Information philosophy for the Semantic Network

Ronald Poell

CTO Semantic Network Technologies, Netherlands Organization for Applied Scientific Research (TNO)

Version 1.0, 14/07/2009

Abstract

In the multi-domain multi-lingual Semantic Network realized at TNO there is only a high level abstract data model. This model allows the existence in the network of all kinds of information and new kinds to be added at any time.

In the perspective of long term usage of the available information, its correct interpretation, and its automatic maintenance, some precautions have to be taken when creating the information. Automatic maintenance becomes more and more relevant as time and money is lacking to merge new information with the ever growing existing pool.

This paper gives a description of our observations on the information in the network including its meta-data. Most of the provided elements come from rules of thumb or derive from common sense and are based on practical experience.

Contents

1 Introduction

In traditional information systems (databases) the data model is designed in most cases with a particular application (usage) in mind. In the Semantic Network of TNO there is only a high level generic data model that allow all kinds of information to exist. There is no a priori intended usage of whatever information is available in the network. We construct the content when we have the possibility and in the semantics of the source. This results in a multi-model multi-author network in which the same kind of information can be available in different forms (and granularity).

This perspective allows us to investigate the ins and outs of manipulating information in its most complicated form. That should open a way, if not *the* way, to automatic information maintenance.

When deciding to add a piece of information in the network we consider only the context in which it is available and the form is has in its source. We respect in most, if not all, cases the semantics of the source.

Our ideas about what the contents of a global semantic network might look like go back to the late eighties and early nineties [19]. At that time it became already clear that traditional relational databases would probably not be able to fulfil the total information need, in particular with a multi-lingual, multi-domain and historical point of view. Although relational databases were and are perfect for restricted domains and a particular use, beyond that limited scope, they create more problems than that they solve, at least in the way they are generally used. Since that time the practical use of precursor of the actual system (Notion System) showed some clear do's and don'ts and helped to forge ideas about what seems a good practice for the way to store information in a semantic network.

Although this paper is written with the historical Notion System and the actual realization at TNO in mind, the ideas are applicable to similar concepts in this domain: RDF [15], Topic Maps [18] and Cyc (OpenCyc) [17].

This paper reflect observations on information with clearly in mind that we need to achieve a high degree of automatism in the near future. Automatism in "understanding" the information and in the evaluation of its quality, trustworthyness, relevance, etcetera.

For the sake of clarity of the text of this paper we will use specific names for the few different elements in the network: nodes, properties and relationships. In other domains, nodes will be called topics (Topic Maps), concepts (conceptual network) or resources (RDF). Properties or sometimes called attributes. Relationships are more of less equivalent to associations (Topic Maps) or to some of the triples in RDF. It would take a much longer explanation to give the, sometimes very subtle, differences between the elements in various domains, and that is not the purpose of this paper.

During endless discussions about the differences between data, information, knowledge and sometimes even wisdom, it occurred that this is not really important for the subject we are handling here. In this paper we will generally use the term "information" and in some rare occasions "data".

Section 2 describes some basic ideas and requirements that are important when realizing an implementation. Section 3 covers the conceptual information model of the semantic network and some basic aspects related to it. Section 4 illustrates the essential operations on information in the network and section 5 handles some special kind of nodes. Section 6 deals with detail information that needs special attention. Information properties are handled in section 7.

2 Basic ideas

One of the major requirements of the semantic network information architecture is that *information can be added to network in the semantics in which is available, at the moment it is available and with the sole condition that it appears worth to be remembered for someone somewhere in the future.* There should be no restrictions imposed by a chosen technical implementation. Practically there are though some restrictions. To mention only one of them: A person who wants to put the information in the network (not necessarily the creator of it) must have time (read money) to effectively put it in (or to create the software to do it automatically).

Another important requirement has been in the past (and still is) not some theoretical ideal situation but the *real world information* where contradictory information is common and where some, or perhaps most, information cannot be uniquely associated with a particular domain. Where my truth might not be your truth and where your overall view does not see the microscopic details I have available.

Perhaps the most important initial requirement was that the semantic network should contain *information in way human beings gather and manipulate it*, including maintenance of it over time (seeming to forget it if necessary). Actually the requirement is somewhat broader: *information in a way human beings and software agents gather and manipulate it.* The reason for this extension has flown from practical experience that showed the benefits of having all meta data also available in the network. The multilingual aspect of the network advantages e.g. international user interfaces if the textual interface components are also elements in the network[1]. Several consequences derive from these statements.

First the network will be *multi-author and multi-model.* There is no direct consequence relationship between multi-author and multi-model. Even in a single-author network the view on how some kind of information should or might be modeled can differ over time. And sometimes eventually agreements exist among different organizations on how things should be modeled. But in most cases different sources with (almost) the same kind of information will have been modeled with slightly different implicit or explicit semantics associated. Respecting the original semantics will often also mean respecting the original model. The *mapping* between the available model elements (ontology mapping) is not the responsibility of the creator of the content (early model binding) but of the applications that use the content for a particular purpose (late model binding). The rationale for this is based on our opinion that it is only the application that can define which view it should have on the available information. Some views might impose cardinality constraints, others might need high level abstractions (see section

[1] In this example the term *software agents* used also includes applications.

3.6).

The *easy way*, from our point of view the *wrong way*, is to put these, and other, application constraints in the information model. This is good practise in a controlled or closed application environment, however it does not hold for an open real world environment where things are not as nice and beautiful as normalized data would suggest.

Second humanity is *multi-lingual and multi-cultural*. As the network reflects (a part of) our world-wide knowledge it should be able to deal with multi-lingual and multi-cultural aspects. The example of the "leg" in section 6.1.1 shows an example of which kind of complications to expect in a multi-lingual environment. An example of the multi-cultural aspect is the color associated with death which is in many countries black but in other ones it is white.

Third the network will be *multi-domain*. In the world of data-modelers and ontology builders the focus is very often, and with good reason, on a specific domain. But the information in the big real world can not fit in *separated* boxes, it is highly intermingled. An ontology about ecology in its original sense — the relationship between an organism and the environment — should contain elements of (or probably cover totally) the domains: biology, climatology, geology, hydrology, pedology, sociology, and psychology (and perhaps some more). What can actually be observed is that in various ontologies some kinds of "classes" occur frequently, sometimes extending existing ones from other ontologies, sometimes defining entirely new ones. We expect this phenomena to be explosive and further more we expect to see multiple ontologies for the same thing due to a "not invented here" syndrome. If we would like to handle all the populations of these ontologies, we have to face ontology merging and mapping.

Fourth *each peace of information is "true" or "valid" in specific context*. In many cases this context can be reduced to a four dimensional space (geographic and time[2]). When looking at our world knowledge many things appear not to be true (or to make no sense) at some points in time or in some places. Saying that "horses have legs" (or « les cheveaux ont des jambes »– French) makes no sense in the Cretaceous geological period as horses didn't exist at that time. The implications of this apparently trivial statement are quite important. When looking at the network in order to find the "truth" (or the valid information) about a subject you do it (more or less explicitly) from within a certain context. Often this context is composed of at least the current point in time (actuality) and often your actual task should also be associated. What you see when you look at a astronomical picture are states of astronomical objects in an extreme large range in time. Only monogamic marriages exist in most western countries but many countries also allow

[2]See for more detailed aspects related to geography section 6.3 and for time sections 6.2 and 7.2

polygamic marriages. An example of another kind of context is the validity of nicknames where the context is commonly a community of people.

Information should never disappear (be overwritten) from the network but should contain the appropriate annotations about the context in which it is (or was) valid. In case the network is used to draw conclusions of some kind, which conclusions should be justifiable anytime afterwards, even the state of the network at some moment in time can be considered to be a context. Errors during the creation of information in the network should in this case not be corrected by throwing it away or overwriting it, but by invalidating it at the moment of correction.

Now that we had a quick view on some of the aspects we will have to face with this information architecture, we will see in the next two sections the general model of a semantic network and how to manipulate the information in it.

3 Semantic Network model

The information model we handle is the result of the search for simplicity in the model and for simplicity of the manipulations of the information in the network. There is a great similarity with the ideas expressed by Stefen Wolfram in his book "A new Kind of Science"[22]. One of the things he expresses is that complexity and randomness can be obtained through the application of a few simple rules with relatively simple initial conditions. The ideas behind the model of the semantic network can be formulated in a similar way. What is the most simple information architecture that allows a representation of our complex world and what will be the minimum complexity of the operations on these data to be able to draw elaborated conclusions?

First some essential aspects of a semantic network as we see it will be treated. What will be handled next are its impossibilities and the difficult question of concrete or abstract nodes. The following parts will handle some other aspects belonging to the overall information architecture: predicates, property types and finally cardinality.

3.1 Essentials

It was in the late eighties and early nineties that I forged the fundamental ideas on the actual semantic network architecture. The need for it came from a very simple requirement. As my domains of interest are very broad, I wanted to have a *one* system in which everything I would like to retain for future use could be stored, regardless of what is was about. But at that

time the relational database did not provide the flexibility I needed[3]. Some of the ideas came from a thorough introspection and discussions with other people. How do we do things with regards to retaining and remembering information? Why some topics are so tightly related to other topics in our mind? What happens when we are wandering through a cabinet file with bibliographic references? These, and a lot more questions, and trying to find the answer to them, lead to the information concept of a semantic network as it is used nowadays.

So what does the model look like? As you might expect it is simple. The network consist only of three kinds of things: nodes, relationships and properties. Figure 1 provides an example of the different elements in a

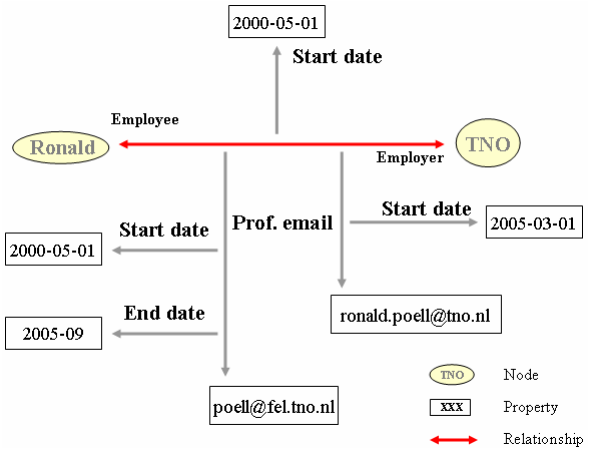

Figure 1: Example of a part of the network showing the different elements and the different occurrences of properties.

semantic network.

Nodes represents concepts as we, human beings, consider them. They are not scientifically well defined with clear borders. If I would describe their purpose, I tend to say that their main function is to allow communication. Everybody who looks at a node should be able to have a relative correct perception on what the node stands for. It is clear that no two persons will have the same awareness of the topic the node is representing because this awareness is directly influenced by their own (acquired) knowledge. Knowledge that will always differ in some degree from the knowledge of all other persons.

So how nodes represent the topics they stand for? That is the point where the two other kinds of things in the network come to play a role. In fact they are the only ones that really matter. Through the properties of nodes (like theirs names) and their relationships with other nodes they

[3]In fact they did and still do, but they were not used in that way and it did not occurred to me that I could misuse them for my purposes.

communicate what nodes are about. Nodes are basically nothing more then groups of properties and relationships and are probably the least important things in the network. This seems strange as the network is composed of nodes, but fundamentally they only exist through their properties and relationships.

Properties allow the addition of information to nodes, relationships and other properties. These elements are handled in more detail in section 3.5.

Relationships illustrate the meaningful links a node has with other nodes in the network.

Fundamentally we can distinguish two different kinds of relationships: real and virtual. The real relationships are the ones you can normally see when looking at (nodes in) the network. Virtual relationships are not directly registered in the network but can be derived according to some "rules". Figure 2 shows an example of virtual relationships. There is a virtual rela-

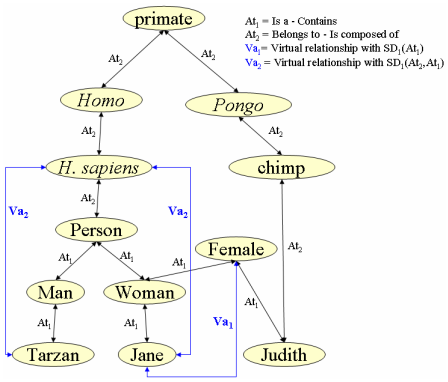

Figure 2: This part of the network illustrates the existence of virtual relationships.

tionship between "Jane" and "Female" if At_1 is transitive. When we consider the predicates At_1 and At_2 semantically equivalent and both are transitive there is also a virtual relationship between "Tarzan" and "*H. sapiens*" but also with "primate".

Virtual relationships are extremely important in communication between humans. When somebody asks me where I am born, the answer I should give depends on what I think is a geographic entity that belongs to his knowledge. It might be Europe, the Netherlands or Breda or even Zeisstraat 43. My node has a real relationship "is born in" with Zeisstraat 43 (the building where I took my first breath). But my node has virtual relationships "is born in" through the predicates like "is located in", "is situated on" and "is part of" up till the Universe. But if the answer provided would be "I am born in the Universe." it will be true but not very useful though. We have to make the junction between what information is available in the network and what is already known by the requester. When providing the answer

you have to hook into the world known by the receiver.

3.2 Fundamental impossibilities

As said above the concepts represented by nodes do not have strict well defined limits. In our dally usage of concepts only a few of them are strictly defined. You might think that the concept of color "red" is well defined. And in fact it can be. In the RGB (red, green, blue) color scheme the colors are defined as a triple of values ranging from 0 to 255 and pure red is (255,0,0). But in our daily life the color red is much more then just that pure definition. In figure 3 you can see that we cannot strictly define where

Figure 3: A color bar showing a continuous mix of color from pure red to pure green. It is impossible to define a strict limit on where a color cannot be called red anymore.

a color is not red anymore. The example is a simple one but its principles stand for almost all nodes. Even if for one person he or she might have a well-defined idea about what a node should represent, other people probably have ideas about that node that might be similar but probably not perfectly identical.

Nodes thus represent a more or less common agreement on a concept and, save some rare cases (like a "pure red" concept), are not strictly defined concepts. Most nodes form in theory a continuum with one or more other nodes and the frontiers between them are more or less arbitrary. When adding information to the network it might be difficult at some stages to decide whether a piece of information still belongs the concept other users created already or when adding it to the existing node you make a *conceptual extension* to it. This phenomena will be more frequent in immature networks than in full-blown or mature networks[4] In case of doubt we recommend the creation of a new node and specifying it as much as possible. Tools should be able to cope with the similarity and difference later in time when both concepts have become more mature.

3.3 Physical or abstract concepts

In our daily life when we talk about a person we might consider the physical person (you can phone him / her or send an email) and sometimes we consider that concept to be an abstraction of that physical person (e.g.

[4]An immature network has nodes that are so incomplete that the perception you have of them is not clear and leads to ambiguity. Needless to say that semantic networks can be immature in some parts and very mature in others.

the author of a book). The strict separation between these two concerns is applied for example in the ABC ontology[14].

As said before, we would like nodes in the network to represent concepts as close as possible to our human natural way of representing things. Following this axiom the representation of a physical object and its abstraction(s) should not be different nodes in the network. But how are we (humans) handling this? When we talk about a particular subject we switch between the physical and abstract concept depending on the nature of the property or relationship we are dealing with. Property types can have properties themselves that indicate whether the carriers of these properties should be regarded as being a representation of a physical object or an abstraction of it. If so, smart services dealing with nodes can make the distinction between the physical or abstract concept represented by the nodes and there is no systematic need to separate the two kinds in the network.

There are cases though where it is useful. In a library for example the physical instances of a book are important because those are the ones that are lend out. A library can have more than one specimen of a book. In this kind of cases it might be very useful to create separate nodes for the abstraction of a book and as many nodes as there are physical specimens available.

The most important to retain from this part is not so much on the choice when and when not to create separate nodes for the physical and abstract concepts, but on the fact that in the network there will be situations where this separation is applied whilst in other parts they are not. In general we expect unique physical things (at a conceptual level like a person) to be represented as a combination of the physical and abstract concepts. When more then one physical instance can exist (like books) we expect the representations of the physical objects to be separated from the abstract one(s).

3.4 Predicates

Depending on the context where the term *predicate* is used, the meaning of it is completely different. We use this term here to indicate the semantics or meaning of a relationship. A relationship is composed of three elements the <subject >is related through a <predicate >to an <object >. Predicates, subjects and objects are all nodes in the network.

As said in the introduction, one of the basic ideas is to conserve the model of the source and to apply appropriate grouping of similar elements from different models as necessary. When a source stipulates that <a lion><is a><carnivore>transforming this in <a lion><has a hyponym relationship with><carnivore>(or <a lion><is a hyponym of><carnivore>) the intrinsic meaning doesn't change very much but its semantic representation does. As for the names of the predicates there might be big differences in various languages depending on the context in which a predicate is used.

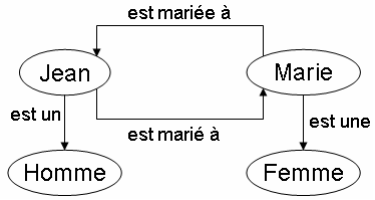

Figure 4: This figure shows the difference in names of the predicates "is married to" and "is a" in French.

In figure 4 this is illustrated for the French language. If the semantic network is used somewhere in the future to generate natural language, this will be greatly facilitated if the predicates used are those from our normal spoken or written language. Our recommendation is at this point: use as many predicates as are used in our normal language, provide the necessary properties for these predicates in order to be able to exploit them in a more generic way.

One of the useful kinds of information for predicates is between which kinds of nodes relationships with that predicate are normally established. In the example of figure 4 the predicates "est marié à" could have a property indicated that is used for men and the "est mariée à" for women only (whatever the other end of the relationship might be).

Most of the relationship types (predicates) are not symmetric (are identical when applied in one direction and the other, see the example of figure 4, but most of the predicates do come in pairs (asymmetric bi-directional relationships). Although many relationships could be bi-directional that does not imply that both relationships do exist (figure 5).

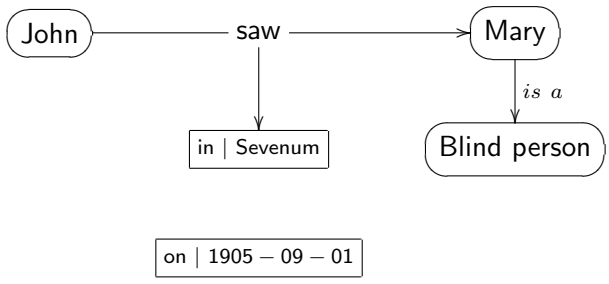

Figure 5: Representation showing why not all relationships are necessarily bi-directional. In this case it is surely not symmetrical because Mary cannot have seen John. If an inverse relation should exist, it would be: *Mary has been seen by John.* This combination is bidirectional and asymmetric (the two predicates are different).

In other words, two nodes have in general at least two distinct relation-

ships with each other in opposite directions. There is practical argument not to fuse two mono-directional predicates into one bi-directional with for example different roles. As we have seen there is a possibility to associate properties to relationships. In one bi-directional relationship there is no possibility to associate different properties to only one of these relationships. Figure 6 shows an example where the "display relevance" property value is

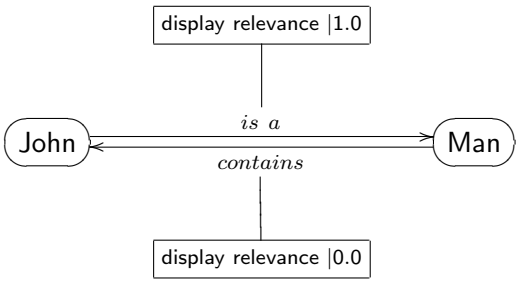

Figure 6: Representation justifying separate predicates and relationships through the usage of properties.

quite different for both relationships.

3.5 Property types

Properties of nodes, relationships and other properties consist of two elements: a property type and a value. The property type provides the semantics (meaning) of the property. For example the property (date,2005-07-04) has a property type "date" and a value "2005-07-04". When adding new information in the network one of the recurrent questions is the choice when to add something as a property and when to make it a relationship. If we look at the example of the color of a car we could add this information in the network as illustrated in figure 7.

Figure 7: Illustration of two ways of registering the color of a car in the network. On the left a property is depicted, on the right a bidirectional relationship with the "red" node.

An argument in favor of the solution with an asymmetrical bidirectional relationship with the "red" color node is quite straightforward. The "red"

node can be expressed in different languages. An attribute value cannot, it is just a string in this case.

Figure 8 illustrates a second example. If we would create a node for a particular day what does that node represent? It might be a period in time of 24 hours, you could say. But does it? A day expressed in terms of a year, a month and a day number represents 48 hours! You will need the time zone information if you would like it to represent a period of 24 hours. Then there would be as many day nodes as there are time zones with different time shifts, i.e. 34^5!

Figure 8: Two ways of representing a birthdate. On the left a asymmetric bidirectional relationship with a date node is shown. On the right the date is in the form of a property.

Fortunately in most cases the choice between the creation of a property or a relationship has a more or less *natural* or *common sense* solution. Nevertheless in some occasions you will have to make a choice.

What about property value typing? In most current and past ontologies property values are typed (i.e. integer, date etcetera.). Our experience shows that this is not a good thing to do unless you are in a entirely closed and controlled environment. There is an important distinction between the information itself and its *interpretation*. To stay within the example of a birthdate in genealogy (and history), figure 9 shows some examples of forms of dates you can encounter. Particulary the "About 1900" and the "begin

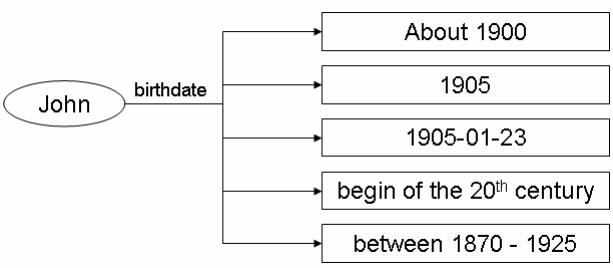

Figure 9: Illustration of the various forms in which sources provide birthdate information. It is clear that a strong typing (date) for the property value would lead to problems.

of the 20th century" cannot be associated with well defined dates in terms

[5]This not a typo. There are 34 time zones with different time shifts.

of our Gregorian calendar[6]. Some interpretation rules are necessary like "about means in genealogy from -5 to +5 the expressed unit". Obviously constraining the birthdate property as a date would quite quickly lead to problems. In our opinion the way to handle this correctly is composed of two steps. The source information is stored as is (perhaps a string) and second other properties are added giving the interpretation of the original value. Figure 10 shows an example of this. The interpreted properties are

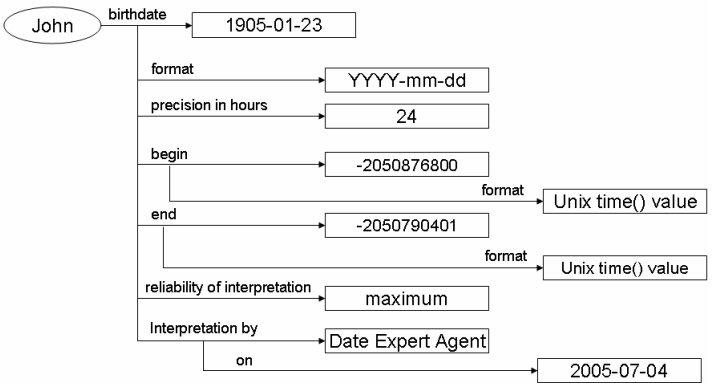

Figure 10: An example of the birthdate property showing the interpretation of the original information. The agent realizing the interpretation is also mentioned as well as the moment it was done. For the clarity of the picture the meta data about the interpretation (who and when) is attached to the original property, but normally they should be attached to each of the properties resulting from the interpretation. The original source of the information is not shown.

the ones that are used by services and applications. These derived properties can themselves be annotated in terms of reliability, interpreter, confidence interval and whatever might be useful and is available. For the "about" case of figure 10 we could just add some contextual information (genealogy for example) to specify the range in which the interpreted information is valid.

When some system tool that provides a calendar based date is the real source of information the source and its interpretation might be identical. For more detailed information and examples about dates see section 6.2.

Concluding this typing aspect of properties, we recommend not the use them at the source but to provide all the necessary information to make the correct interpretation on the fly. The interpretation can also be realized off-line (it will be quite stable over time) and the interpretation can by added in the form of other properties.

We call this kind of typing "loose typing" where a *property value type* property indicates what kind of interpretation *could succeed* on the actual

[6]Numerous literature is available on this subject like [20]

value provided. This interpretation is done on a "best effort" bases and might not (yet) succeed.

3.6 Cardinality

The cardinality of a piece of information indicates the number of occurrences of that piece of information that should or can exist.

In relational databases cardinality constraint are implemented in a straightforward manner as they are embedded in the physical data model. Within ontologies, they are often explicitly integrated.

The use of cardinalities is a good thing but there are some important pitfalls and, although well known, we still see some ontology creators jump into them. The problem comes from two different issues.

The first is that cardinality constraints sometimes have a geographical or ethnological context in which they are true but outside that context other constraints are valid. For example the marriage between one man and one woman is only true in some countries or states. Other countries accept homosexual marriages and polygamy.

The second issue is related to the fact that our information world is not even close to the perfect unambiguous real world we tend to model. The problem often occurs during information fusion between incoherent sources (e.g. two different birthdays for a same person), but is not limited to that situation. During a fusion process there is often no possibility to define which information is the correct one. So instead of having a logically unique birthdate for a person you have to be able to handle two or more.

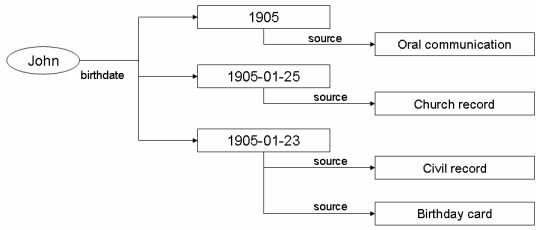

Figure 11: The node John with some of its birthdate attributes that have in turn attributes for the source where that information came from. Smart services or agents using for instance *source reliability knowledge* and *smart frequency counters* can establish the date with the highest probability. These probabilities could be associated in a persistent way with the birthdate properties but are not represented here.

Figure 11 illustrates a very common real situation where only a service that has some "smart rules" will be able to provide "the" birthdate for John.

Given these aspects, it occurs that cardinality constraints do not belong to the information itself but to the exploitation of the information. It is good

practise to integrate them in the information model in closed and controlled environments, but not in the open multi-model situation we are dealing with here. The rational of this remark is identical to the one explained for the typing of nodes and for classes and instances that will be handled in section 5.1. The implication of this can be illustrated by the birthdate example. A specific person has only one birthdate but different sources can have different records of it. The information architecture (semantic network) allows the registration of all of them (of which only one is supposed to be true). Applications that have a view on this kind of data with a cardinality of 0 or 1 (no or one birthdate) have to decide which of them (the most likely) to present to the user. Smart services ("get the best" function) can assist the applications in this task.

4 Operations on the network

In the previous sections we saw how information can be represented in a semantic network. The logical question popping up is: "What can you do with a semantic network?". The answer will be quite difficult to provide as would be the one on "What cannot be done with it?" We will start our perception on what is possible at an atomic level.

As we have seen in section 3 there are only a few kinds of things in the network: (nodes), properties and relationships. Nothing more. Given this, we can define an *elementary operation on the network*. Such a basic operation consist of only four elements: a set of nodes to start with, a set of predicates, a set of property types with optionally constraints on the values of the properties and finally instructions on what to do.

The set of start nodes indicates where to begin the operation. How to get an appropriate start set is a complete different aspect and depends strongly on the specific operations carried out. It will not be handled in this paper.

The set of predicates enables the isolation of a part of the network through a subset of all the possible paths (relationships). It allows a sort of reduced view on the total network.

In most cases the set of property types (and the property values) enables the service to identify nodes that have or have not some required aspects.

This instruction set makes the basic operation specific for a particular task.

Each specialized service that executes an elementary operation does nothing more then walking through the network according the paths he is allowed to follow, compare the encountered properties (of nodes or relationships) and do something specific to its particular task.

Great complexity can be obtained by chaining several *elementary operations* that are themselves extreme simple and comprehensible. There is a great similarity with the aspects of complexity and randomness described by

Stefen Wolfram[22] and this is not a surprise. Wolfram has come to his conclusions starting from mathematics (in particular the cellular automatons) and through years of experimenting and investigation extends his findings to the whole universe. I started looking for something that represents a workable form of elementary information and elementary operations on it to achieve complexity in the final operations. If Wolfram's *New Kind of Science* is correct, it is not a surprise that my findings also do deliver the desired results.

A small example might illustrate an elementary operation. The mission is to *find an appropriate picture for a node*. A service providing this capability needs to walk through the network following the paths that express some kind of "IsA" or a "has as picture" relationship until it encounters a node that has a characteristic of a picture. Picture nodes may be characterized by a specific set of properties (url, a name ending on a specific extension, particular mime types etcetera). Its specific task consists of building a list of nodes that represents pictures and return that list (figure 12).

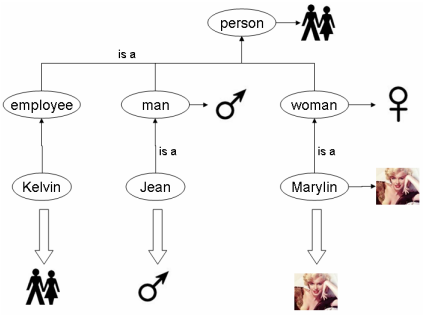

Figure 12: Illustration of the operation *find an appropriate picture for a node*. The relationships of the kind "has as picture" are not labelled in this figure. The selected pictures for the concepts Kelvin, John and Marylin are displayed at the bottom.

This example is in fact nothing more than a form of multiple inheritance of the *has picture* virtual property. The *has picture* property is not a real property because it exist through a relationship not through a property of a node.

A slightly more complex service that must *provide the html tag of a picture for a node* will use the service described above and provide it with the starting node reference and a second one that doesn't need to walk through the network (and needs no predicates) but takes a set of *URL* like property types, visits the nodes provided by the first service and takes the appropriate property values. It can optionally check the availability of the URLs and transform an available one into the desired tag (with or without scaling).

The second basic service might have an auxiliary task (side effect) that

it registers for unreachable URL properties that, on a particular moment in time, it could not be accessed by him. This might be useful for other services that check the validity of URLs over time and perhaps invalidate, according their own (statistic) rules, the "permanently" unreachable URLs.

The first service looks like a specific one (find an appropriate picture for a node) but can be made very generic: *walk through the network along the authorized paths, look at the encountered properties and build up lists of nodes that have or have not appropriate properties or relationships.* What makes the service suitable for a specific task are the provided predicates and property types to use[7].

The most generic description of an elementary operation in the network can be formulated as follows: *take a set of nodes to start with, walk along the authorized paths, look at the encountered properties and do something with it.* In fact it occurs that nothing more can be done at a atomic level with the semantic network. Every complex operation can be constructed as a sequence of elementary operations.

A classical "find" operation can be expressed in two different ways according to this definition: *(i)* "take all nodes as a start set, do not walk (no predicates) and if there is property of the required type and value add this node to your result list" or *(ii)* "take one node from all isolated parts in the network as the start set, walk through the entire network (use all predicates) and if there is property of the required type and value add this node to your result list". A good implementation of a general find operation would probably be quite similar to that from a relational database. But consider the following: "I would like to see all nodes *within a network distance*[8] *of 4* of the node that represents the Unified Medical Language System and that are diseases". Realizing this in SQL might be quite a puzzle[9]. Expressed in terms of elementary operations on a semantic network, it might look like:

1. take all nodes as a start set, do not walk (no predicates), look at the name properties and if one is similar to "Unified Medical Language System" add that node to your result list.

2. take the node(s) from 1) as a start set, walk through the entire network (use all predicates) until the distance from the start node is 4 and add all these nodes to the result set.

3. take the result set from 2), walk through the network along the paths

[7]The implementation at TNO uses these kind of services, where they are called *spiders*. They serve various purposes. The one described in the example is one of them, others for example are used to build up sets of nodes from cluster hierarchies.

[8]The network distance between two nodes is the minimum number of steps you have to take to go from one node to the other. Nodes that have a direct relationship are at a network distance of 1.

[9]We suppose that the database is not a specialized one for the UMLS.

of a "IsA" like kind. If one of the nodes reached has a name property similar/equal to "disease" add the start node to the result set.

The result set from 3) will be the answer to the question[10]. As you have noticed we introduced a new "parameter" in 2): the maximum network distance. In fact when walking through the network it is often useful, if not necessary, to limit the wandering distance. If not so, with a (almost) complete list of predicates the entire network might be visited. When a reduced set of predicates is used this parameter is often not necessary.

Classical inferencing rules can be expressed as an elementary operation or more often as series of elementary operations.

Recent reflections about inferencing new information from the network showed that even the inferencing rules should be considered to have a *context* in which they can be applied while outside that context they should not. This remark is an extension to what has been said before that each information is valid in a specific context. Their usage (within the context in which they can be applied) makes them useful. Inference rules are often task related and only some of them are universally applicable (for all tasks).

Sometimes inferencing rules are language dependant. In Dutch there is only one single word ("neef") for the English cousin and nephew. A rule based on the Dutch single word will have its equivalent as two rules in English (and in French).

5 Special kinds of nodes

We saw in section 3 the overall architecture of the semantic network and in section 4 how to manipulate information in the network. In next three sections we go a bit further to explain our vision on *what* information to put in the network and *how* to do it. We will describe in this section some of the kinds of nodes that require some attention. We will have a closer look at model nodes, events and the last part of this section is dedicated to technical nodes. Section 6 explains some specific kinds of information and finally section 7 draws your attention to some of the aspects of information itself.

5.1 Models

It is strongly recommended to integrate the model elements in the network, just like any other information. The greatest advantage of doing so consist

[10]It is clear that the formulation of 3. is directly related to how the various diseases are identifiable as diseases. We suppose in the actual formulation that there are direct or indirect links to the node disease. For example they can form a hierarchical classification of diseases. The node disease might be at a greater network distance than the maximum allowed as this distance applies only to the diseases (2) not the disease node itself (3).

in the fact that they can be manipulated, analyzed, etcetera just as any information in the network and with exactly the same tools. Models typically consist of a set of predicates — illustrating the nature of the possible relationships between nodes — and a set of property types for nodes and relationships. Figure 13 shows some *model* nodes and model relationships[11] integrated with *normal* information.

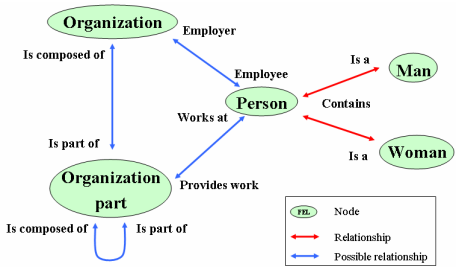

Figure 13: A *model* part of the network integrated with a *normal* part. Only the "possible relationships" belong purely to the model. There is no real relationship between the organization node and the person node. All the other information in the picture can be considered as a model element in some situations or as normal facts in others.

It is important to bear in mind that the types of entities that occur in a model are eventually not the only ones that will be applicable to the nodes of each type. A specific node can be considered for one purpose to belong to a class/entity A while for a second purpose it has to be considered of class/entity B. Loose typing can be realized through the use of properties or relationships with nodes that represent "classes" but are used only within a specific context. A good example comes from the classification of living creatures. Traditionally we use the Linne classification but in past years the *phylogenetic* tree is piece by piece constructed[10]. The leaves in both trees are identical (the creatures) but the intermediate groupings are often completely different. The sets of properties are also quite different.

In ontologies *instances* and *classes* are fundamental different elements. This is a perfect situation in closed domains with clear objectives for the usage of the ontologies. In an open domain this gives often more problems then advantages. Remember that you can only partially foresee for what purpose the information in the network will be used. At some levels it is even hard to define whether a concept should represent a class or an individual. The relationship *a Bernese Mountain Dog is a (kind of) Dog* can be interpreted as an instance (Bernese Mountain Dog) of a class (Dog). On the other hand in if we have the relationships *Archie is a Bernese Mountain*

[11]Model relationships are "possible" relationships. They indicate that, in this example, a node that is a person *can have* a relationship employee/employer with a node that is an organization.

Dog and *a Bernese Mountain Dog is a (kind of) Dog*, what does the *Bernese Mountain Dog* represents? It will depend on what you want to do with it whether you consider it to be class (sub class of Dog) or an instance of a Dog. Those who are used to create or work with ontologies will know, perhaps unconsciously, that something is a class for implementation reasons. Inheritance of properties often plays a role in the final choice.

In our point of view the *typing* of nodes is something that does not belong to the fundamental *structure of the information* in the network, but exclusively to domain of the *usage* of the network.

In a multi-model network there will be model elements (predicates, properties or classes of nodes) that are slightly different in meaning but close enough to be considered identical for a particular purpose. There are basically two ways of grouping "similar" model elements so applications and services can use them. You can add one or more specific properties to an element so you can recognize them or you can make a (hierarchical) network of model elements enabling clustering of them at a chosen granularity. It will be the way you are going to use the information in the network that will define which (model) nodes propagate their properties through an inheritance chain and which should not. The need for this non-propagation is not new. In the Unified Medical Language System[6] for example, explicit inheritance stops are build into the information structure.

When adding a new source of information to the network you will often encounter a new model. Parts of it might identical or almost identical to the models you have already in the network, parts will be entirely new. For new kinds of information things a relatively simple: you will create a new model element that will be semantically equal to the model element of you source. But often it will be not very easy to establish perfect equality between a model element of your new source and one or more existing model elements. You will face the choice between two different actions. Either you apply *early model binding* or you use *late model binding*. With early model binding you *establish equality* amongst model elements during the creation of information in the network and with late model binding you *assert the binding in a dynamic way* during the exploitation of the network. As should be clear by now, our recommendation is to apply in almost every case a late model binding. Between model elements from different models you can of course create relationships of the kind *similar to* which might greatly facilitate the job of services that you order to consider these elements to be equal.

5.2 Events

A huge set of definitions of events exist[12]. We follow a very general one: something that happens at a given place during a period of time possibly involving some actors with various roles. It is of course not necessary that all the information is available or precise. Events exist as nodes in the network with all the information about the event represented in properties (e.g. the period of time during which the event happened) and relationships with other nodes (e.g. the actors). In case of the actors the used predicates can be generic — is actor in / actor — with properties of the relationship specificating the role, or specific — is husband in / husband — and no extra role properties might be necessary.

When should some piece of information become an event and when not? This question is a legitime one for all the creators of information. There is of course no strict rule for it as we do not impose strict modeling rules at all. Let us have a look at an example for genealogy. Suppose the following information available: "*John and Mary married on September 1, 1905 in Sevenum*". This could be represented as illustrated in figure 14. The same could be represented in an "event" form as shown in figure 15.

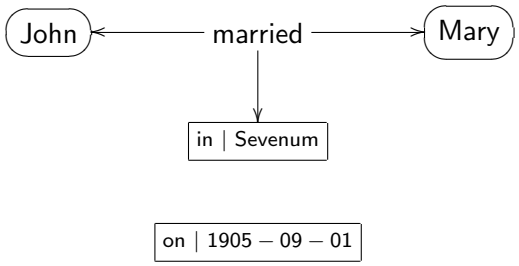

Figure 14: Representation of "*John and Mary married on September 1, 1905 in Sevenum*" as relationships with properties.

Other representations are possible. The choice of the form depends often on the purpose or the nature of the source information. If in the example we have information about e.g. the witnesses the *event* form becomes a more natural one with several kinds of actors and other associated information. The relationship between John and Mary in figure 14 doesn't represent the *event* but only the action "married". If we would have called the predicate "is married to", the relationship would represent the nature of their relationship or a state.

Events often represent state transitions of information (like the state *is married to* of both John and Mary in the previous example. It will be more

[12]See e.g. [1] for several definitions in a broad contexts and [3] for a specific one in computer science

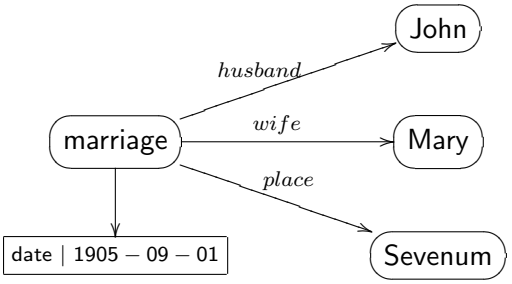

Figure 15: Representation of "*John and Mary married on September 1, 1905 in Sevenum*" as an event.

difficult to define the essential non trivial state transition of a *colloquium event* although there are many smaller, less important, state transitions one can imagine.

Two perhaps trivial remarks are appropriate. All events, whatever their nature might be, are always composed of several smaller events. And inversely: all events are part of a bigger (broader) event. The question remains if these statements are invalid at infinite big and infinite small borders. The biggest event we might conceive at a first glance might sound like "the existence of the Universe". But perhaps there is one bigger "the existence of all possible universes". Or perhaps there exist something bigger then "all the universes". The same reasoning can be applied to the infinite small limit. Even if the question might be important in philosophical point of view, its practical consequences for semantic network implementations are reduced.

Important to retain here is that in a multi-author and multi-model network all the forms of event related information (action, state or event) will co-exist. Exploitation services (applications) should take this into account.

5.2.1 Scenarios

Scenarios are excellent examples of event based histories. Whether scenarios are build to illustrate a possible event or the tracing of something that happened really and is used in a lessons learned situation doesn't make any difference. One of the aspects of many scenarios is that there are various scales of granularity all occurring within the same scenario. There will be global scenes, describing on a high level what happens in an area up to very detailed scenes describing who is doing what in a chronological manner.

A scenario might describe a complete virtual world, a virtual time line anchored in a real world, a hypothetic series of future events or series of past events occurring in the "real" world. From a semantic network information architecture point of view these different kinds of scenarios can all be treated in the same way. A precaution must be taken, not on the information

architecture level, but on the service or application level, in relation with time. Information manipulators (services or applications) should not take for granted that time information is represented in the way it traditionally is (as in our calendars). These services should be able to cope with e.g. relative events (time is represented in relative terms of other events) or absolute numeric data in e.g. pys (the π part of a Saturn day) or in revs (revolutions of a spacecraft like in the Gaïa trilogy by J. Varley). Section 6.2 describes in more detail time related concepts.

It is quite easy (although a lot of work) to integrate the contents of a book like "Lord of the Rings" (J.R.R. Tolkien) or the stories of Harry Potter (J.K. Rowling) or even the more general "Transmedial Worlds" as in [13] into a semantic network. When this is realized, we can observe some interesting facts if we look at the semantic network from a high altitude (helicopter or satellite view). A cluster of the nodes of the cited examples will be relatively isolated in the network and only a few kinds of relationships will exist with the rest of the network. Lets have a look what happens. There is of course the original story (a book, film or whatever so) that exists in our real world. Next we will see that at various level of detail, relationships will appear with concepts from our real world. Some of the swords in the Lord of the Rings and the cloths Harry Potter is wearing are similar to the ones from our daily life. But also some of the behaviors exposed by the actors in the story will be recognizable. Some of the used time lines are related to the ones we use currently (in science fiction for example), other time lines only exist through the story and are not related to our real world as in Lord of the Rings.

In a scenario for a crisis management situation the only thing that is isolated in the network (beside the events themselves) is the time line. All the other topics are the ones from the (future) real world. This time line exist only in the scenario, probably as series of relative events. In some cases scenarios illustrate situations somewhere in the future and have an anchoring on our real time line.

A semantic network or similar information architectures seem perfectly suited for the description of scenarios. They refer more or less to parts of the "known" world and parts will only have relations and properties of the scenario alone. Multiple time lines can coexist, conditionally diverting and perhaps joining again later on in the script. Levels of detail can vary as desired and be changed at any moment in time.

5.3 Technical information

A semantic network will be multi-model and nothing hinderers the existence of information in the network that is technical information for the "operators" on the network. The reason why we speak of "technical" information is that these categories of information are not for human users in general. The technical information can be divided into at least two different categories.

The first kind is information that serves identification. This can be the node's own identification (UUID) or a property that refers to a record in a database that contains information about the same concept. These identifying properties allow cross links between the semantic network and other sources of information. When realizing a mass import of information from another source, our recommendation is to import also the identifiers from the foreign source.

The second kind of information are configuration parameters and other application or services related parameters. One group of information is formed by the textual elements of user interfaces. Buttons, menus and menu items can exist as nodes in the network. One of the quick wins shows up in the capability of providing multi-lingual applications with very little effort. Parts of these user interface component nodes can be shared by many applications and adding new name properties in other languages makes these languages immediately available for all the applications. The advantages of the use of application or service parameters is a little more subtle. In figure

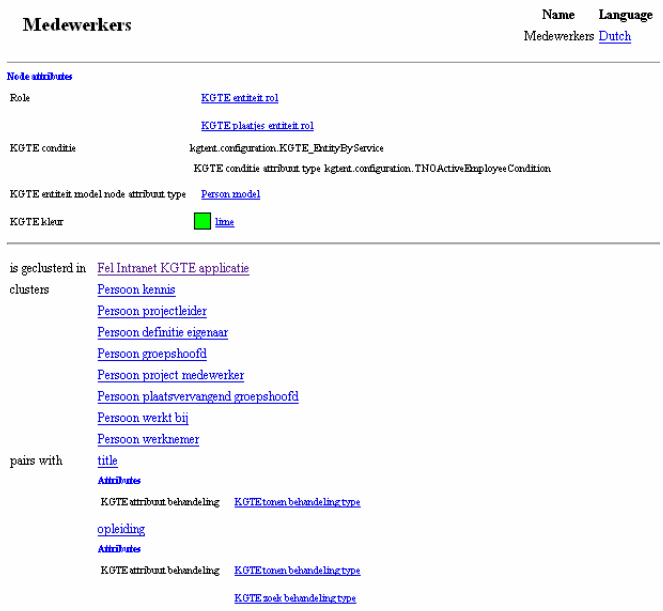

Figure 16: Screen shot of a configuration node of the employee entity for the knowledganizer.

16 a screenshot shows a part of the configuration of a visual navigator on the semantic network (knowledganizer or KGTE) at TNO. The KGTE is commonly used for the knowledge map of the employees of TNO but can easily be configured to display completely different kinds of information. Furthermore you might be able to "copy and paste" a configuration part of the network, apply some changes to it, and dispose of a personalized

application configuration.

We saw in section 4 that elementary services are configurable through sets of predicates and sets of property types. These sets are probably good candidates to be retrieved from the network itself. And there is even more to it than one might expect at a first glance.

Imagine the following scenario. Somewhere in the future another information set is added to the network with again its own model. The model of this set contains a "is a form of" predicate. The service in charge of finding pictures for nodes will at first not be aware of this predicate and will not find any pictures for the new nodes. In order to use this new predicate it should occur in its list of predicates. If these predicates are clustered in some way in the network and the set is composed from that cluster, the only thing to be done is to create in relationship of the appropriate kind between the new predicate and the cluster. This can be a manual action, but we think we might go to a form of a self learning network where this link is created automatically. We even think that this self learning capability will be the only viable solution for the future. We expect the amount of information and its variety of models to be that big that no manual surveillance of it will be possible anymore.

As inferencing rules consist of series of elementary operations and these can be configured via the network we consider that these rules are not fundamentally different from all other kinds of information in the network. They are equally subject to automatic enhancement.

Recent research[12] indicates that automatic discovery of some level of "understanding" of the semantics of nodes might be relatively easy to realize. If the subset of predicates of the *IsA* kind (used by the service) have properties in common like "transitiveness" and "occurs as a pair with some other predicate itself also transitive" and the new predicate has the same kind of characteristics (they have a certain similarity) then the new one might be a good candidate to be added to the configuration cluster. These two characteristics of predicates can easily be deduced from the information in the network if a certain critical mass of information using these predicates is available. Although this research is far from finished, the first results are very promising.

6 Special kinds of information

In the section 5 we saw some specific types of nodes. This chapter is dedicated to several kinds of information that can be associated with nodes. Some of them require special attention and they are detailed in this section. We start with the aspects of *names* of nodes. Next we have a look at the complex subject of *dates and time* and finally *geographical* aspects are highlighted.

6.1 Names

The multi-lingual aspect of the nodes has already been mentioned. Each node can have one or more name properties each of which is related in turn to one or more languages. Proper names can have a specific property indicating that they are invariant across different languages.

Using the textual representations for nodes (their names) brings us to field of linguistics in a semantic network. We will cover briefly in this section aspects related to the following topics: homographs, synonymy, polysemy, antonymy, hyponymy — hypernymy, meronymy — holonymy and misspelling.

6.1.1 Homograph

Homographs are words spelled alike but semantically dissimilar. Real homographs only occur within a same language.

As the names in the network are properties of nodes and nodes represent different concepts, no specific homograph relationships are necessary. But as nodes with the same name exist (homographs), it is important when representing a node not to use only the name property but something more allowing the user to identify or to recognize which of the homographs he is looking at. In general this is only a point of attention when representing *lists* of nodes where the easy way would be to show the name only.

There are cases where nodes can have identical names but are not (real) homographs but related to the relative coarse definition in one language and some other more fine distinctions in other languages. The example of the English word "leg" illustrates this case.

In English there exist a word "leg" that has according to [2] in WordNet (1.7) 9 meanings. These might be represented as nine nodes in a sematic network. Two of the meanings are defined as follows *(i)* "a human limb; commonly used to refer to a whole limb but technically only the part between the knee and ankle" and *(ii)* "a structure in animals that is similar to a human leg and used for locomotion". On the same web page but in the section from Webster's Revised Unabridged Dictionary from 1913 (providing also 9 meanings) these two are grouped into *(iii)* "A limb or member of an animal used for supporting the body, and in running, climbing, and swimming; esp., that part of the limb between the knee and foot." The WordNet definitions are more fine-grained then the one from Webster. In other languages like French or Dutch there is, compared to the animal definition from WordNet, another word for the legs of human beings and horses *(iv)* ("jambe" and "been") then the one for all the other animals *(v)* ("patte" and "poot"). For these languages we would have either two definitions with a distribution of "humans AND horses" and "animals other then horses" or three definitions "humans", "horses" and "other animals then horses".

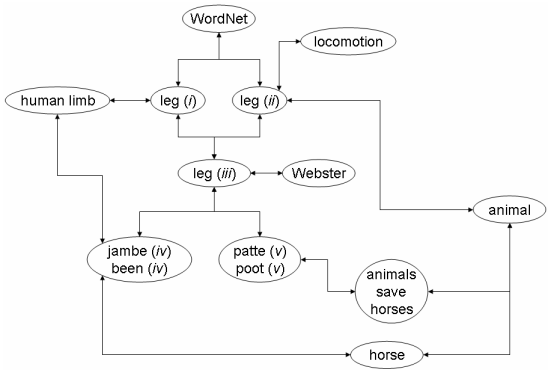

Figure 17: Part of a network illustrating one of the forms of the "leg" problem.

There are several ways to represent this apparent problem in a semantic network. Figure 17 illustrates the creation of a node representing the concept *(iii)* with a name property in English only (and the other languages where the same concept exists) and two other nodes representing *(iv)* and *(v)* with French and Dutch names. Specific relationships between *(iii)* and *(iv)* and *(v)* can clarify the relation between these concepts. In the same way the *(i)* nodes *(ii)* and *(iii)* are "grouped" in *(iii)*

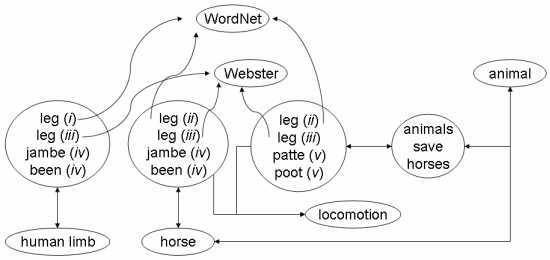

Figure 18: Another illustration the of the "leg" problem. The two English "leg" names in the nodes serve to illustrate where the term comes from. Each of these nodes will have only one name property in English.

Another solution is illustrated in figure 18 and consists of the creation of three nodes where two will have the *(iv)* property and one the *(v)* and the English name for all of them.

Whatever solution you might choose in a particular case, keep in mind that as long you don't have a global knowledge about the usage of words in *all the languages of the world* you will never do it perfect. Be prepared to extensions of the network in future where both types of solutions coexist and perhaps even more than these two.

Another example of homograph like situations comes from a practical case and is illustrated by figure 19. An ethnologist interviewed a person

about the usage she made of plants. This person uses the same word for two different species and it is not clear during the interview when she refers to one and when to the other species. Apparently, in her mind, there is only one concept referring to *both* plants. The creation of *one*, rather personal, concept/node appears to be the most natural solution (to me, but not necessarily for you). This node can have relationships with both species

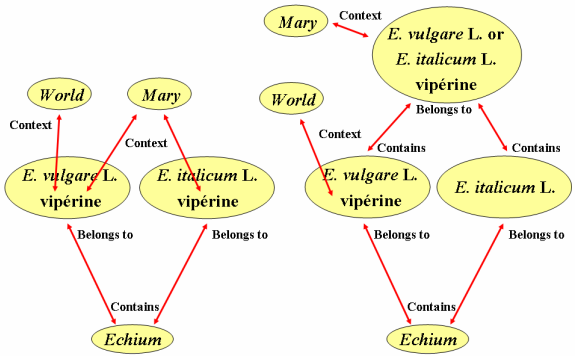

Figure 19: A practical case from ethnology. On the left the solution where names are valid within specific contexts. On the right the creation of a specific node that is only valid in the context of Mary.

indicating that is a kind of composite node representing either one of the related nodes. Another solution might consist of the addition of the name she uses to both species and adding context information to the name indicating the (mono individual) community in which the name is *valid*. Both solutions are possible.

No global rules should impose one unique uniform solution for these kinds of aspects for a multi-author multi-model information architecture. But services handling this information should be made "clever" enough to handle all kinds of solutions.

6.1.2 Synonymy

Synonyms are different words or expressions with the same semantics.

The way real synonyms exist in the semantic network is probably the most straightforward of all the items handled in this section. As synonyms represent the same idea or concept they are all associated as names of the *same* node in the network.

But two expressions are synonymous if the substitution of one for the other never changes the truth-value of a sentence in which the substitution is made[13]. This is extremely rare or perhaps even doesn't exist at all. What we can say is that within a particular *context* expressions can be considered

[13]This statement is usually attributed to Leibniz.

synonyms (if they don't alter the truth-value during substitution). Different expressions are more or less semantically similar. This similarity will be in most cases a continuum and not a discrete interval (within this continuum the context changes). This aspect of meaning within in particular context refers to the essence of semantic networks where different nodes represent different concepts.

One of the major questions when defining a new node is whether the meaning of the node is sufficiently different from all the other nodes in order to justify a new node. Is there only a context shift but in essence we handle the same subject as an existing node? We saw in section 3.2 a detailed explanation on this important topic.

The major issue identified in relation with synonyms is not related to the nature of the semantic network itself, but to its usage. Choosing the most appropriate name when displaying a particular concept often depends on the (usage) context.

6.1.3 Polysemy

"Polysemy is the ambiguity of an individual word (or phrase) that can be used (in different contexts) to express two or more different meanings"[8] or "Polysemy refers to a word that has two or more similar meanings"[9]. These two slightly different definitions don't have a fundamental different impact on how this semantic aspect might be handled in a semantic network. The ambiguity of the meaning of words exists in phrases, not in the names of nodes in the network. Names are just one of the kinds of properties of nodes, very useful in communicating about them, but they do not need to be unique. Nodes that have polysemic names should have some properties and relationships that allow everybody to distinguish them.

On the other hand polysemy is one of the difficulties when extracting information from text, i.e. transforming the content in terms of nodes, properties and relationships. All the nodes with polysemetic names are potential candidates for the mapping of the word in the text. But in mature networks the position of a node in the network, i.e. the (virtual) relations it has with other nodes, define some form of "context". Most nodes with identical names belong to different "contexts". For texts that are not to short (provide enough contextual information) disamguiation is often possible.

6.1.4 Antonymy

"Antonymy is the semantic relation that holds between two words that can (in a given context) express opposite meanings"[7]. According to this definition antonymy is not a semantic relation between nodes but between names of nodes. It is a bi-directional and symmetric relationship.

But can we extrapolate from a lexical antonymy relationship on word

forms an antonymy relationship between nodes? We think you can in most cases. Sometimes though, as stipulated in the definition given above, a context restriction might apply. The question remains whether the *antonymy* relationship between nodes should become a permanent *fact* in the network or dynamically derived from the real antonymy relationship between the names of these nodes. Whatever the answer might be, I have no fixed idea about this, but the presence of antonomy information is important. Natural Language Processing (NLP) services might use this kind of information in order to discover possible second-degree meanings where the opposite of what is really meant is formulated[14].

6.1.5 Hyponymy and hypernymy

Equivalent terms are: subordination - superordination, subset - superset, is a (kind of). Maple is a hyponym of tree. And tree is a hyponym of plant. Plant is a hypernym of tree and tree is a hypernym of maple.

The hyponym and hypernym relationships are transitive and asymmetrical. They are opposite relationships.

Within the semantic network there might be more than one predicate of this kind. For example in the living creature classification we can distinguish two major currents. The traditional one uses the Linne classification, the second on is based upon phylogenetics (clades). Both predicates can be used as a hyponym-hypernym couple but should remain separated because the meanings are obviously not the same.

For this reason the hyponym-hypernym couple is to be considered as belonging to a meta-model describing exploitation rules. Services can hook into this meta-model and get a reference to all the predicates that are to be considered as possible transitive and asymmetrical indicators of relationships between nodes. The terms hyponym and hypernym are used for the nodes in the meta-model and considered to be characteristics of predicates that can be activated (used) by services.

Hyponymy is often used to construct inheritance trees. This characteristic can also be used in the semantic network. The hyponym-hypernym property of a predicate is dynamically activated during the exploitation and can be used by inference engines to apply conditional inheritance.

Combinations of predicates in a virtual hyponym-hypernym cluster can exist and even automatically be discovered for use in specific contexts. Conditions for the automatic discovery of these kinds of clusters can be for example a set of shared characteristics of the source and targets of the relationships using these predicates.

[14]The discovering of this second-degree meaning can not only be based on the existence and use of the opposite but should be realized in combination with contradictory information coming from the analyzing process compared to what is already in the network.

Hyponym-hypernym predicates form basically a one level hierarchic structure with various clusters (and thus becomes a network in itself).

From the above the question might come up what the granularity of the predicates should be. The section 3.4 explained already our vision on the ins and outs of this topic.

6.1.6 Meronymy and holonymy

This couple of predicates is known as HasA or part-whole relationships. A is a meronym of B if you say that A is a part of B or B has an A (as part). Meronymic and holonymic relationships are transitive and symmetrical.

A holonym relationship (has) is often associated with a qualifier indicating the quantity (A car has 4 wheels, a tree has many branches). In most cases the meronym relationship has no qualifier associated (at least in human usage).

A concept can have many holonyms (can be part of many things) and concepts have many parts (meronyms). This is true for abstract concepts. But a node representing a "physical" thing can only be part of one other node (the motor with serial number xxx is part of the car with badge yyy) at a certain moment in time. The term "physical" doesn't represent necessarily a thing that provokes "bang - ouch" when you drop it on your feet but can also be a non physical concept like a particular subproject that is part of a specific project.

The same remarks about the existence of a network of predicates as for the hyponym-hypernym relationships apply for the meronym-holonym relationships.

6.1.7 Misspelling

Misspellings are words where the orthography is slightly different from the real orthography. Misspelled expressions should be associated as names with a node but with a property indicating it is a misspelled name. The misspelled names should remain searchable so humans beings as well as Natural Language Processing (NLP) tools can find the associated nodes. It might be difficult for automated processes to discover a new misspelling with enough certainty to consider the misspelled expression for what it really stands for. It will be quite easy in a long text where the context might refer to a sub-network in which the node with the real orthography is present but in short texts like questions this will be far more difficult as there will probably be not enough context in it.

We consider misspelling as extremely useful and recommend to keep or add that information in the network. Referencing to what has been said in section 5.3 about technical nodes, misspelled names can be considered also

to belong to the technical kind as it should, in most cases, not be showed to human users.

6.2 Dates and time

A practical way to illustrate some of the problems we have to face (and not to deny) can be realized through dates. We will develop this aspect a bit more in this part.

In our actual information world we are more and more confronted with the exploitation of time based information. Within a specific domain, for example the rendezvous calendars, no specific problems occur (save perhaps the ones related to the different time zones). But when we would like computers to use time based information in a correct way across domain borders ("Give me all the objects of this museum between the dynasty of Shang and the death of Louis XIV of France") things are somewhat more complex.

We have to face two different kinds of phenomena. In the first place human beings don't speak a language easily understandable by computer software. Ongoing work in the field of annotations for the Semantic Web provides new perspectives on parts of the problem. Second we (human beings) switch contexts in a natural way whereas software must be "taught" to make these switches. These context switches contain often also a switch in granularity (coarser or finer) more appropriate for the new context. The transformation of human expressions for time information into a machine *understandable* form imply almost always an explicit context for high accuracy. But this explicit context is rarely available and an implicit context should be evaluated if possible. When an implicit context cannot not be defined with enough accuracy a more coarse *global* context should be applied.

In the following we will illustrate some of the aspects of time instances, time lines and periods. A few topics are on stage for the interaction between computers and humans. Finally we have a look the aspect of time-based information in part 6.2.5.

6.2.1 Time instances

Many ontologies dealing with time and dates define a *point in time* also called a *time instance*. In our opinion this is often not done in an entirely correct way. Mathematically you can define an point in time with a duration that is infinitesimal small. But even in mathematics when trying to formulate a time instance we see that we cannot express it independently of the used time unit.

(1) A time instance can be expressed in combination with an implicit or explicit time unit or associated with its precision in a given unit.

There is no *practical* perfect point in time. There are only periods of time. Whatever unit you might use (day, second, millisecond) there is always

a smaller unit (or fractions) subdividing the unit you have chosen. This is not really important when storing times but it becomes really important during exploitation of time-based information (including also the famous border problem).

Each time instance in unit U_1 can be expressed as a period in the more detailed unit U_2. A time instance θ_{U1} can be expressed as a period $\Phi(> \vee \geqslant$ begin $< \vee \leqslant$ end).

$$(A)\theta_{U1} = \Phi(\geqslant \theta_{U1} < \theta_{U1} + U_1)$$

$$(B)\theta_{U1} = \Phi(\geqslant \theta_{U2} * U_1/U_2 \leqslant \theta_{U2} * U_1/U_2 + U_1/U_2 * (U_2 - 1))$$

(2) Each time instance can be expressed as a period in some smaller unit.

In software it is common practice to set the smaller units of time to one (2004-07 is considered to be the first day of July i.e. 2004-07-01) or zero (2004-07-01 is considered to be at a time of 00:00:00) depending on the logical zero-based or one-based property of the unit. But when a human generated expression of July 2004 is given he/she probably refers to the whole month and not to 2004-07-01T00:00:00 GMT.

(3) In software time instances often express the beginning of the associated period.

A date or time should be associated with a precision or even better should be interpreted as a period with its inferior and upper limit based on the precision. July 2004 precision: month.

6.2.2 Periods of time

As explained previously, time instances can always be considered as continuous periods. The other type of periods are the discontinuous ones. They are composed of two or more continuous intervals that do not overlap.

Each continuous period can be expressed by either one time instance or two time instances.

If one time instance is used, the rules for the inferior and superior limit of that period are the ones explained in the previous section. For example the period 1956 is equal to ($\geqslant$ 1956-01-01 00:00:00Z, $\leqslant$ 1956-12-31 23:59:59Z) with the second as unit. A more precise way to formulate it would be ($>$ the end of 1955, $<$ the beginning of 1957) where the "definitions" of 1955 and 1957 will define the exact borders[15]. In the last formulation we do not miss the last second of the year. Similar a day on Earth lasts for about 48 hours!

[15]To be precise we assume that these values are geographically grounded in the time zone with no time shift. If we would consider that year for the whole Earth it would have one more day!

When two time instances are used for example in (1956,2005-07-07) the inferior limit of the period will expand to the inferior limit of the first time instance and the upper period limit to the upper limit of the second time instance giving ($\geqslant$ 1956-01-01 00:00:00Z, $\leqslant$ 2005-07-07 23:59:59Z) or more precisely ($>$ the end of 1955, $<$ the beginning of 2005-07-08).

Notice that if you want to make an exact interpretation of time references we commonly use, you will often need at least one "relative" reference ("after the end of" or "before the beginning of").

6.2.3 Time lines

Most ontologies containing time information deal with a particular range in time as they are (often) domain specific. Problems arise when we want to use various time scales in one ontology or in a semantic network for example. Some examples of time scales are:

- subatomic particles lifetime (nanoseconds)
- calendars (often centuries, years, months and days or their equivalents)
- historical (expressed as periods referencing historical events or states: Middle Ages[16])
- geological (expressed as relative periods - stages - and regularly revised for the absolute ages)
- astronomical (expressed in billions of years[17])
- time scales using unusual units of time[18] that might or might not be related to the second
- pure relative scales see 5.2.1 and [20]

Most if not all of these time scales are geographically positioned. A day in our normal Gregorian calendar doesn't start everywhere on the world at the same time due to the time zones[19]. The geological ages don't exist everywhere and cover different absolute ages in different regions. Furthermore

[16] "The period in European history between antiquity and the Renaissance, often dated from A.D. 476 to 1453" in http://www.answers.com/middle-ages.

[17] The astronomical unit (AU) and light years (LY) are distances used by astronomers. The actual observations are realized from within our solar system and the observed phenomena occurred back in time according to the distance of the phenomena observed. There is thus a relationship between the distance of these objects and time.

[18] like revs (revolutions of a spacecraft like in the Gaïa trilogy by J. Varley)

[19] The 1st of May 2004 (30th April 2004 at 22:00 UTC - Coordinated Universal Time) 10 new countries jointed the European Union. In the Czech Republic, Hungary, Malta, Poland, Slovakia and Slovenia it was midnight (CEST - Central European Summer Time) but in Cyprus, Estonia, Latvia and Lithuania it was 01:00 (EEST - Eastern European Summer Time). All the countries use Daylight Saving Time (DST). [5] stipulates that it was at midnight CET (22:00 GMT) but obviously this should have been midnight CEST.

the time scales have a range in which they are valid (covering a limited period). The French Revolutionary Calendar[20] is valid from 22nd September 1792 till 1st January 1806 and from 26th March till 20th May 1871 (this second interval is valid in Paris only).

(4) All "time scales" have a (discontinuous) period of time in which they are "valid" in a particular geographic space.

Modern computer systems provide proleptic[21] calendars (a kind of converters) that can be applied without errors back in time until somewhere in the eighteenth or nineteenth century. Beyond that they are often not accurate with regards to the passage from the Julian calendar to the Gregorian calendar. This passage was accompanied with the absence of a few days (10 in most European countries). But even more important, the passage didn't occur at the same time in different countries[22]. And even for these calendars there remains some ambiguity when a human date expression is converted to some shared format (dd-mm-yyyy versus mm-dd-yyyy, and the famous historical millennium problem).

For the historical time line we need calendars/converters that can handle terms like 200 B.P., 197 A.C., 12th century, or Second Dynasty of Shang. At least if humans are allowed to communicate with computers in natural language. Referring to the topic about classes (section 5.1) something to consider also is that a node like "Second World War" can be considered as an event (encapsulating, or related to, a big amount of smaller events) but also as a date (= period in time). Adding (geographically bounded) *begin* and *end* properties to this node enables you to *use* it as a date.

The geological time line has again other characteristics. The terms used are relative stratigraphic terms like Permian, Cretaceous or Bédoulien. All of these terms have an absolute time stamp depending on the geographical region they are applied to. Furthermore there is historical difference in the absolute time stamp because scientists reconsider the real age of stratigraphic terms occasionally. But a particular fossil from a particular stratigraphic unit will always belong to that unit, whatever the scientific world will think about the real absolute age of that unit.

The astronomy time line has fewer problems in itself although the reference point for "5 billion years ago" is not always obvious. The real difficulty

[20]The French Revolutionary Calendar is valid from 22nd September 1792 till 1st January 1806. Officially stated on 5th October 1793 it was applied though from 24th November 1793 till 1st January 1806. It reoccurred during a short period from 26th March till 20th May 1871 (Commune de Paris)[21][11][16].

[21]Proleptic calendars apply a subset of the real rules indefinitely far backward and forward in time. These time scales can be used without any problem for some operations as long as you use only one (proleptic) time scale. Some operations (e.g. providing a duration = number of units) might not provide correct results outside the period(s) in which the implemented rules really apply.

[22]Some implementations, like Java, do have these possibilities but they are only scarcely used.

will occur when you want to annotate a photo of the night sky. There is of course the date (and place) on which the photo has been made, but the things you can see on that photo occurred far back in time and will be different for each stellar object. This is not the case for a classic photo where the moment/period the photo has been taken is also the moment/period of the event or state illustrated on the whole photo (in fact with a negligible difference for the most common usages).

As long as we build applications that limit themselves to one of the time lines there are no real challenges (although not everything has been done yet). But when you need applications that must be able to handle several of these time lines at the same time, you have to connect them in some way or another.

6.2.4 Computer versus human generated times

When talking about time information we should be aware of some fundamental differences related to the way the time information is generated. This can be either programs or people.

Computer generated times

A computer generated form of time is always based in a well defined time scale, at least at the moment of creation. In most cases two factors are influencing this time:

- the choices the programmers made (the kind of "calendar" used and its preferences)
- the specific instance of the computer the program is running on (its internal clock, its localized time settings and its synchronization with the "real time")

Most programs allow the transformation of one representation form of a time scale to one or more other representation forms. Once the time information has been generated it will probably be stored somewhere (else why would it be generated?). The correct usage of that time data afterwards can only be realized if the creation context is known (and can be reused). In most cases however we will assume a specific creation context and that assumption might or might not be correct.

Human generated times

Humans use a great variety of forms for time expressions. They are often understood by convention by a more or less extended community. They assume the same usage (creation) context. In human to human communication some problems exist though if the participants belong to different communities and the necessary precautions are not taken (e.g. on a global level with the time zones). When a Dutch and French person are talking about the Second World War (Tweede Wereldoorlog = Seconde Guerre mondiale) they

take the same major event into account. But when they talk about the begin or end of this war things are more complicated because there are various interpretations of the begin and the end[23]. This second example illustrates another aspect of people using time references. We often use an expression representing a known concept without the need for an exact representation of time borders of this concept.

Computer usage of human generated times

When creating user interfaces for programs the developers often force the user to adopt a time format that can be interpreted by the machine without ambiguity (but the MM/DD and DD/MM errors are still numerous). There are not many programs (with good reason) that allow people to fill in their time data in the way they use them regardless of the domain context. Many programs use a converting algorithm to produce some absolute value for a time expression in a user interface. When the precision of the data from the user interface is different from the precision of the resulting absolute value an error might be introduced. If X is valid from 1997 until 2003 and the precision of the used storage is the second, these values might be stored as the absolute values of 1997-01-01T00:00:00Z and 2003-01-01T00:00:00Z. The fact that the first border is included in the interval and that the last one is excluded is in most cases hidden. In the case where X is valid from 1997 up to and including 2003, the values should be 1997-01-01T00:00:00Z and 2003-12-31T23:59:59Z with inclusion of both limits. Depending on how things are implemented the time zone might or might not be taken into account. If the conversion is realized in Western Europe during the summer the values should become: 1997-01-01T00:00:00+2 and 2003-12-31T23:59:59+2. In the scenario where there is no user interface but a text containing the same information: X is valid from 1997 up to and including 2003. This text has been produced somewhere in the world (production context) talking about a phenomena not necessary in the same location (context of the content) and analyzed again in another place with another time zone (analyzing context), the errors might accumulate.

Example The Java virtual machine lets you access the system time expressed as milliseconds since 01-01-1970. If this value is stored in this form there will be no problem. But if this form has gone through a user interface where it is presented in a canonical format like 4th of July 2004 10:11:12Z there is a loss of precision: some milliseconds disappeared. If the time origin *Z here or UTC or +02:00* has been omitted it could refer to a period lasting for almost 48 hours. If this presentation form in turn is used for validation of a date and the generated amount of milliseconds was the basis of the

[23]On the September 1st 1939 Germany attacked Poland. The French declared the war to Germany on September the 3rd of the same year. The Germans entered Holland on May 10th 1940. On May 5 1945 Holland was liberated by the allied forces. The official end of this war was for the Netherlands on August 15th 1945 but for France already on May the 8th 1945.

default value and the precision the second all should be OK, isn't it?

But imagine a server on one end of the world say in Amsterdam in summer (+02:00) and a user interface in Los Angeles (-07:00). The intention of a user providing a date and hour and the interpretation by the server will probably not take into account the difference in time between these two locations.[24]

6.2.5 Time meta-data

In order to be able to exploit the available information in a "best as you can get" manner, several kinds of meta-data should be associated with each piece of information. We will not handle all of them, but restrict ourselves to the ones that have a direct relationship with time. We distinguish technical data and context data.

The technical data describes time information related to the creation and modification of the information to which it is associated. We can distinguish:

- creation time
- invalidation time[25]

The context time data describes various aspects of the information it is associated with. Some examples are:

- start and end time
- known since
- accepted since and accepted until
- hour of birth and hour of death

There are some subtleties that might occur for some kind of data. For a person e.g. you could say that his "start time" is equal to his "birth date, hour, minute or second" or in some perceptions this might be equal to his "moment of conception". His "end time" might be the "hour of death" for the physical living person, or his "cremation time" for his physical body or even not having any end time at all for its abstract representation. A statement might be true over a complete period or it might become true somewhere in the period. Somebody is alive from his birth moment until his moment of death. But when only a birth year is known his status *is living* becomes true at some moment during his birth year period. In a similar way

[24] We suppose there is no client-side processing that communicates the value from the user interface to a shared format e.g. UTC taken into account the local time settings.

[25] Within a global context of traceability there is no room for "modification time". A "modified" piece of information should be invalidated and a new piece of information should be added.

we can use a time period with or without ambiguity for the associated or derived information. The French presidents in 1995 where François Mitterrand and Jacques Chirac, but on May 28th 1995 it was Jacques Chirac[4].

This consideration leads us to another recommendation on the storage of information and its usage. In traditional (database) systems the selection criteria for the stored information are directed by the intended use. This is perfect for domain specific applications. Within in broader view on the potential use of the available information (for example Enterprize Application Integration), the original reason for the creation of the information is eventually not the only usage that will be made of that information. For that reason we recommend the conservation of "all" available information *as it exists*, regardless of the immediate apparent usefulness of it within a specific context. This means a shift in the way storage systems are conceived (from an information architectural point of view) but several solutions exist nowadays.

6.2.6 Conclusions on dates and time

There are clearly things we should do and things we shouldn't. When a human user or a computer makes a date reference we need to conserve the original expression. Some service (agent or function) will convert this expression to a useful exchange format (derived data)[26]. This converter has a version reference and is possibly using some external information. So we need to annotate the derived data with this information. Future releases of the service or changes in the external data are triggers for the derived data to be re-evaluated. The derived data consists of two elements: the upper and lower limit. The expression "January 2003" is evaluated to 2003-01-01-00-00-00Z and 2003-01-31-23-59-59Z (if the format is yyyy-mm-dd-hh-mm-ss and the unit is seconds). These derived data can be used by other functions/agents to determine whether the referenced information is within a specified time interval.

What you shouldn't do is throw away the original reference and keep only the converted value.

6.3 Geography

The actual Geographic Information Systems (GIS) deal in most cases with co-ordinate systems on Earth. There exist more then 30 different projections and conversion tools between them are freely available. Even for the Moon and Mars there are now geographic co-ordinate systems defined. On Earth the continental drift causes positions of geographic points to change over time. A town on Earth will not be at the same position in a few million

[26]The geographical creation context should be taken into account during the conversion. The exchange format is ideally geographical context independent.

years, nor was its actual location at the same co-ordinates a few million years ago. The height is also a none-stable property (horsts move up and grabens move down). So a geographic position in co-ordinates (XYZ) should have a time stamp. Be aware that the reference point of the co-ordinate system also moves around.

During the actual use of co-ordinate systems various levels of abstractions of accuracy are used and perfectly suitable for the needs. When we change scale and move on to cross-stellar objects geographic co-ordinate comparisons we must face the relative movements of the stellar objects. Each geographic reference not only has a time stamp but also is relative to a reference point. These reference points have their relative movements described. Imagine what you need to calculate the distance of a fly in my car driving on a highway to the hand of one of the astronauts on the future moon base (not a very useful distance but containing all the issues). Furthermore there are situations where geographic position systems are not the traditional geography based systems. The position of a cellular phone can only be estimated in a range (sphere) around its access point or the intersection of several. A network of interconnected PDA's will only have a strange blob form probability position relative to the directly connected other PDA's in the network. The network as a whole might or might not be positioned in the real world like the cellular phone.

It is important to be aware that traditional GIS is not the only thing in life. Complexity appears with scaling (smaller or bigger units) and time aspects.

For these reasons geographic information should *explicitly* contain a reference to the geographic system used to express its values and a time stamp referring to the moment these values are *true*. In order to be entirely complete there should also be a precision associated with the values. This information is not necessarily associated with each geographic co-ordinate property of a node. It can be associated on a model level. The *geographic co-ordinates* property type can have itself a property that relates all the properties of this type with the geographic co-ordinate system on Earth. A simple example is worked out in figure 20.

If you do not provide reference system information, exploitation services must make assumptions for them. Problem with this is that these assumptions will be based on an interpretation of the information as it is available in the network today. But what about the information available in twenty years? Do we still know these assumptions have been made? Are these assumptions still true? Do they hold for all the information in the network or only a part of it? You might think of this aspect as something that will be handled somewhere in the future when the problem occurs. But will the available resources be sufficient at that time to face the investigation of the geographic information in network and find the appropriate solutions?

It is useful to have permanently in mind that a little more effort now en-

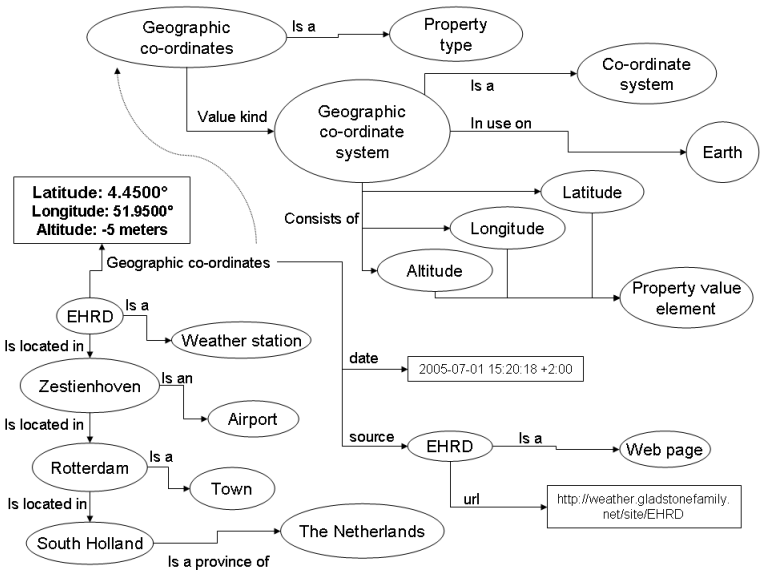

Figure 20: A part of a semantic network showing model elements in the top-right part and some "content" in the bottom-left part. Notice the presence of the source of the co-ordinates and the date and time when this source contained the co-ordinates. Furthermore notice the two trailing zeros in the latitude and longitude values. Although not impossible it is not likely that these digits are correct. They are probably the result of a change in precision after some conversion. The real precision is probably on the second decimal digit. This information is not represented in the picture.

ables operations on the data/information for a long period without extreme costs for information maintenance.

When collecting geographic information from the Internet you will rapidly find out that there seems to be no consistency in the geographic data. There are two major reasons for this.

First the used reference systems might be different. Very often the used system for a particular group of data is not mentioned with the data itself but might be available somewhere in the information associated with the data set. If the reference systems are different this will lead to a systematic error between identical geographic objects in different sets, and this can be detected.

The second origin of this apparent inconsistency is related to the fact that objects have a geographic extension. Co-ordinates for towns are often provided as a point (latitude and longitude couple or triple with the altitude). But towns are no points and even they tend to extend over time. Depending on the way the "central" point of a town is calculated the result will be not immediately identifiable as identical. Some of the co-ordinates

for Amsterdam (Netherlands) are: *(i)* latitude=52°21' N longitude=4°53' E[27], *(ii)* latitude=52°22' N longitude=4°53' E[28], *(iii)* latitude=52°21' N longitude=4°52' E[29] and *(iv)* latitude= 52°22'11.99" N longitude= 4°53'24.00" E[30].

Whenever contour information is available (enclosing box or detailed polygon) you might expect the point information to fit in it. Services that are responsible for geographic information consistency checking or the ones that have to decide whether information is referring to the same geographic object have to take into account a lot of aspects mentioned here.

7 Information properties

During the past decades the majority of the generated information has been stored in media that had a reduced accessibility (books, local databases, etcetera). Since the first days of Internet this medium has become world's most important source of information. But what about its quality? And how to find relevant information? Not mentioning the fact that we often don't want to find documents containing the words we searched for but that we are looking for an answer on a particular question (we are looking for a piece of information).

Fortunately modern search engines allow us to retrieve quite often quickly documents that contain what we were looking for. But will they be able to keep up with the evolution of the Internet? Well, let's hope so, but one thing is sure people must learn how to use the engines to have a chance to find within the first few hits documents containing the information they were looking for.

It becomes more and more frequent that previously existing documents disappear because the owners of sites frequently change hosting companies or simply let their site die and disappear.

This chapter enumerates some properties of information (meta-data) that might be useful for the evaluation of its relevance in whatever point in time in the future. This evaluation of relevance is in my opinion one of the things that will become more and more important as the total amount of available information grows (explodes) and when classification algorithms used by the search engines (the precursors of the answering engines that are coming available) will need more thorough understanding of what the information is (and is about).

[27] source: sn:875dee-eda23a4752-0669df0f031fb83e345267a9679bbc6a (a node in the semantic network)

[28] source: http://www.infoplease.com/ipa/A0001769.html

[29] source: http://www.timeanddate.com/worldclock/city.html?n=16

[30] source: Google Earth

7.1 Relevance decay

The relevance of information can be seen through two different glasses. Information can be relevant for a particular user at a specific moment in time. This is what we call the user context based relevance[31]. If we could take all potential users together (all human beings and software agents/services) we might define a global or absolute relevance. This global relevance would be time-based. The global relevance has a certain start value (not necessarily the maximum) and in many cases this relevance will become less important over time (relevance decay). We might define several relevance decay models (RDM) describing the global relevance over time. A linear flat model will indicate that there is no decay over time (e.g. the relationship between article and its author has such a model). A news article will be important (or not) at the time of its creation but will probably loose quite quickly its importance (globally spoken, not from a historical point of view). The kind of model would essentially be determined by the nature of the information. Its half-life depends on the environment of it. Factors influencing might be the number of new information items on the same topic, the usage of it, etc. The half-life factor might change over time when the environment is changing and will remain stable afterwards letting the relevance decrease over time (the information is left alone).

Relevance should never reach the level zero because the exploitation context (user) should be able to increase it for this particular context (and will probably apply some multiplying factor). The relationship between an author and one of the articles he has written (the inverse is the previously mentioned flat horizontal line) follows such a curve in some cases. If a person realized its PhD on a particular subject (e.g. in geology) but changed its professional activity to something completely different afterwards (e.g. semantic network technologies), then, twenty years later, the fact that that person wrote some articles about geology is not relevant[32] anymore for that person. All the information about that person, related to geology (the person's geological part of its environment), has become more or less irrelevant.

The actual search engines classify/order documents compared to each other in relation to the expressions provided in the query. The ranking they apply takes some absolute value depending on the parametrization of the indexing algorithm(s).

Recent work[33] has shown that contextless [34] relevance decay of a piece of

[31] Most of the literature handling information relevance does so in combination with Information retrieval (IR).

[32] The expression "not relevant" does not mean that its relevance expressed as a numeric value is zero. Its meaning can be seen as "not relevant enough".

[33] Study executed in the ICIS project, Architecture cluster under the topic "Information Relevance Decay".

[34] Contextless is in fact not quite correct. It would be better to talk about the sum of all possible exploitation contexts.

information cannot be established. It appeared impossible to define a decay model for a piece of information that would be valid in whatever exploitation context.

The idea is appealing but as far as we know now its practical use is reduced at a level of detailed information. It is reduced because for some kinds of information one might establish a general model (e.g. flat lines for authors), but for most it will make no sense.

If Relevance Decay Models are applied to concepts (i.e. the nodes in the network), instead of pieces information of those concepts, things become much easier. Imagine (in a global context) the relevance curves of some craftsman and a politician. Are they the same? Do they have similar forms? Which states make the inflexion points of those curves? What might be key events that may change the shape of the initial curve? And what happened with the planet Mars (imagine the curve in a perspective of the past fifty years)? How could an automatic process have detected the regain in interest (relevance)?

7.2 Time dependency

Each piece of information is situated on a time line. And in fact it exists on two time lines.

The first time line is the one on which the existence of the information itself is indicated. This information "Availability Time Line" (ATL) starts at the creation of the information and theoretically lasts until the destruction of the universe (or perhaps even beyond that point). An ATL can consist of several location parts indicating where the information is available: in the head of some individuals, on paper or in information systems. ATL will be used whenever tracing of causes of actions is necessary, by determining what was known by who en when.

The second time line could be called the information "Life-cycle Time Line" (LTL) illustrates the valid state of the information. Some kinds of information are valid from the moment they express a real existing fact and keep that state forever (the birthday of a specific person is example of this kind). Other information has a limited period in time in which it is valid (the address a person is living or its email address). Some other LTL can be discontinuous: the position of a person will be the same with a certain regularity: in bed or in the chair at his office. Granularity of this kind of information can change its aspects completely. Somebody can have lived its whole life in a particular country, while he can have lived at several different addresses[35].

[35]One general remark is in place here. When deciding what information to store for future use it is not recommended to apply information reduction (e.g. if you know the exact birthplace of a person - the building - do not change/reduce it to the town. You or somebody else might regret it sometime in the future. Where in the past storage space

7.3 Community dependency

A large amount of information is useful only within a specific community. Nicknames are a good example because outside a community in which the nickname is known and used one cannot communicate using that name (it doesn't refer to an identifiable person). The language dependency of names (different communities based on the use of different languages) is a generalized form of this. The nickname remains valid outside de community but it is not useful.

7.4 Persistency

When you are exploiting the information available in the actual systems there is an implicit context: actual valid information (or at least the most recently known). Few systems allow you to change this context element. Answering questions like "what was available in the system at a given moment in time" are useful for the sake of traceability of e.g. inferences realized at that moment. To be able to answer these kinds of questions no information should be permanently removed. When new values for properties are known the old values should be invalidated (time dependency) and the new ones added. In the same way a divorce doesn't mean the "is married to" relationship should be removed. It should be invalidated and the "divorced from" relationship should be added. Replacing the above relationships with one single relationship "has been married to" with a start and end date might seem another possibility. But this will disable the traceability of the state of the network (till when it wasn't known that they had divorced). The "Lifecycle Time Line" (LTL) will still be correct but "Availability Time Line" (ATL) will probably be completely different. The "has been married to" is though a virtual relationship that can be easily inferred from the existing information.

8 Acknowledgements

This paper has been realized with funding from TNO through the DECIS Laboratory by the means of the ICIS project. Thanks to all the people who were willing to have (sometimes very deep) discussions during the past fifteen years on some of the fundamental aspects mentioned in this paper.

was expensive, it isn't anymore. But there is a counter side on this choice. If for your actual needs you have to access the birth town or country you will need a "smart" service that derives this information from the information originally provided.

References

[1] Anonymous. Event - definition by dict.die.net. http://dict.die.net/event/, 02 2002.

[2] Anonymous. leg - definition by dict.die.net. http://dict.die.net/leg/, 02 2002.

[3] Anonymous. Glossary - event. http://www.w3.org/TR/2003/WD-DOM-Level-3-Val-20030205/glossary.html, 02 2003.

[4] Anonymous. 22 présidents depuis 1848. http://www.droitconstitutionnel.net/22presidents.html, 2004.

[5] Anonymous. Cnn.com - eu welcomes 10 new members - may 1, 2004. http://www.cnn.com/2004/WORLD/europe/04/30/eu.enlargement/, 05 2004.

[6] Anonymous. The unified medical language system (umls) semantic network. http://semanticnetwork.nlm.nih.gov/, 05 2004.

[7] Anonymous. Definition of antonymy - wordreference.com dictionary. http://www.wordreference.com/definition/antonymy, 2005.

[8] Anonymous. Google search: define:polysemy. http://www.google.com/search?q=define:polysemy, 2005.

[9] Anonymous. Polysemy - glossary definition - usingenglish.com. http://www.usingenglish.com/glossary/polysemy.html, 2005.

[10] Anonymous. Tree of life web project home. http://tolweb.org/tree/phylogeny.html, 2005.

[11] Arno Lagrange et al. Commune de paris (1871). http://fr.wikipedia.org/wiki/Commune_de_Paris_(1871), 07 2005.

[12] T. Gentenaar and J. Tiel Groenestege. Enhancing information. information extraction and concept matching using the semantic network engine. Master's thesis, University of Utrecht, Netherlands Organisation for Applied Scientific Research (TNO), 2005.

[13] Lisbeth Klastrup and Susana Tosca. Transmedial worlds - rethinking cyberworld design. In *Proceedings International Conference on Cyberworlds 2004*, pages 409–416. IEEE Computer Society, 2004.

[14] Carl Lagoze and Jane Hunter. The ABC ontology and model. In *Dublin Core Conference*, pages 160–176, 2001.

[15] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax specification, 1998.

[16] Peter Meyer. Julian day numbers. http://www.hermetic.ch/cal_stud/jdn.htm, 03 2005.

[17] Opencyc. http://www.opencyc.org/.

[18] S. Pepper. The tao of topic maps - finding the way in the age of infoglut, 2000.

[19] Ronald Poell. Notion system. http://www.notionsystem.com, 2001.

[20] Roser Sauri, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. Timeml annotation guidelines. http://www.cs.brandeis.edu/~jamesp/arda/time/timeMLdocs/annguide12wp.pdf, 2004.

[21] The Historical Maritime Society. Nelson and his navy - revolutionary calendar. http://www.hms.org.uk/nelsonsnavyrevcalend.htm, 2001.

[22] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.