered and the computer application acts as if it knows it.

The difficulty with a semantic network using only a predefined set of relations is that it is impossible to be able to represent all views on the world that are possible. When person *X* has knowledge about person *A* loving person *B*, how can it be represented using only the relations: "has a", "is a" and "is part of"? It is possible to extend the set of predefined relations with the relation "loves" and also describe very precisely which inferences can be made from it. But there are many more relations which can not be all included in the predefined set. Therefore, an extension is made on this kind of semantic network.

2.3 Extension on the theory of a Semantic Network

2.3.1 Extension on the Syntax of a Semantic Network

The first extension on the semantic network is the addition of attributes. An attribute stands for a property a concept or relation can have. An attribute is represented by a box connected with a concept or the relation (see figure 2.9). This semantic network represents the knowledge "Yojo is a cat. The eye color of Yojo is green".

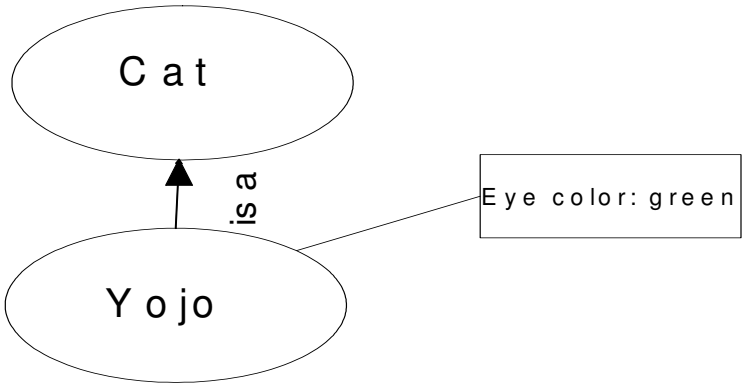


Figure 2.9 A small network with an attribute

An attribute is made up from an attribute type, in this case 'eye color' and an attribute value, in this case 'green'. While this attribute defines a property of a object in the world, it could also represent a property inserted specifically for an computer application. For example an attribute of the concept "Yojo" could be the date this concept was created (see figure 2.10). Such

an attribute is called a system attribute. While in this example the difference between system attributes and normal attributes is clear, there can be cases where it is not.

This extension on a semantic network does not make it possible to represent more knowledge in a semantic network. Every attribute can namely be represented as a relation with a concept.

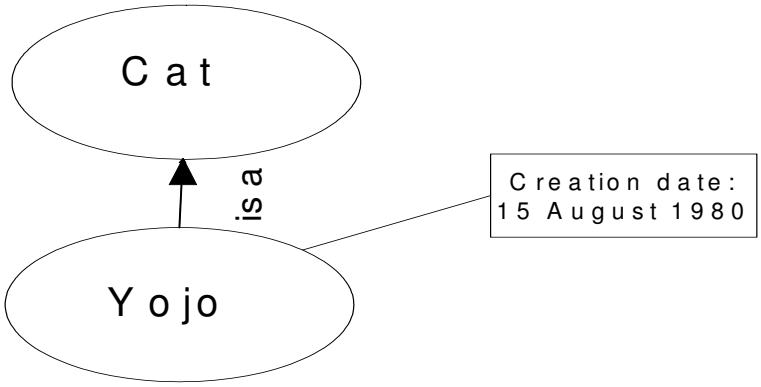


Figure 2.10 Another small network with an attribute

Figure 2.11 represents the same knowledge as figure 2.9. In figure 2.11 only, the attribute on 'Yojo' is translated into a relation. This can be done for every attribute by making the attribute type the relation and the attribute value the concept. While it does not extend the quantity of

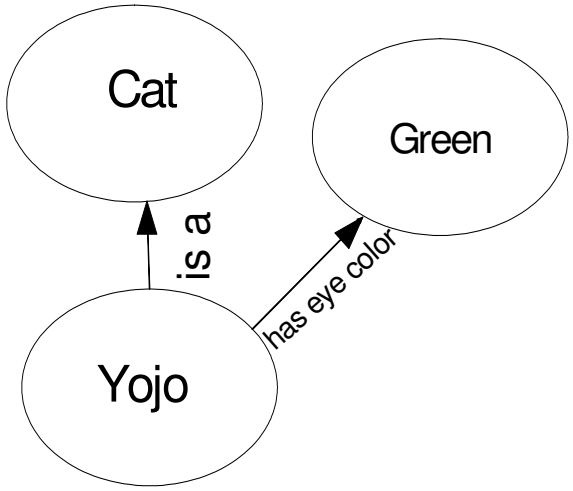


Figure 2.11 Attributes in the network can also be represented as concepts

knowledge that can be represented, it does create the possibility to stop representing knowledge in concept when it is not necessary. Suppose one wants to represent the knowledge that "Yojo has two eyes". It is possible to represent the number '2' as a concept. This concept number '2' could have a "is a" relation with the concept 'number'. While this representation is also valid, it is most of the time unnecessary. An application can be developed with procedural knowledge to behave correctly when encountering 'two' as attribute value.

The possibility to translate attributes into relations is the second extension on the semantic network. While the 'standard' semantic network works with a small set of predefined relations, here this is no longer the case. With the extended network it is possible to describe any

relation one thinks necessary to represent the knowledge. This is done while many different people could have many different representation of their knowledge in mind.

When using an unrestricted number of relations, it is could be useful to represent knowledge on a relation. Here for, the extension of the network splits a relation in three parts, a subject concept, a predicate concept and an object concept. In the sentence “Shakespeare has written Othello”, “Shakespeare” is the subject and “Shakespeare” has the association ‘has written’ with the object “Othello”.

By making the predicate a separate concept in the network, it is possible to represent knowledge that holds for the predicate separate from knowledge that holds for the relation as a whole. The relation “Shakespeare has written Othello” could have an attribute with the date he wrote it. The predicate “has written” could be in relation with the predicate “is writing” (see figure 2.12)

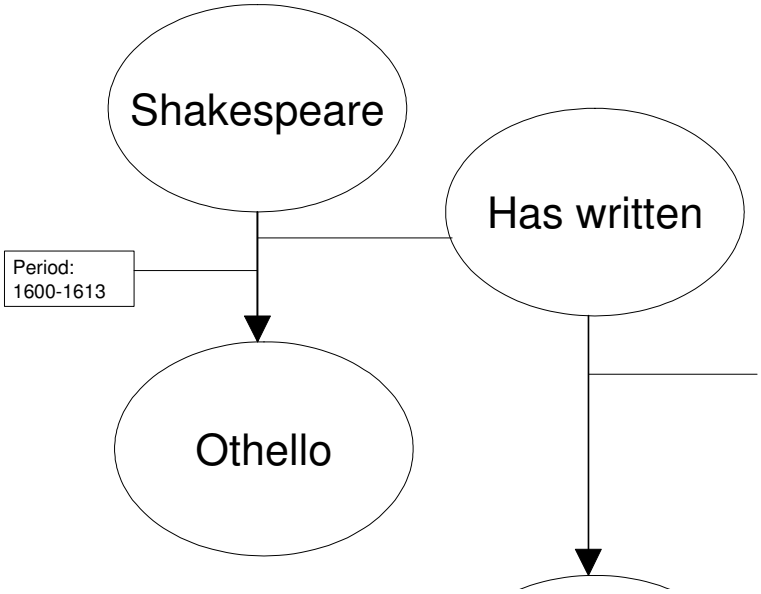


Figure 2.12 In a relation is referred to a predicate

For the rest of this thesis when we use the term semantic network, it refers to the extended semantic network unless explicitly stated otherwise. We will use the following notation to refer to the elements of a semantic network (illustrated by figure 2.13):

C_n : Concept n.
P_n : Concept n which can be used as a predicate in a relation.
R_{nop} : A relation with subject node C_n, predicate node P_o and object node $C_{p.}$
XA_i : Attribute A_i on element X. X can either be concept C_n or relation R_{nop}.

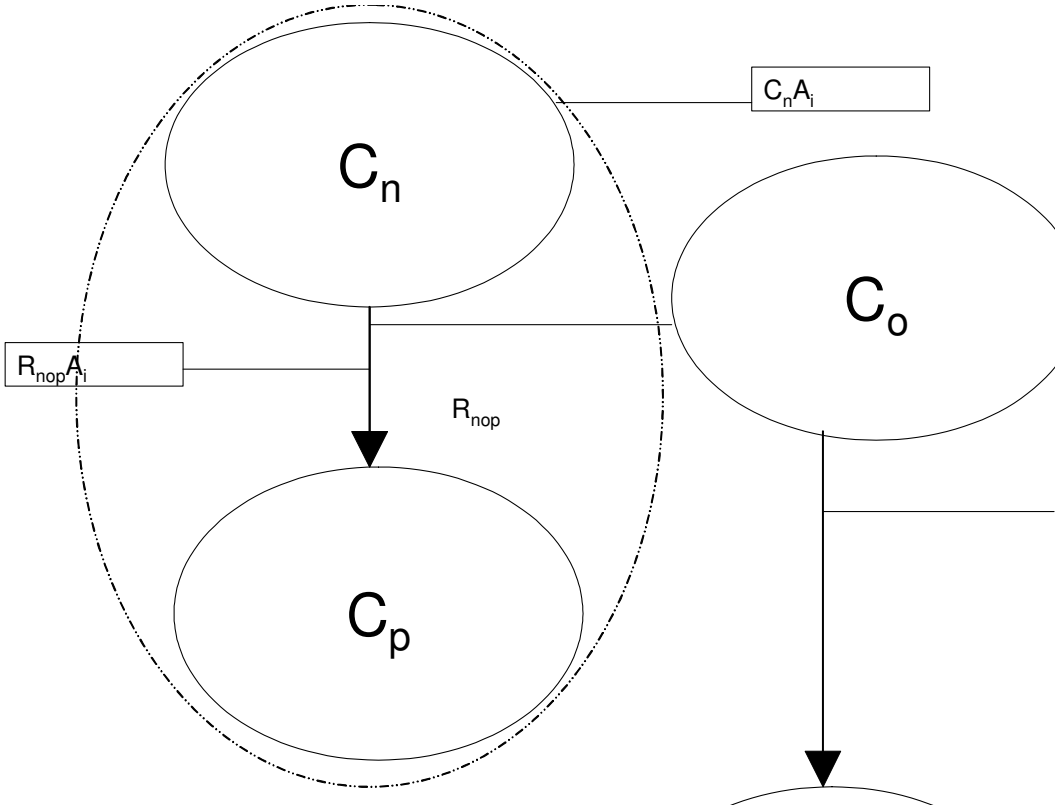


Figure 2.13 A semantic network to illustrate the used notation

2.3.2 Consequences for the Semantics of a Semantic Network

The semantics of the attributes added to the network is not different from the semantics of those attributes in relational form. It does point out there is probably an application with procedural knowledge on what to do when encountering this attribute.

The main consequence for the semantics lies in the extension of a small set of predefined relations to an unrestricted set. When working on a network with a set of predefined relations, it is possible to add the procedural knowledge to an application for each specific relation. An application can be programmed to do action *B* when encountering relation R_{npm} between concept C_n en C_m. With a small set of relations, it is possible to do this for every relation in the set.

It was already mentioned (2.2.2) that while there is a large number of possible concepts, it is hard to put the procedural knowledge in an application which makes the application take a specific action valid to the meaning of a concept for every concept in the network. By working with a large number of possible relations, it also becomes impossible to do this for relations anymore.

It is now possible to model a piece of knowledge in many different ways, specific for the application one has in mind for it. So, the concepts, attributes and relations stored in the network always represent a piece of knowledge. This knowledge will, most of the time, be clear to humans (see criteria a) of the KRH). For computer applications, not developed with the procedural knowledge to act on reading these elements from the network, they can be said to be meaningless to these applications.

So without further knowledge, the only deduction that can be made from a certain concept is that it exists. The only meaning a relation has, is that two concepts are related in that

way. The meaning of an attribute is only that a concept or relation has that property. So for a computer application not developed to use certain concepts, relations and/or attributes, the only meaning they have for it, is that they exists.

2.4 (Dis-) Advantages of the Extension of the Semantic Network

The extension of the semantic network makes it possible for different authors to work on the same semantic network and still represent their knowledge exactly the way they think is right. Using this extension it is possible for multiple applications to use the same network to store their knowledge. Using the network without this extension, the small set of relations only allowed one view to be represented. Here for it was merely impossible to allow multiple applications to store knowledge in such a network.

By allowing unlimited relations and many applications to work with the network, it becomes difficult to equip an application with all procedural knowledge needed to act properly on encountering an element from the network. A semantic network can now contain the representation of the knowledge according to many different views. For application whose only goal is to visualize this network, it gives no problems. When an application needs to deal with meaning however, what procedural knowledge should instruct it how to act when encountering an unknown element?

One approach is to search the network for equivalence in semantics. Suppose an application has the procedural knowledge to do action *A* when encountering element *B*. It has no knowledge about the action to take when encountering element *C*. When the procedural knowledge is enriches with the information that element *C* has the same meaning as *B*, the application can take action *A* in both cases.

The main difference between a ‘normal’ semantic network and an extended one is the possibility to add knowledge to it by many different applications. It’s a balance between completeness of inference and expressiveness. In an extended network there are many forms of equivalence. This will be explored in the next chapter.

3. Semantic Equivalence in a Semantic Network

"De aarde is rond. Net als een pannekoek."
-- Herman Finkers

We have presented a theory on how people deal with meaning and how knowledge can be represented using a semantic network (chapter 2). We have also seen how an extension of the semantic network enables authors to represent knowledge using as many different concepts, relations and attributes as they think necessary. When multiple authors use the same network to represent knowledge, it is possible that multiple representations with the same meaning exist. Groups of elements in the network with the same meaning are called semantically equivalent. This chapter will be about semantic equivalence in a semantic network.

When semantic equivalence is about two groups of elements, the first question we try to answer is: "what can we say about the relatedness of meaning of elements in a semantic network?". To answer this question we need to know what we can say about the meaning of elements in the network (3.1).

Next there is a discussion about different ways one can 'measure' relatedness of meaning in the network (3.2). We will see this gives a lot of hardship. After discussing different measurements, we will concentrate on semantically equivalent representations and semantically equivalent relations in specific (3.3).

3.1 Related Meaning

We are considering a semantic network which will be used by multiple authors to represent knowledge, without having agreements on the meaning of certain concepts, relations or attributes. This way it is very likely different authors will use different representations for knowledge with the same meaning. The next list contains a few ways in which the meaning of two representations can be related:

- There meaning can be the same which is called *synonymy*, for example 'bank' and 'financial institution'
- They can mean the opposite, which is called *antonymy*, for example 'big' and 'small'.
- The meaning of the first can be totally included in the meaning of the second, this is called *hyponymy*, for example 'Yojo' is a hyponym of 'cat'.

Two representations are semantically equivalent, when they are synonymous. The first representation is semantically equivalent to the second when they are hyponymous. Finding semantic equivalence in a semantic network is a matter of finding synonymous or hyponymous representations. This is however not trivial while there are no agreements on the meaning of certain representations in the semantic network.

Semantic equivalence can be seen as a relation two representations have. Representation *A* can be semantically equivalent to representation *B*. Before discussing some suggested methods to measure the relatedness of meaning, it is necessary to first mention 'transitivity' along with two other properties of relations.

'Transitivity' is a property a relation, or to be more specific a predicate, can have. For a transitive predicate P_t holds:

when

relation R_{nmt} exists

and

relation R_{mot} exists

then

relation R_{not} exists

An example of a transitive predicate is 'is larger than'; when 'Jan is larger than Mark' and 'Mark is larger than Rob' then 'Jan is larger than Rob'.

Another property a relation can have is 'generalization'. When there is a generalizing relation from concept C_n to C_m, C_n is a hyponym of C_m.

The last property is 'symmetry'. When a predicate Pp is symmetric, it means for any relation R_{nop} that also R_{onp} exists.

3.2 Measurements of Relatedness

Some definitions of relatedness of elements in a semantic network have been suggested [7]. Two of these definitions are network distance and semantic distance. While the first definition only measure the distance between nodes, without addressing meaning, the latter one does.

3.2.1 Network Distance (ND)

'The network distance between two nodes is the smallest number of relations one could use two come from the first to the second node' [7]

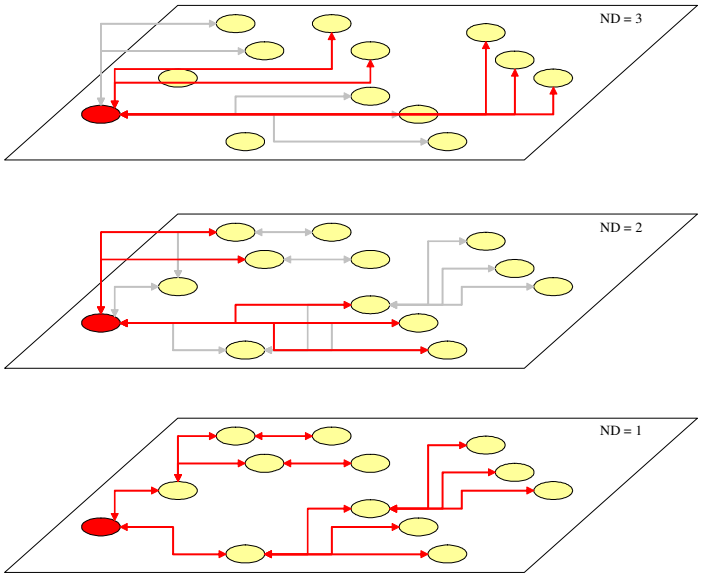


Figure 3.1 Some examples of Network Distance, from [Poell2002]

So the 'network distance' measures the shortest 'path' possible between two nodes. Related nodes in the network have network distance 1. When two nodes C_n and C_m can not be directly connected, but are both directly connected with a third node C_o, C_n and C_m have network distance 2 (see figure 3.1). Note that in this figure, there is no distinction between a connection from C_n to C_m and one from C_m to C_n. This definition will therefore be called undirected, which means there is a simplification of the network whereby we make no distinction between a relation from C_n to C_m or C_m to C_n.

3.2.2 Semantic Distance (SD)

'The Semantic Distance between two nodes is equal to the minimum number of different associations (relations) that are not used in a transitive way: $SD_x(n1, n2)$' [7]

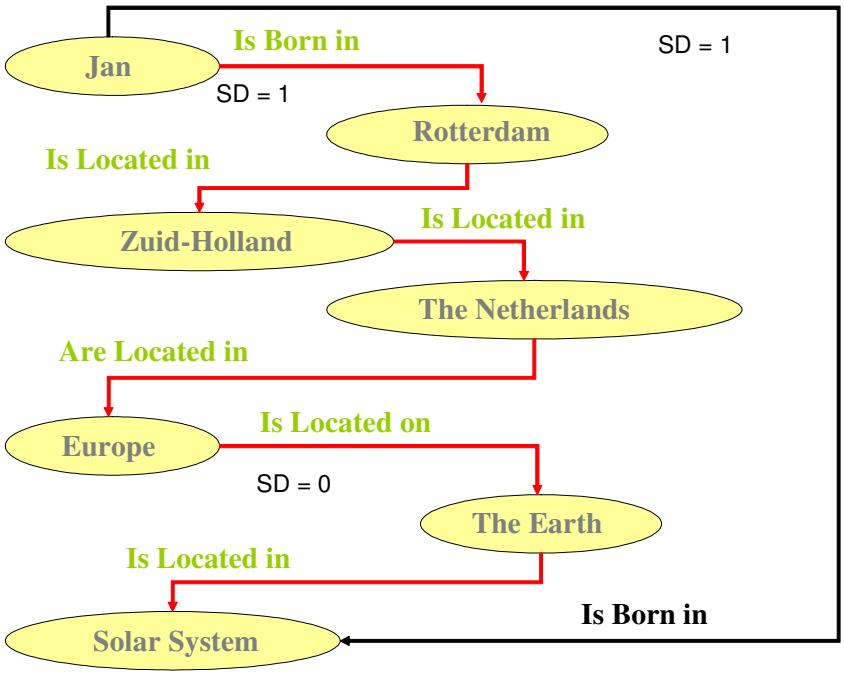


Figure 3.2 Semantic distance in a semantic network, adapted from [7]

So the semantic distance is the network distance, but without counting the relations that are used transitive. In the example semantic network (figure 3.2) the following semantic distances exist:

- SD_1(Jan, Rotterdam), namely the relation "Is Born in" is the minimum relation used.
- SD_1(Jan, Solar System) namely the relation "Is Born in" is the minimum relation used, the relations "Is located In", "Are located in" and "Is located on" can be used in a transitive way.
- SD_0(Europe, The Earth), namely the relations "Is located in", "Are located in" and "Is located on" can be used in a transitive way.

In short, the smaller the semantic distance between two concepts, the closer these two concepts will probably be related.

"*One could say relations between two concepts with distance SD_0 or SD_1 have a semantically significant meaning, while those with SD_2 or higher only have something to do with each other*" [7].

Remember authors are free to use any relation they see necessary. Therefore, they are also allowed to create an "is a concept in the same semantic network as"-relation between all concepts in the network. Hereby it reduces the semantic distance between all concepts to zero.

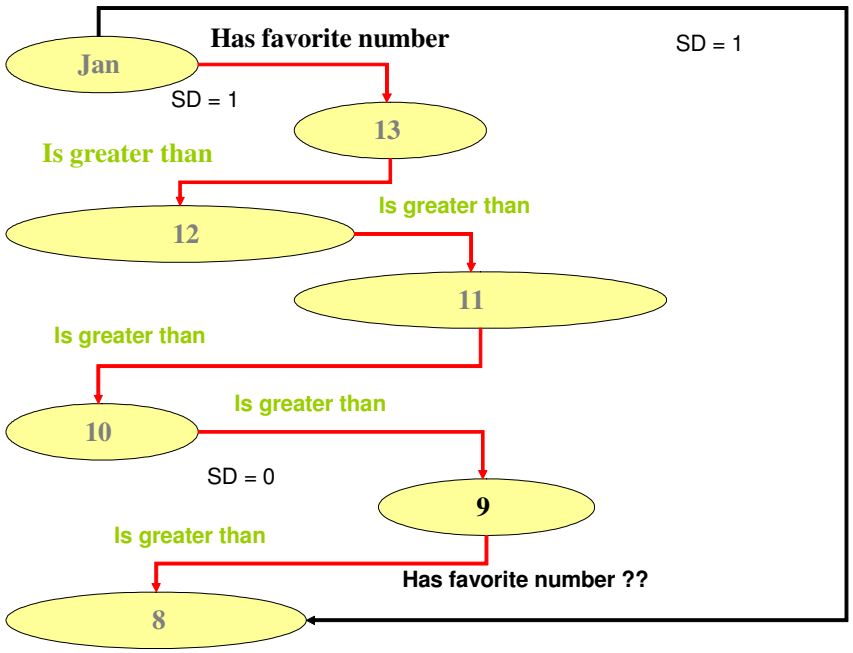


Figure 3.3 Semantic distance in another semantic network

When a representation of an object is not the object itself, it is possible the representation omits information about the object. Therefore it is certainly possible concepts with SD_2 or higher are in fact closely related.

So is it always true a relation between concepts with SD_0 or SD_1 has semantically significant meaning? Consider a semantic network very similar to figure 3.2. The only difference is instead of location, the concepts are numbers and instead of the relation "Is located in", another transitive relation "Is greater than" is used. Assume also that Jan has a favourite number, that number is 13.

In the same way as the previous example, we should be able to say Jan has favourite number 8, but as we see, this would be rather unlikely. The relation 'Jan' has with the number '8' that can be derived from this semantic network is that 'Jan has a favourite number which is greater than 8'. But we can probably make the same kind of derivations for nodes with a SD greater than 1. 'Jan has favourite number 13' is a relation which has only significant meaning when considering the concepts 'Jan' and 'thirteen'. When there are relations between '13' and other numbers, transitive or not, 'Jan will probably have a 'vague' relation with those numbers.

Based on these examples, the following adaptation on semantic distance is suggested. When two concepts have a generalizing relation between them, their semantic distance is zero. So in figure 3.2, the semantic distance between 'Europe' and the 'Earth' remains zero. All other relations between concepts are initially 1, including transitive relations. Semantic distance between concepts which are not directly related can be measured by addition of the semantic distance in between. Hereby placing the remark that semantic distance one added to semantic distance one stays one when the two relations can used in a transitive way.

This measurement results in the following semantic distances. In figure 3.2 the semantic distances stay the same. In figure 3.3 however, it results in the following semantic distances:

- SD_1(Jan, 13)
- SD_1(13, 12)
- SD_1(12, 11)
- SD_1(13, 11), namely SD_1(13, 12) + SD_1(12, 11)
- SD_2(Jan, 8), namely SD_1(Jan, 13) + SD_1(13, 8)

Measuring semantic distance this way still does not consider the context of two concepts. It can only become smaller when more relations are inserted. When considering the context of the concepts, semantic distance can become both smaller and larger by adding extra relations. Two concepts, for example 'Israeli' and 'Palestine' could be semantically close in a geographical context, but semantically far apart in another context.

3.3 Semantic Equivalent Relations

We have seen that semantic equivalence is inherent to the extension of a semantic network to allow multiple authors to work with it. We have also seen the difficulty when we try to make statements about the meaning of elements in the network.

Next follow some examples of semantically equivalent situations that could possibly exist in a semantic network. The examples are divided in hyponymy and synonymy examples. The first examples will be about different types of elements while the last will focus on relations.

Example 1: Total concept-concept synonymy
One author could create the concept 'Bank', which is not seen by a second author who creates the concept 'Financial Institution' (see figure 3.4)

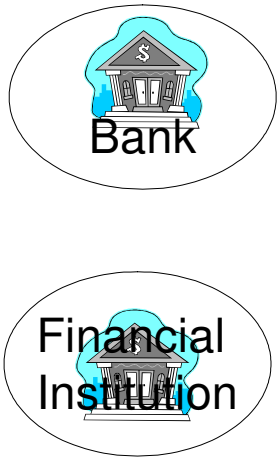


Figure 3.4 The concepts Bank and Financial Institution are equivalent

Example 2: concept-concept Hyponymy
A first author could use the concept bank while a second uses the broader concept company (see figure 3.5). While a bank is also a company, when looking for a 'Company', one could also look for the semantic equivalent 'Bank'.

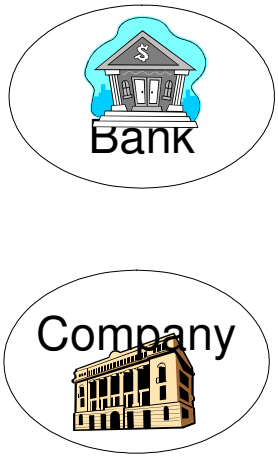


Figure 3.5 The concepts company and bank are semantically equivalent when one is interested in companies only

Example 3: multiple concept – concept synonymy
One author could create the concept 'Torah', while a second describes the books separately as 'Genesis', 'Exodus', till 'Deuteronomy' (see figure 3.6). Note that the first 5 bible books are also known as the "Torah".

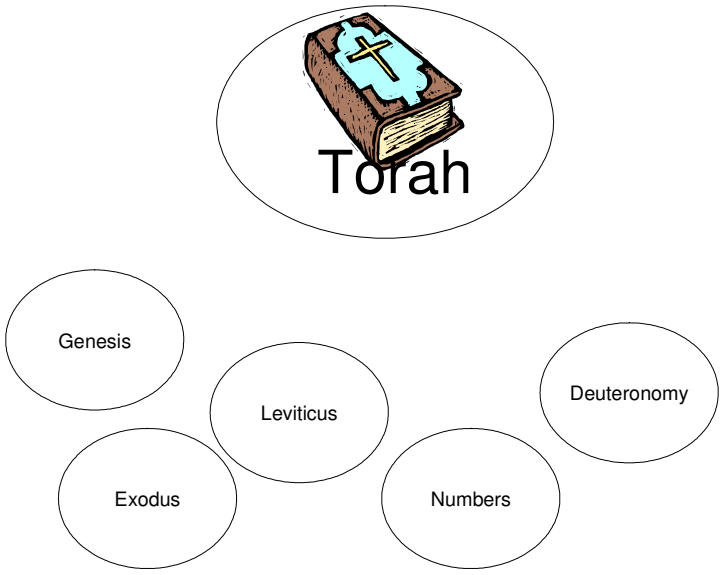


Figure 3.6 The concept Torah could be divided in five diferent concepts

Example 4: relation – concept, attribute synonymy
Every relation could be translated in a concepts with an attribute (see § 2.3.1). Therefore, for every relation created by one author it is possible another author has used a concept and a attribute to represent the same.

Example 5: relation- relation synonyms
One author could represent the relation "Shakespeare is author of Othello" while another finds it more appropriate to represent "Shakespeare has written Othello" (see figure 3.7).

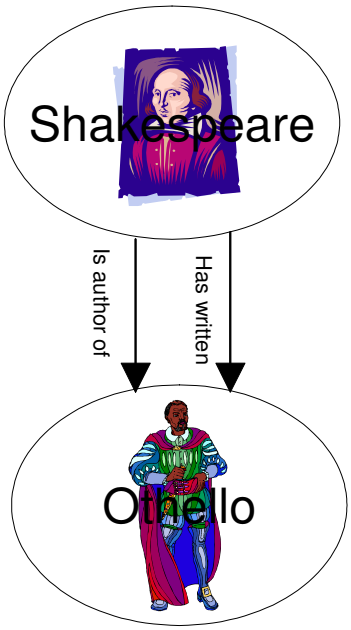


Figure 3.7 Shakespeare has created Othello

Example 6: multiple relation- relation synonyms
Some authors might pay attention to the relation Jim has with his brother John, others might represent the relation they have with their parents (see figure 3.8).

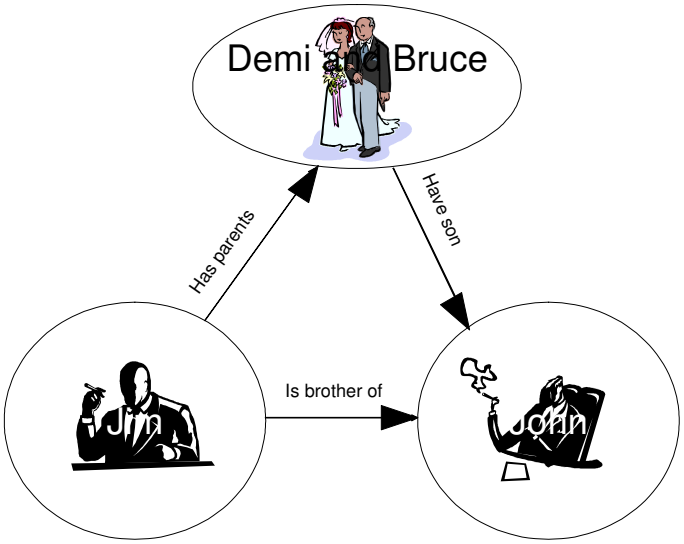


Figure 3.8 A triangular example of semantic equivalence

One can see that by making no agreements on the meaning of elements it is possible to create all kinds of semantic equivalence. However, is it possible the representations give clues about what is semantic equivalent and what is not? This will be part of the next chapter.

4. Searching for Semantically Equivalent Relations

"Ik houd niet van generaliseren, Surinamers doen dat ook altijd."
--Theo Maassen

We have seen that the semantic network holds semantic equivalent relations. In what degree is it possible to automate the search for semantically equivalent relations? In this chapter we will look at conceptual solutions to automate the search for semantically equivalent relations.

To automate the search, three search strategies are being suggested. The first strategy is based on the patterns semantically equivalent relations can have when being described by different authors. The relation two concepts can have can be represented in multiple ways. Different authors can choose the representation differently. A method is suggested based on the number of relations between two concepts. This approach is called pattern based, while multiple relations between concepts can be seen as a pattern (4.1). This method can be extended by using human knowledge about the meaning of elements indicating transitivity and generalization in the network (4.2). Another extension can be made by using the generalization property of relations (4.3).

The second strategy is based on the assumptions that the meaning of an element or group of elements is somehow connected with its representation. For example, a numeric value in an attribute says something about the meaning of the attribute and the name of a concept says something about the meaning of the concept. From this assumption follows that (groups of) elements that have similar representations can also have similar meaning. To compare the representation of elements, they can be translated into a multidimensional vector we call a 'fingerprint'. Elements with a resembling fingerprint can also have a resemblance in meaning. This approach we call association driven (4.4).

It is possible the method suggested in 4.4 is not sufficient to create effective hypotheses on semantic equivalence. Although we assume a representation hold properties of the represented, this does not mean that the representation holds properties about the equivalence of meaning. The suggestion is to use machine learning on 'fingerprints' of elements that are already classified as semantically equivalent or not. This way it could be possible to learn the properties that lead to semantic equivalence (4.5).

4.1 Pattern Based Search

As we have seen in the previous chapter during the definition of semantic distance, transitivity and generalization are two properties in a relation that enlarge the probability of related meaning between nodes. As seen with semantic distance, when a chain of transitive relations is used between two concepts, it can be rewritten as one relation between the two random nodes in the chain.

Let us look at the simplest form this could have. In the simplest form, two authors can both have defined the same association between two concepts. This could happen for several reasons:

- Both authors define the relation at approximately the same time.
- The second author does not realize the relation is already defined by the first author.

- The second author is unaware of the meaning of the relation defined by the first author.
- Both authors define both concepts as well as relation separate and later the concepts are merged.

When two concepts have two associations, this could indicate semantic equivalence of the relations, maybe even equivalent predicates. The pattern such a situation will have is illustrated in figure 4.1.

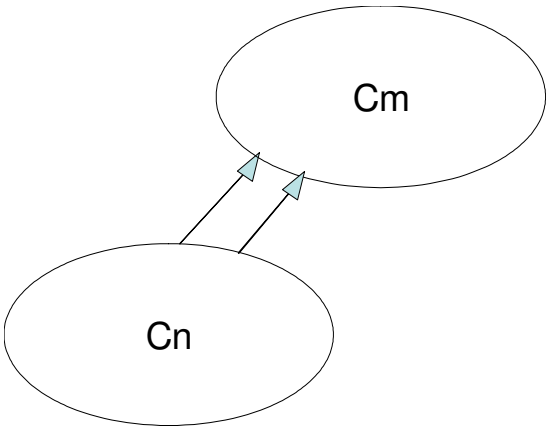


Figure 4.1 Occurence of pattern one

This pattern we call "pattern one", it involves two relations (R_{npm} and R_{nqm}) with the same subject (C_m) and object (C_n) concepts. Examples of pattern one containing semantic equivalence are:

- "Shakespeare has created Othello" and "Shakespeare has written Othello"
- "Shakespeare has created Othello" and "Shakespeare is author of Othello"
- "Shakespeare has written Othello" and "Shakespeare is author of Othello"

In these examples, a computer application which uses the information in the first relation can also use the second relation for the same goal. While the occurrence of pattern one increases the probability of semantic equivalence, the occurrence of pattern one does not by definition indicate semantic equivalence. Examples of the occurrence of pattern one, while there is omission of semantic equivalence are:

- "Shakespeare is author of Othello" and "Shakespeare is not satisfied with Othello"
- "Shakespeare is author of Othello" and "Shakespeare has been rewarded for Othello"

Another simple pattern that is imaginably semantically equivalent happens when two authors describe the same association between two concepts, but define them inverse. By inverse is meant, one author describes the relations with object concept C_n and subject concept C_m, while the other author describes the association with object concept C_m and subject concept C_n. This pattern will be called pattern two and is illustrated in figure 4.2.

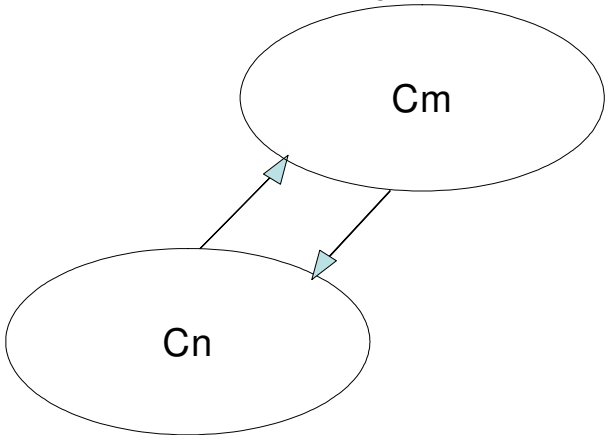


Figure 4.2 Occurence of pattern two

So pattern two involves two relations (R_{npm} and R_{mqn}) with both the same concepts (C_n and $C_{m)}$, but reversibly used as subject and object concept. Examples of pattern two containing semantic equivalence are:

- "Shakespeare has created Othello" and "Othello is written by Shakespeare"
- "Shakespeare has created Othello" and "Othello has author Shakespeare"
- "Shakespeare has written Othello" and "Othello has author Shakespeare"

As also seen with pattern one, the occurrence of pattern two does not by definition indicate semantic equivalence. Following are some examples of the occurrence of pattern two, where there is no semantic equivalence:

- "Shakespeare is author of Othello" and "Othello is being regarded as a failure by Shakespeare"
- "Shakespeare is author of Othello" and "Othello did not make money for Shakespeare"

As we see with both patterns, when predicates relate as hyponyms, relations with that predicate are only semantically equivalent in a certain context. So the a relation using the predicate "has written" does always indicate a "has created" relation, but not the other way around. Consider the relation "Shakespeare has created a lot of garbage", this does not implicate that "Shakespeare has written a lot of garbage" and these relations are certainly not semantically equivalent. Still the occurrence of one of these patterns increases the probability of semantically equivalent relations.

While pattern one and two are based on the simplest representation, there is at least a third pattern that could indicate semantic equivalence. This pattern is based on the assumption that one author will use two relations that go from one concept to another to a third, while another author

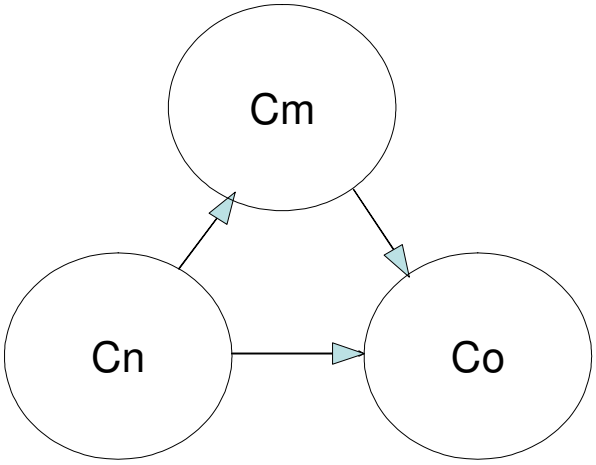


Figure 4.3 Occurence of pattern three

describes the relation direct from the first to the third. This pattern we cal pattern three and is illustrated in figure 4.3.

When using this pattern, the hypothesis is made that the relations R_{npm} and R_{mqo} are equivalent to R_{nro}. This semantic equivalence could be both by definition as through transitivity or generalization. Examples of pattern three representing semantic equivalence are:

- "Jan is son of Sylvia and Rob", "Sylvia and Rob are parents of Kees" and "Jan is brother of Kees".
- "Jan is born in Rotterdam", "Rotterdam is located in the Netherlands" and "Jan is born in the Netherlands".

These occurrences of pattern three represent semantic equivalence, but there are also plenty situations imaginable where pattern three occurs, but there is no semantic equivalence. Examples of these situations are:

- "Jan is son of Sylvia and Rob", "Sylvia and Rob are friends of Kees" and "Jan is a friend of Kees".
- "Jan is born in Rotterdam", "Rotterdam is located in the Netherlands" and "Jan is living in the Netherlands".

When trying to automate the search for semantic equivalent relations, it seems a valid strategy to search for these patterns.

4.2 Knowledge Extension on Search

While it could be a good start to search for these patterns, it is possible to extend these patterns based on transitivity and generalization in the network. When it is clear which relations can be used transitive and which generalizing, it is possible to bring more complex patterns back to one of the simple forms. When we look again at figure 3.2, there is a generalizing relation between the concepts "Rotterdam" and the "Solar System". Because the concept "Solar System" includes the concept "Rotterdam" the association "Jan" has with "Rotterdam" is also valid for the concept "Solar System". This way figure 3.2 can be brought back to pattern 1 (see figure 4.4).

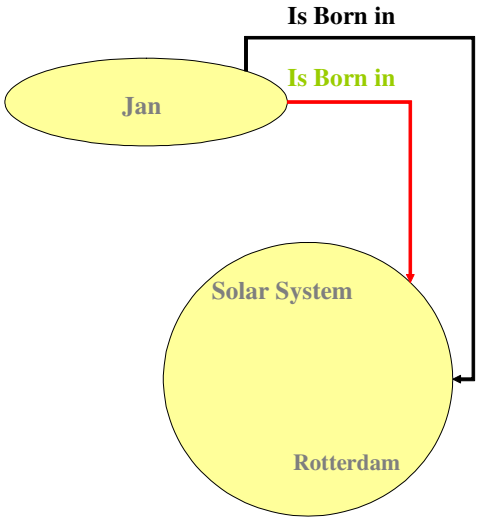


Figure 4.4 Using generalizing relations in a semantic network

While in this example the hypothesis will be “Jan is born in the Solar System” is semantically equivalent to “Jan is born in the Solar System”, it is not unimaginable one of the association would be “has first seen the light in” or “Is given birth to in”.

The same way, a transitive relation (like the one in example 3.2) can be brought back to pattern three. Already we have seen that the creation of a direct association between “Jan” and the number “8” does not sound natural

While using knowledge about transitivity and generalization is an option to extend the search for semantic equivalence, the question how to get this knowledge remains open. The first suggestion is to use human knowledge to indicate which relation has these properties or which elements indicate these properties. Another suggestion is to use a ‘standard’. One could ‘demand’ for every generalizing relation to be indicated with the attribute ‘generalizing relation’. Or one could demand for every two concepts where one is a generalization of the other, to be indicated with a ‘is a generalization of’ relation.

4.3 Generalizing Extension on Search

Another extension on the search for patterns could be based on the usage of generalization relations. Suppose we have knowledge on generalization and find no occurrences of pattern one or two. It is possible to take an association two concepts have and ‘lift’ that association to a more general concept. By doing so, the occurrence of a pattern can be created. Let us for example, look at a situation where there is a relation “Shakespeare has written Othello” and “J.K. Rowlings is author of Harry Potter”. No occurrence of pattern one can be found here. Suppose however both “Shakespeare” and “J.K. Rowlings” have a association with the concept “writer”. In the same manner “Othello” and “Harry Potter” have an association with the concept “writing” (see figure 4.5).

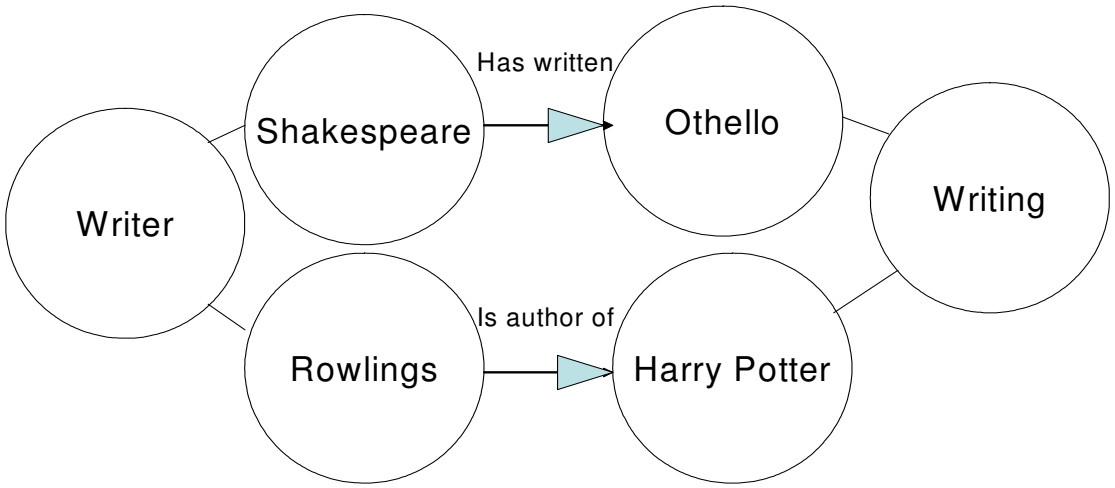


Figure 4.5 Using generalizing relations in a semantic network

For simplicity, the generalizing relations have remained nameless. When two concepts in such a relation share a common super-type, it is still possible the relations are semantically equivalent. This extension we call generalizing extension while it is more likely relations are semantically equivalent when by 'lifting' relations to a super-type a pattern occurs more often.

4.4 Association Driven Search

Another strategy to search for semantic equivalence does not only apply to relations, but should apply to all equivalent elements one looks for. When we assume the meaning of knowledge is showing through its representation, elements that share characteristics in the representation could also share meaning.

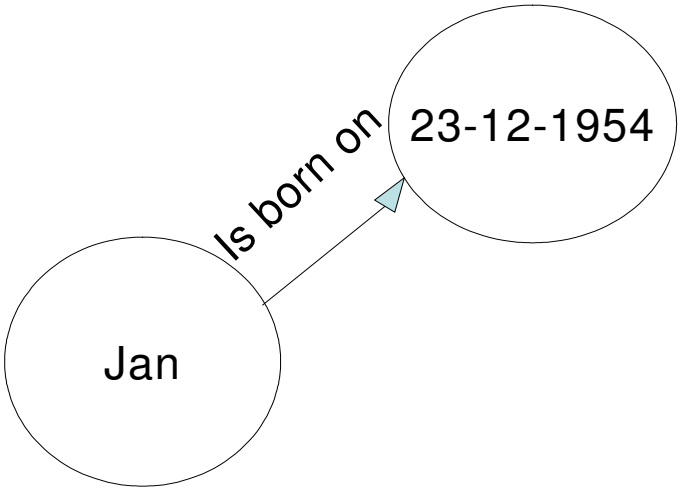


Figure 4.6 Another small network containing one relation

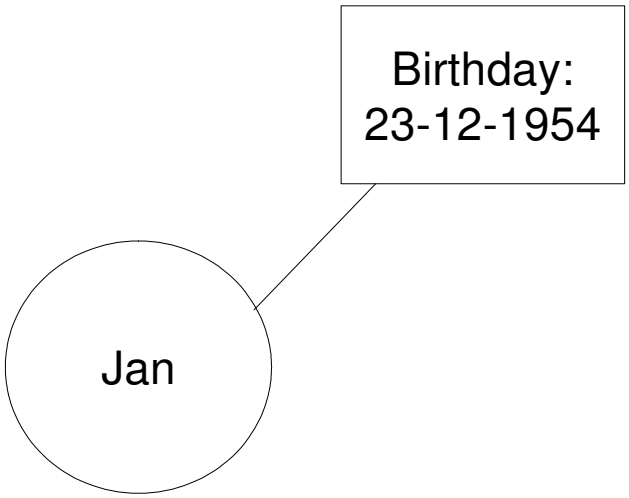


Figure 4.7 Another small network containing a concept and an attribute

Take, for example the situation where a birthday is modeled. One author uses a representation like figure 4.6.While another represents it as in figure 4.7. Although the representations are different, there are indications their meaning is equivalent. This way group of elements could be translated in a vector in multidimensional space, a 'fingerprint'.

In this 'fingerprint" one could put information about the number of concepts, relations and attributes, and about their names and values. Using current AI techniques like Self-Organizing-Maps or Associative Conceptual Space, these fingerprints can be organized, leaving resembling fingerprints close together. Under the assumption representation gives an indication on its meaning, elements with a more or less equivalent meaning will be close together.

4.5 Machine learning

Using a 'fingerprint' of elements in the network could possibly result in a rude organization of meaning. By rude is meant, that 'fingerprints' which are near each other have some resemblance in meaning, but can in no way be called semantically equivalent. This could possibly happen while some elements in the network can indicate semantic equivalence, while others have very little to do with it.

The third approach is suggested to find which elements or values (possibly none) contribute on semantic equivalence in a specific situation and which do not. One can imagine classifying some groups of elements as being semantically equivalent or not. An AI learning technique like a neural network is then fed with the ‘fingerprint’ representation of these elements. This way an artificial intelligence (AI) technique could be used to ‘learn’ which fingerprints are more likely to be semantically equivalent. The AI technique should be able to classify based on previous examples. Examples of such techniques are decision trees or neural networks.

5. Finding Semantically Equivalent Relations

Three strategies for searching semantic equivalence have been suggested. In this chapter, an implementation of the first strategy will be discussed. First we will look at the existing architecture for the semantic network at TNO (5.1). Next some properties of an algorithm that searches semantic equivalence will be discussed (5.2). Finally the pseudo-code used for the algorithm that finds semantic equivalence is given (5.3).

5.1 Architecture of the TNO Semantic Network

Before looking at the algorithm for finding semantically equivalent relations, we first will look at the architecture in which the algorithm will be placed. This architecture is used to implement the current semantic network at TNO.

As part of the technical implementation, the semantic network is split up in a storage part and a semantic network part. The application is split up in a functional part and a visualization part called the 'client'. This makes a technical architecture of five distinct parts. Each part will be discussed separately.

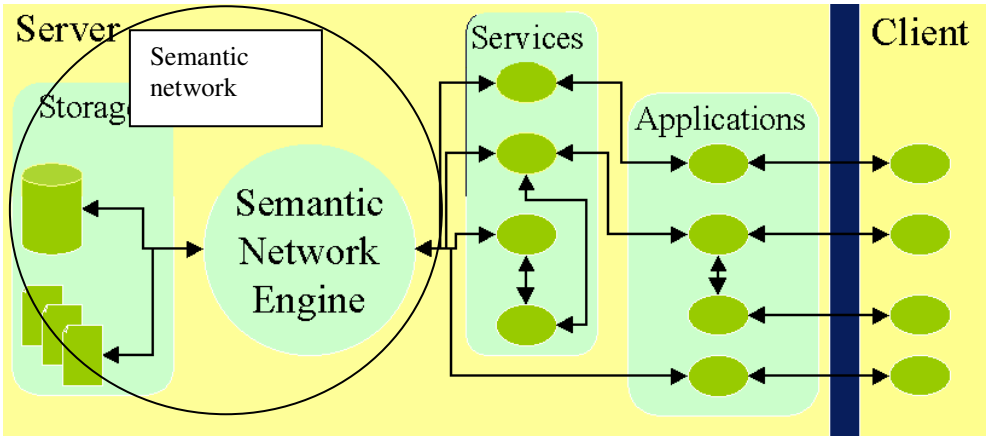


Figure 5.1 Overview of the architecture used to work with a semantic network

Storage

The persistent storage used at the moment is a relational database. In this database, the tuples represent concepts, relations and attributes. While this is the data for the semantic network, it can not yet be accessed as a semantic network. To make this possible, there is need for a piece of procedural knowledge called the Semantic Network Engine. So, a SN is implemented as storage and SNE.

Semantic Network Engine

The Semantic Network Engine (SNE) gives an interface to retrieve, insert and update concepts, relations and attributes from the storage. In the future this is the location that should take care of transaction and security management for writing to and reading from storage. So the SNE and storage are the internal representation of a semantic network in a computer.

Services

Services are a new part here. Services can be used to take over general tasks from applications. One could expect several applications to work with dates stored in the network, when this functionality is implemented as a service, it is easily reusable for different applications. The algorithm for finding semantic equivalence is an example of a service.

Technically speaking, services make function-calls on the SNE as if it is a semantic network. From the SNE they can retrieve concepts and method calls on the concepts make it possible to retrieve the relations defined on it. The concepts and relations retrieved from the network are always a copy from the data in storage and never contain the data itself. Based on the data retrieved from the SNE, a service can build an internal representation of a part of the semantic network. If necessary, the SNE can be notified of updates of the concepts and relations and can apply the changes to the data in storage.

Applications

Applications will have a domain specific goal and will work with knowledge provided by the SNE, services as well as other applications. Applications are equipped with procedural knowledge to enable it to behave as if it knows it. The outcome of this behaviour and the sometimes necessary interaction with user is handled through the client.

Client

The client can be something like a HTML-browser on the computer of the user. It is needed to show the action (or reaction) of a computer application to the user.

5.2 Properties of an Semantic Equivalence Finding Algorithm

When an algorithm for finding semantic equivalence is tested, one can look at its effectiveness on a test set. An algorithm is most effective on a test set, when it makes a hypothesis of semantic equivalence for all semantic equivalence elements in the test set, no more and no less.

Given a set of elements (S), an algorithm to automate the finding of semantic equivalence will be most effective, when it makes a hypothesis of semantic equivalence of all pairs of elements from S (S') which humans will classify as semantically equivalent. Table 5.1 illustrates this. Although it seems as if finding semantic equivalence is only a matter of classification, this is not the case. However, when using a test set, this approach can be used to measure its effectiveness.

Table 5.1 Categories usable to judge the result of an algorithm to find semantic equivalence

	Hypotheses by the algorithm	Hypotheses not made by the algorithm
S' s= S''	Correct hypotheses of semantic equivalence	Incorrectly not made hypotheses of semantic equivalence
S' s!= S''	Incorrect hypotheses of semantic equivalence	Correctly not made hypotheses of semantic equivalence

S can be divided in S' and S'' where for every two subsets S' and S'' (where S' not equals S''), S' can be semantically equivalent to S'' (notated as S' s= S'') or S' could be not semantically equivalent (notated as S' S!= S''). Note that there can be a very large number of subsets S' formed from S, it is however not necessary for an algorithm to compare all these subsets with one another, a heuristic may also be used which discards most of the subsets at once. Pattern based search is such an algorithm which only tries to find semantically equivalent relations.

5.3 An Algorithm to Find Semantic Equivalence

The first remark that will be made on the algorithm is that it will only find semantically equivalent relations when the duplicate concepts in the network, which are put there by different authors, are merged. The second remark is the occurrence of pattern two is only of interest when authors have chosen to represent an association only one time from object to subject ("Shakespeare has written Othello") and not also the other way around ("Othello is written by Shakespeare"). If association is represented both ways, a hypothesis based on pattern two is also made based on pattern one.

The service which will make use of this algorithm will use a reporting implementation. (see chapter 3.4). Here after the pseudo code for finding the patterns is given.

Pattern one, based on which the service makes a hypothesis of semantic equivalence, is based on two associations between the same subject- and object concept. The pseudo-code for finding pattern one is:

When considering concept C_n.

Retrieve all relations where C_n is the object node, call this R_{n--}
Count the number of relations in R_{n--}, call this N

For i = 1 to N-1 (the first relation till the last minus one) {

For j = i+1 to N (the relation after the ith till the last one) {

Retrieve the subject node of the i^{th} relation in R_{n--}, call this C_m
Retrieve the subject node of the j^{th} relation in R_{n--}, call this C_o

If Cm is the same concept as Co {
Report hypothesis R_{n-m} s= R_{n-o};
}
}
}

By searching the relations on a concept this way, each semantic equivalent set of relations will be counted only once.

Pattern two, is based on a association between the subject- and object concept and one from where the subject is the object concept and vice versa. The pseudo-code for finding pattern two is:

When considering concept C_n.

Retrieve all relations where C_n is the object node, call this R_{n--}
Count the number of relations in R_{n--}, call this N

For i = 1 to N (the first relation till the last) {
Retrieve the subject node of the i^{th} relation in R_{n--}, call this C_m

Retrieve all relations where Cm is the object node, call this R_{m--}
Count the number of relations in R_{m--}, call this M

For j = 1 to M (the first relation till the last) {

Retrieve the subject node of the j^{th} relation in R_{m--}, call this C_o

If C_n is the same concept as C_o {
Report hypothesis R_{n-m} s= R_{m-n};
}
}
}

By searching the relations on a concept this way, each semantic equivalent set of relations will be counted once, unless one or both associations are defined both ways. In this case the association will count R_{n-m} s= R_{m-n} as well as R_{m-n} s= R_{n-m}.

Pattern three, is based on an associations between three concepts. The associations in this pattern form a triangle. The pseudo-code for finding pattern three is:

When considering concept C_n.

Retrieve all relations where C_n is the object node, call this R_{n--}
Count the number of relations in R_{n--}, call this N

For i = 1 to N (the first relation till the last) {
Retrieve the subject node of the i^{th} relation in $R_{n—}$, call this C_m

Retrieve all relations where C_m is the object node, call this $R_{m—}$
Count the number of relations in R_{m--}, call this M

For j = 1 to M (the first relation till the last) {

Retrieve the subject node of the j^{th} relation in $R_{m—}$, call this C_o

For k = 1 to N (the first relation till the last) {
Retrieve the subject node of the i^{th} relation in $R_{n—}$, call this C_k

If C_k is the same concept as C_o {
Report hypothesis R_{n-m}, R_{m-k} s= R_{n-k};
}
}
}
,

The number of times a hypothesis is made here on the same concepts depends on the number of associations that are defined in both directions between the concepts.

5.4 When Relations Are Found

Whenever semantic equivalence is found, the next question is what the service should do with it. The first thing it always should do is report the made hypothesis of semantic equivalence. Reporting the hypotheses seems necessary to enable humans to judge about the semantic equivalence. Based on this report, applications and services using the relation which has an equivalent relation could be updated to work with both of these elements. This however, seems to be very labor intensive. Here we consider two other options.

One option is to alter the network when finding semantic equivalence. After a report of semantic equivalence, it is possible to replace the equivalent elements with the other. By doing this, only the applications and services using the replaced elements should be updated to use the other elements instead. While this could be used for totally equivalent elements, it should not be used for others. As we have seen, meaning can be very subtle, by replacing one element with another, subtle differences can be lost.

Another option is to use a set of elements in the network that indicate semantic equivalence. This way whenever semantic equivalence is found in the network, it could be indicated by for example a relation between the groups of elements. We could suggest the predicate “is semantically equivalent with” which can be used between semantically equivalent concepts or groups of concepts. We could also store the semantic equivalence separately in an attribute. While there is actually no restriction on how semantic equivalence should be indicated,

it should be able to indicate semantic equivalence for all possible semantically equivalent elements and the restrictions under which the equivalence holds.

The second option is probably the most desirable. Semantical equivalence depends on context, this way it would in some cases be incorrect to replace one element with another. The development of elements that indicate semantical equivalence seems the best option to enable applications to work with it.

6. Experimental Results

A method for finding semantically equivalent relations was introduced. In this chapter, we will look at the current content of the semantic network on which our proposed pattern search algorithm will be used (6.1). We will see the semantic network actually contains four separate networks.

First results of the semantically equivalent relation finding algorithm on parts of the network made it clear that there was a need to combine sources (6.2). The combination of the different sources was done by merging duplicate concepts from different domains. The merged concepts appeared in similar relations with other concepts. These relations were gathered in a test set.

The result of the algorithm on the duplicate concepts is given (6.3) followed by a discussion of these results (6.4)

6.1 Content of the TNO Semantic Network

To fill the semantic network, TNO tries to import large sources of data. These sources also use a symbolic approach to represent knowledge. This far, four large sources are imported in the semantic network:

Notion system

Notion System is a knowledge base allowing the storage and retrieval of any kind of knowledge you would like to store for later use (www.notionsystem.com). The data in Notion System forms a semantic network. This semantic network is based on mostly the same principles as the semantic network used here. Notions system contains approximately 210.000 concepts, called notions in notions system.

Paranoid

Paranoid is a way to structure unstructured intelligence data. The data is stored in a relational database. It uses a small set of predicates (is related with, …) to relate different pieces of data. This data is always a sub-type of a small set of general types (person, event, location, and some others)

FEL-Intranet

The information for employees of a division of TNO, the division FEL, is also structured according to a small set of predicates. On the FEL-intranet, information is stored together with the information on the context it is created in.

WordNet

WordNet provides an online dictionary with the descriptions not being words, but being ‘synsets’ (a collection of synonyms). The words in a synset refer to the same concept.

Each source contains knowledge on several domains. In some domains knowledge comes from different sources. The domains and the sources they are handled in are listed in table 6.1.

Table 6.1 overlapping domains from the different sources

Notion System	WordNET	Paranoid	FEL-Intranet
Animals	Animals	Persons	Persons
Plants	Plants	Events	Projects
Locations		Locations	

6.2 The Need to Combine Sources

After running the algorithm several times at random places in the network, it has shown that there was no occurrence of pattern one and large occurrences of pattern two and three. It is possible to retrieve the nodes in the network by number. The algorithm was fed series of 50 nodes from different places in the network. Starting at node number 1 and increasing by 20.000. The results are shown in table 6.2.

Table 6-2 Occurences of the patterns on different places in the network

Startnode	1	20001	40001	60001	80001
pattern 1	0	0	0	0	0
pattern 2	228	212	388	516	748
pattern 3	210	190	224	234	272

Here we see that there is no occurrence of pattern 1. The lack of occurrence of pattern 1 could be explained by looking at the 4 different imported sources. In each of these sources we see that two concepts are only in one relation. For example, when in one source the defined "Shakespeare has written Othello", they do not also define "Shakespeare is author of Othello".

It seemed pattern 2 occurred this much, because almost all relations are defined along their inverse relations. Let Pn be the predicate 'is written by" and predicate Pni its inverse predicate "has written". This way every time a relation Rnpm was defined, Rmpin was also defined. This means every time a relation was defined, pattern two could be found.

The large occurrence of pattern three was also unexpected. After studying which relations formed pattern three, none of them appeared to be semantically equivalent. When a semantic network is filled with a lot of structured content with few relations on each concept, occurrences of pattern three seem rather common. Here for the search for pattern 3 is stopped, while this does not seem to result in semantical equivalence.

The search for pattern 1 and 2 also does not result in semantically equivalent relations, some altering of the network could however change this. There is a need to combine sources in the network, while without it, there is no semantical equivalence in the network.

So not finding pattern one could be explained by a wrong assumption on the network, namely a network filled with content made by many authors where duplicate concepts would be merged. The current network contains only four different sources, with overlapping domains. The duplicate concepts however, were not merged. Another reason is that each of these sources uses a set of predicates that are normalised. Authors of a source do not use two different predicates that express the same meaning. So within one source, "Shakespeare has written Othello" and

"Shakespeare is author of Othello" will not occur. So this in fact leaves two authors for each domain to create possibly semantic equivalence.

So to find different predicates that express the same meaning, one has to look in different sources at the same time. When two sources discuss the same domain, it is quite likely they use the same concepts and describe the same relation between them, only with different elements.

To recognise two concepts are identical, a service could be developed. For this experiment however, finding duplicate concepts and merging them, will be done by hand.

Both sources Notion System and WordNet handle the domain animals. In this domain, there was a search for the Latin name of a animal. This search sometimes resulted in the finding of two concept, the animal described in Notion System and the animal described in WordNet. These found concepts where merged to form one concept. Between these merged concepts were semantically equivalent relations. These relations are listed in appendix A. Eventually there were 36 nodes to merge, which left 18 nodes. The algorithm was tested on these 18 nodes to see if it could find the semantically equivalent relations from appendix A.

To recapitulate, the following steps where taking to come to the results:

- We have taken a series of n (n=50) nodes from the semantic network.
- Fed these n nodes one by one to the algorithm to let it look for pattern 1,2 and 3.
- This was done at 5 different places at the network with steps of 20.000, from node 1 till 80.001.
- While this did not result in the finding of semantical equivalence, synonymous concepts from the different imported sources were searched for and the search for pattern 3 was dropped.
- There was knowledge about the domains the sources used. In the domain animals, a search for the Latin name of an animal sometimes resulted in duplicate concepts from different sources.
- These duplicate concepts were merged.
- The merging of these concepts resulted in some semantically equivalent relations between merged concepts.
- The merged concepts were fed to the algorithm, which looked for occurrences of pattern 1 and 2 on each concept.

6.3 Results

In the table below, the number of times the algorithm reported the occurrence of a pattern is listed.

Table 6-3 Results of the algorithm on the merged concepts

Pattern	Number found
1	18
2	237

6.4 Discussion of Results

As was expected after inspecting the test set by hand, all occurrences of pattern 1 were found. The result gives 18, while there were 18 nodes merged. On every of these 18 nodes was a semantically equivalent relation. So while appendix A only list 11 relation sets, the relation in each set were also valid for their inverse relation.

From the 237 times pattern 2 is found, for 201 of them it is already indicated in the network that they are an inverse relation. The occurrence of pattern two is twice the occurrence of pattern one minus the number of relations that are indicated as inverse in the network.

The algorithm can be therefore be seen as a pattern based search with some knowledge extension to indicate inverse relations. The performance of the algorithm on the test set can therefore be said to be most effective while all semantic equivalent relations were found and no false hypothesis was made.

7. Conclusions and Future Research

In this final chapter we will make a summary for the thesis (7.1). This summary will be followed by the conclusions that can be drawn from the research (7.2). Based on these conclusions, suggestions for future research will be made (7.3)

7.1 Summary

In the beginning of this thesis we introduced semantic networks as a way to represent knowledge. To enable many authors to use the same semantic network to represent their knowledge, an extension on the network was introduced. This extension made it possible to use any concept, relation or attribute an author finds necessary to represent his knowledge. On the other hand, while there were no agreements on specific relations or concepts to use, it became more difficult to determine the meaning of elements in the network.

Without boundaries on the elements to use, multiple authors could use different representation for the same knowledge. A semantic network that is sufficiently large will probably hold semantically equivalent representations. It is useful to find these semantically equivalent elements, while semantic equivalence gives an indication to computer application what certain representations mean. Hereby enabling these applications "to act as if they know" more. We focused on a specific type of semantical equivalence, namely semantically equivalent relations.

After defining semantically equivalent relations and the need to look for them in a semantic network, three main strategies to search for them were suggested. The first was based on patterns, the latter two on the translation of the symbolic representation into a multi-dimensional vector representation. The first strategy is specific for the search for relations, the others could also be used to search for other semantical equivalence.

The first of the three strategies was implemented. In the implementation, the number of times a pattern occurred was counted and the found patterns were listed. This algorithm was tested for effectiveness on a set of nodes from the network. This resulted in finding of the pattern 2 and 3, but the found relations were not semantically equivalent. The search for pattern 3 was dropped. Some concepts in the network were merged. The algorithm was used on the merged concepts.

7.2 Conclusions

In introduction of this thesis, the research question was given, namely

To which extent is it possible to automate the finding of collections of semantically equivalent relations in a semantic network?

We see now that this question was too much to handle in one thesis. One of the first conclusions that can be drawn from this research is the difficulty computers have to deal with meaning. Using a symbolic approach one is tempted to see the meaning of a symbol as something clear and

sharply carved. A cat is a cat and Shakespeare is Shakespeare, but what do these concepts really mean? The Oedipus-example (see figure 2.8) has shown conclusions are not always that easy to make. Did Oedipus marry his mother? This all depends on the context.

What does the concept 'cat mean? Does its meaning lie in the difference between a cat and other object, for example a couch? Can one distinguish a 'cat' from a' couch' when one 'knows' a cat can be stroked and one can sit on a couch? Some people could prefer the other way around, in which case they need some psychiatric help, but a 'cat' stays a cat. The point being made here is that one needs a lot of knowledge to 'know' the meaning of a concept. Semantical equivalence could exist in every aspect of this knowledge. When someone is manager of a department, it probably also means he has knowledge if the department.

Although we have been able through to some experiments to develop an algorithm that can find the semantically equivalent relations on concepts presented to him, this is just the tip of the iceberg. The theory presented in this thesis points out that semantical equivalence is far more complex than the search for patterns. The search for patterns is however a good start and results in the discovery of the same semantically equivalent relations on some merged concepts as we have done by hand.

To answer the research question; we do not yet know to which extend it is possible to automate the finding of semantic equivalence in a semantic network. Perhaps there is need for rules and definitions about the meaning of certain elements. These rules and definitions can than be used to search for semantical equivalence. The results of the experiment of so far encouraging to do more research on the topic.

7.3 Future Research

One of the first things that could be researched is how to find duplicate concepts in the network. It is merely impossible for a semantic network with more than 200.000 concepts to search for duplicate concepts by hand.

Furthermore, semantic equivalence depends on the goals of man or computer, something which is not yet used in the current algorithm. When the goal of someone is to find out where Jan is, it is irrelevant to him if he is standing, sitting or lying down the hall. Standing, sitting and lying are semantically equivalent when one should act *as if he knows* where someone is. This goal is closely connected to the term context. Two representations can be semantically equivalent depending on their context, one can also say, depending on the goal of the searcher.

To use this goal (context) based semantic equivalence, one should at least have a minimum set of agreed elements which one can use to express their goals. Based on this expressing of the goals can be decided which elements are semantically equivalent. It seems however that this is a shift of the problem of meaning to the set of goal related elements. Because when someone can make clear to you what his goal is, and you know how he can reach this, why not give tell him directly?

The search strategies suggested in chapter 4 do not take context into account. Association driven search and machine learning however need further research, while they seem to have a strong ability to deal with the fuzziness of meaning.

References

[1] A. Analyti, and N. Spyratos and Panos Constantopoulos, *On the Definition of Semantic Network Semantics,* Fundamenta Informaticae, Vol. 36, Nr 2-3, pp. 109-144, 1998.
http://citeseer.nj.nec.com/analyti98semantics.html

[2] R. Davis, et. alt, *What is a Knowledge Representation*, AAAI AI Magazine, Vol. 14, Nr 1, pp. 17-33, 1993.
http://www.aaai.org/Resources/Papers/AIMag14-01-002.pdf

[3] R. Gaizauskas, and K. Humphreys, *Using a semantic network for information extraction*, Journal of Natural Language Engineering, Vol. 3, Nr 2-3, pp. 147-169, 1997.
http://citeseer.nj.nec.com/92663.html

[4] E.L. Gettier, *Is Justified True Belief Knowledge?,* Analysis 23, pp. 121-123, 1963.

[5] J. MacCarthy, *Programs with common sense*. Proceedings of the Teddington Conference on the Mechanization of Thought Processes, 1958.

[6] C. K. Ogden, and I. A. Richards. *The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism*. New York: Harcourt Brace, 1923.

[7] R.A. Poell, *The semantic network of IKM-I3*. Presentation at the Knowledge Technologies 2001.
http://www2.gca.org/knowledgetechnologies/2001/proceedings/26

[8] A.S. Poeran, *De visualisatie van een semantisch netwerk,* Master thesis Computer Science. Leiden University, 2002.

[9] M. Schuemie, *Associatieve Conceptuele Ruimte, een vorm van kennisrepresentatie ten behoeve van informatie-zoeksystemen,* Master Thesis Erasmus University Rotterdam June 1998.

[10] A. Th. Schreiber, et. alt, *Knowledge Engineering and Management, The CommonKADS Methodology*, MIT Press ,1999.

[11] J.F. Sowa, *Semantic network,* Encyclopedia of Artificial Intelligence*,* New York, 1987.

[12] C. Welty, *An Intergrated Representation for Software Development and Discovery*. Ph.D. Thesis, Rensselaer Polytechnic Institute 1995.

[13] M. Zandee, *Syllabus Algoritmiek voor biologen*, University of Amsterdam (UvA), 1996.
http://wwwbio.leidenuniv.nl/~zandee/algo/syllabus/index.html

[14] M.R. Quillan, *Semantic memory*, in M. Minsky, Semantic Information Processing, Cambridge, MA: MIT Press, 1968.

[15] P. Resnik, *Using Information Content to Evaluate Semantic Similarity in a Taxonomy*, IJCAI, pp. 448-453, 1995.
http://citeseer.nj.nec.com/resnik95using.html

[16] A. Budanitsky, *Semantic Distance in WordNet: An Experimental, Application-oriented Evaluation of Five Measures*,Workshop on WordNet and Other Lexical Resources, in the North American Chapter of the Association for Computational Linguistics NAACL-2000, Pittsburgh, June 2001.
http://citeseer.nj.nec.com/budanitsky01semantic.html

[17] N. Guarino and C. Masolo and G. Vetere, *OntoSeek: Content-Based Access to the Web*, IEEE Intelligent Systems, Vol. 14 , Nr 3, pp. 70-80, May 1999.

Appendix A

Semantically Equivalent Relation Sets

Here follows the set of relations used to test the semantically equivalent relation finding algorithm. It is a numbered set, containing 11 relations-sets. In every set, the first relation is equivalent to the second and vice versa. This set was found by hand in the semantic network at TNO in its state during June 2002 till July 2002. These set is not supposed to be exclusive. It is however expected to be representative while all other semantical equivalence found appeared to involve the same predicates.

1. Pongo pygmaeus is member of Pongo
   orangutan appartient aux Pongo

2. Physeter catadon is member of Physeter
   Physeter catadon appartient aux Physeter

3. Macropus giganteus is member of Macropus
   Macropus giganteus appartient aux Macropus

4. Cetorhinus maximus is a member of Cetorhinus
   Cetorhinus maximus appartient aux Cetorhinus

5. Alopiidea comporte Alopius
   Alopiidea, has member Alopias

6. Aquila chrysaetos is een Aquila
   Aquila chrysaetos is a Aquila

7. Betulacea comporte Carpinus
   Betulacea has member Carpinus

8. Betulacea comporte Ostrya
   Betulacea has member Ostrya

9. Betulacea comporte Corylus
   Betulacea has member Corylus

10. Betulacea comporte Betula
    Betulacea has member Betula

11. Betulacea comporte Alnus
    Betulacea has member Alnus