Automatic Generation and Evaluation of Sentence Graphs out of Word Graphs

Dennis Reidsma

University of Twente, Department of Computer Science
Parlevink Group, The Netherlands
dennisr@cs.utwente.nl

Abstract. This paper reports on the development of a system that automatically constructs representations of the meaning of sentences using rules of grammar and a dictionary of word meanings. The meanings of words and sentences are expressed using an extension of knowledge graphs, a semantic network formalism. Furthermore the system contains an algorithm that calculates an evaluation measure for alternative sentence graphs of one sentence, making it possible to disambiguate the meaning of sentences on semantical grounds.

1 Introduction

This paper reports on a project in which a system was developed that automatically constructs representations of the meaning of sentences using rules of grammar and a dictionary of word meanings. The meanings of words and sentences are expressed using an extension of knowledge graphs, a semantic network formalism developed by Hoede et al. at the University of Twente. The most important extension was the introduction of link weights to support the algorithms described in this paper and to improve the expressive power of the formalism. The so-called *sentence graphs* are created using knowledge about the grammatical function of parts of the word graphs. The unification is completed by removing redundant information using a similarity measure expressing the amount of contradictory or complementary information introduced into the sentence graph through the different word graphs and their grammatical relations.

Furthermore an algorithm will be presented that can be used for ambiguity resolution. This algorithm evaluates different possible representations of the meaning of a sentence using a variant of the similarity measure mentioned above. Both algorithms have been implemented in a working system, which has been tested on some cases of PP-attachment disambiguation and lexical disambiguation.

The paper is organized in the following way: the first section contains a short presentation of the knowledge representation formalism of Hoede et al. That section also describes link weights, the new extension to the formalism. The second section is about constructing a representation of the meaning of sentences within this formalism. The third section describes the outline of the algorithm

U. Priss, D. Corbett, and G. Angelova (Eds.): ICCS 2002, LNAI 2393, pp. 151–165, 2002.

that calculates an evaluation measure on the different possible sentence graphs used to choose the best alternative in cases of lexical or syntactical ambiguity. The fourth section is about a variant on this algorithm that is used to remove redundant information from the resulting sentence graphs. The paper ends with a discussion of some preliminary testing of the algorithms and the conclusions.

2 Knowledge Graphs

This first section will introduce the formalism that has been used for knowledge representation in this project. This is de Knowledge Graph formalism, developed at the University of Twente by Hoede et al [3]. Those knowledge graphs have been described extensively elsewhere, so only a short description will be given here.

2.1 Introduction

The formalism is one of subjective and intensional semantic networks. Its main characteristics are the fact that it aims at mimicking actual thought patterns in the human brain and the fact that this should be done using only a few relation types. Willems [7] describes the restrictions that are placed on these relations in the theory of knowledge graphs. The most important of these restrictions are:

- The relations should be low-level, i.e. it should not be possible to split them into combinations of other (smaller) relations.
- The relations should have no overlap, because if they had a clear overlap it would be possible to split them into relations i.e. for the common part and the rest.
- The relations are designed to be abstractions of human understanding, because the knowledge graphs were designed to be a model of human thinking.

Reidsma [5] analyzed the knowledge graphs in the context of the schema [1] in Fig. 1 (derived from [2]). This paper contains only the schema, for the full evaluation see [5]. The schema contains three areas and their relations:

- The area of *language*: this is where you find the words and the language expressions.
- The area of *intension*: this is where you find the concepts. A concept is an idea of something, occurring in a human mind.
- The area of *extension*: this is where you find the (sets of) objects and situations in a world. These worlds need not necessarily be real: it is possible to reason about green elephants or white Martians.

[1] This schema, together with the corresponding outlook on mathematical abstraction, is the result of many discussions with Jan Kuper, for which I would like to thank him greatly.

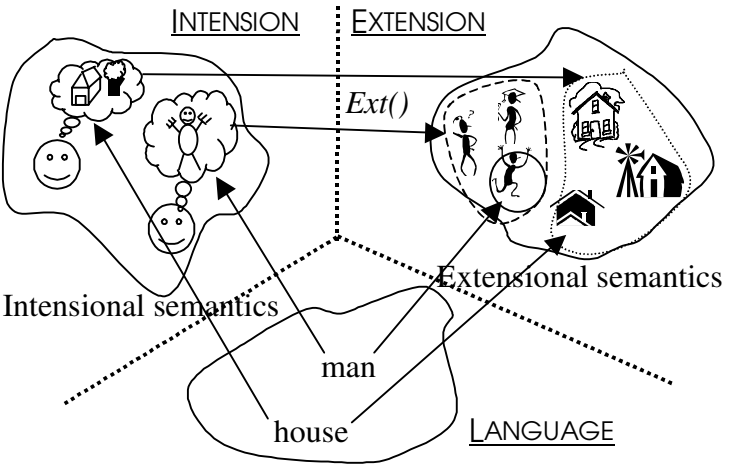

Fig. 1. The intensional triangle

This schema, or a variant thereof, plays a central role in many knowledge representation theories. The knowledge graphs are placed in the upper left corner, in the area of intension. A graph describes a concept. This means that the domain of the function $Ext()$ is the set of all graphs and its range is formed by all sets within some reality. $Ext()$ maps a concept to the set of all that "satisfies the concept".

This "satisfying the concept" can be subjective and flexible (such as with the concept "big" in "a big house"). It might be possible that someone perceives something as satisfying a certain concept even though it does not have all characteristics as described in the graph (the intension). For example, the intension of "christmas ball" may state that christmas balls are fragile, since they are fairly easy to break. But if a christmas ball were made very strong, this would not necessarily mean that it would not be a christmas ball. It might for example be considered an "unbreakable christmas ball", which is still a christmas ball. This flexibility will be clarified further in the next section.

2.2 Changes to the Formalism

During the development of the system a few changes have been made to the knowledge graph formalism. The first change followed from the formal analysis of the formalism in [5], where it was shown that one of the primitive relations that was introduced to express quantification had not been defined or used correctly and consistently. However, this only resulted in removing the relation; no new relation was suggested to fulfill this role. The second change concerned the introduction of *weighted links*. This change will be discussed in the rest of this section. The last change concerned the introduction of *role nodes*, which are discussed in section 3.

Justification for Link Weights. Weighted networks are fairly common. There are several arguments for using link weights in knowledge graphs. First of all it is intuitively a correct way of describing relations between concepts, as some parts of a concept are considered to be more important than others. Furthermore the link weights result in several properties of the knowledge graphs, some of which are described here, that are an improvement of the formalism.

- Flexibility: Without weighted links there are only two ways to change the semantics of a concept: a link can be added or removed. Weighted links make gradual changes in a concept possible.
- Cumulative influence of relations: Given a sentence graph resulting from the unification of two partial sentence graphs, the right use of link weights make it easy to express in the unified graph that some association is more important when it was present in both partial graphs than when it existed in only one of the partial graphs.
- Context influence: Given a context graph and a sentence graph, link weights make it possible to emphasize those relations in the sentence graph that are also present in the context graph and de-emphasize other relations, without throwing those other relations away.

Link Weights and 'Meaning'. The link weights have an impact on two operations: *the comparison of two graphs* and *the evaluation as to whether something in the world satisfies a certain concept*. Those two operations resemble each other greatly but occur in different places in the triangle in Fig. 1: the first is between two graphs (or the concepts represented by those graphs) in the area of intension (upper left corner), the second between intension and extension (the line between the upper left and upper right corner).

Determining the Extension of a Concept. In Sect. 2.1 we defined the extension of a graph to be all instances in the world that "satisfy the concept as expressed by the graph". This "satisfying" can be evaluated: one by one all aspects of the instance will be compared to the concept. The outcome of this evaluation is determined by the number of contradictions and correspondences between them. Link weights determine the *importance* of a link for the concept. When some relation in a concept has a high weight this means that it is important that this aspect is present in the instance in reality if the judgement should turn out positive. The example below will make this clear.

Example 1. In the concept "red ball" the aspect of the red color is connected with a high weight to the ball concept. When a particular ball that is blue instead of red is compared with the concept "red ball" it will turn out not to satisfy the concept. It satisfies all aspects except the color aspect, but since this color aspect is connected to the ball concept using a high weight the negative effect of this deviation is high.

Example 2. Glass as a material is usually transparent. On the other hand, everybody knows examples of glass that is opaque but still considered to be glass. This suggests that the aspect of transparency is connected to the material concept with a low weight.

Comparing Graphs. When comparing two graphs, the weights determine how much influence the presence or absence of some aspect has on the measure of equality. This operation is much like the previous one, so another example will suffice to clarify this.

Example 3. The concepts "book" and "cigar-box" resemble each other very much in form and size aspects, but differ greatly in use aspect (transferring information versus storing cigars). The concepts "book" and "computer document" on the other hand are of a completely different form, but their uses are very much alike. When people compare those three concepts, they are inclined to cluster "book" and "computer document" together. This suggests that in those concepts the aspect of use is connected in the concepts with a higher weight than the aspect of form.

Remarks. The effect of a *negative link weight* on the semantics as expressed by a graph are simple: when some aspect is connected using a negative weight it means that the presence of this aspect in the instance in reality will have a negative influence on the judgement whether it satisfies the concept.

When using link weights it is also very important to remember what a link weight does *not* mean (though these are sometimes related). A link weight does *not* mean:

- the fraction of all people that think this association belongs in the concept.
- the fraction of the elements in the extension of a concept for which one person thinks this relation holds. Even if an association holds for everything in an extension it does not necessarily mean that this association is vital for satisfying the concept, as is shown in example 4. This difference could be called the difference between "defining" characteristics and "incidental" characteristics.
- the statistical chance that the association holds for something in the extension (a somewhat more objective description of the above cases)

Example 4. Stars are roughly spoken a mass of a certain chemical composition in which a certain reaction takes place, which are more or less ball-shaped. However, most people would probably still call a mass of that size, composition and temperature a star if it were shaped like a pyramid or a box.

Link Weights: Form and Operations. Every link in a knowledge graph has a weight associated with it in the range $< -1..1 >$. To ensure that the weight values stay within reasonable limits, without losing the monotonicity of

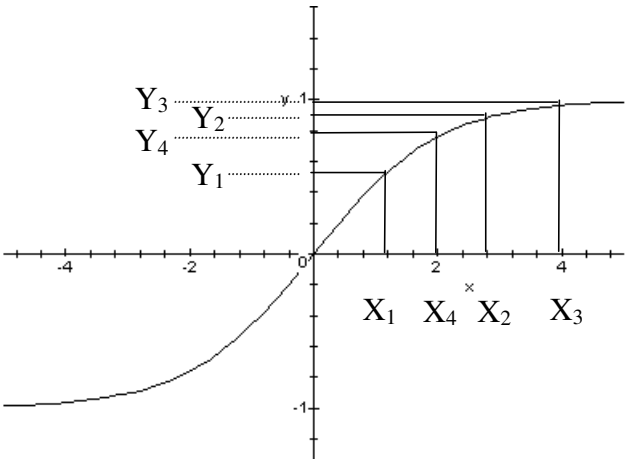

Fig. 2. The sigmoid function. $y_3 = f_{\text{boundedAdd}}(y_1, y_2)$, $y_4 = f_{\text{boundedMean}}(y_1, y_2)$

addition, weights are added and substracted using the sigmoid function that is also commonly used in neural networks (see below). Multiplication is done using the standard multiplication operation.

$$f_{\text{SigLog}}(x) = \frac{2}{1 + e^{-x}} - 1 \tag{1}$$

$$f_{\text{boundedAdd}}(y_1, y_2) = f_{\text{SigLog}}\left(f^{-1}_{\text{SigLog}}(y_1) + f^{-1}_{\text{SigLog}}(y_2)\right) \tag{2}$$

$$f_{\text{boundedMean}}(y_1, y_2) = f_{\text{SigLog}}\left(\frac{f^{-1}_{\text{SigLog}}(y_1) + f^{-1}_{\text{SigLog}}(y_2)}{2}\right) \tag{3}$$

The operations mentioned above are also used in link integration. Link integration rules can be devided into two types. *Path integration* derives the relation between two end points of a path from the links along the path. *Parallel integration* combines two edges between the same nodes into one edge. Two links with the same label for example will be combined into one link with the same label.

The system that has been developed uses a path integration similar to the one described in [6]. The weight of a path is defined as the product of all weights along the path. In parallel integration the weight of the new link is defined by the operation $f_{\text{boundedAdd}}$ on the individual weights. These link integrations are used extensively in the algorithms described in the following sections.

3 Creating Sentence Graphs

One of the most important aspects in the project concerned the manipulation of knowledge graphs in a language processing environment. More specifically, one

of the aims was to develop algorithms that would use a lexicon of word graphs and some rules of grammar to create sentence graphs expressing the meaning of a sentence. Other projects in this direction are the work of Willems [7] and Hoede [4]. The method described in this paper is partly inspired by these two projects but uses a different approach. In [5] we analyze a few advantages and disadvantages of those two projects and extract some requirements from this analysis, which are presented in Sect. 3.1. The rest of the chapter describes the syntactical unification process in greater detail.

3.1 Requirements

Connecting the Right Nodes. The first of these requirements concerns the linking of word graphs in the right places. Take for example a verb expressing a state change in the object, such as "kill". Killing someone causes him or her to be dead. So the graph expresses "causing someone to be dead", which means that there is a node or subgraph in the verb graph standing for the person who dies. When you use the verb, as in "Kennedy kills Oswald", you want the sentence graph to express among other things that Oswald ends up being dead. In fact, you want the graph representing Oswald to *replace in the verb graph the subgraph standing for the unspecified someone who is the victim*, since by now we know who actually dies.

So for creating a sentence graph it is not (always) enough to simply draw an arc between the frames surrounding the word graphs or even between nodes in the various word graphs. Sometimes a subgraph in one word graph must be replaced with a subgraph from another word graph.

Retaining Similarities. This requirement is a well known issue in language technology. It concerns the problem of keeping similarities in meaning intact when the grammar of a sentence is completely different. This is very important when you want to compare the meaning of different language expressions, as for example in search technology or text generation. The importance of this requirement is most easily expressed with yet another example:

Example 5. The verb "break" can be used in may ways. Transitive, intransitive, with or without prepositional phrases, anything is possible. The intensional semantics of this verb overlaps in all cases: something is caused, possibly by a person and possibly using an instrument, to turn into fragments. Abstracting from other details, take a look at a few sentences using the verb "break":

S_1: "The glass breaks."
S_2: "The brick breaks the glass."
S_3: "Peter breaks the glass with the brick."

Even though the syntactical relations are different in all sentences, the semantical graphs should be similar on the parts where they overlap. So all three sentence graphs should express the same information about the glass turning into fragments, while the role of the brick in the last two sentences is the same, despite the syntactical differences.

The Intensional Triangle. Combining the word graphs into a sentence graph should be consistent within the intensional model of meaning. A sentence graph is also a knowledge graph and should therefore be interpreted within the triangle of Fig. 1.

3.2 Overview of the Process

We assume that a parse tree of the sentence is available, providing the grammatical relations between the words. Furthermore we have a lexicon containing a word graph for every word. The problem then is to connect those word graphs into a sentence graph. This is done using so called *role nodes*. Role nodes are stored in the lexicon and define the syntactic function of a few of the nodes in a word graph for each lexicon entry. This is a mechanism similar to the γ functions of Willems [7].

Starting at the leaves of the parse tree the word graphs are glued together along these role nodes. Where needed the information about the role nodes is retained and passed up the tree, so it can be used to glue larger substructures together. In this way the tree is followed upwards until at last the largest substructures are glued together to form a sentence graph.

The next section will explain in greater detail how the role nodes have been used.

3.3 Roles

In a language processing system a lexicon is used to store information about words. In the system that has been developed during this project entries in the lexicon contain not only grammatical information about a word but also a reference to a knowledge graph expressing the meaning of the word. As an example consider the graphs in Fig. 3. The graphs presented there are naive versions of word graphs for the words “man”, “to paint” and “painting” as noun. (Better graphs could be devised, but these suffice for illustrating the role nodes.)

When one tries to manually unify these graphs to a sentence graph for “A man paints a painting” a few observations can be made. In the verb graph you can clearly identify a node that represents the one who is doing the painting (the left hand node) and a node that represents the resulting painting (the right hand node). Furthermore it is possible to identify a node in the graph for “man” that would represent the actual man (the central token in the “man” graph): the other nodes are simply associations and qualifications of this “man” concept (the graph expresses more or less the following sentence: “a man is something like an adult male human”). For “painting” the central node has the same function.

These observations lead to the suggestion that those nodes are a good starting point for the unification process. This unification is done using syntactical considerations: “man” is subject for “paints” so the central token of “man” will be unified with the left hand node of “to paint”.

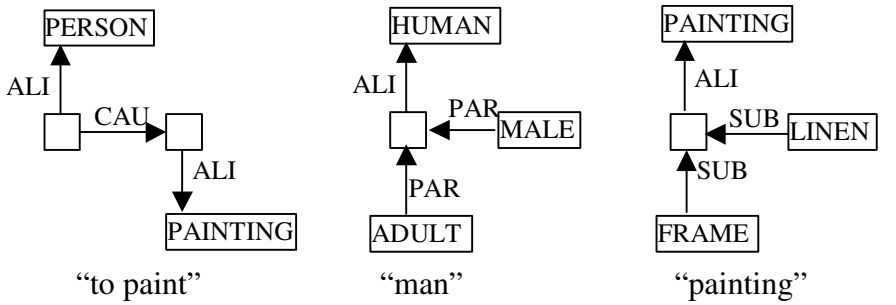

Fig. 3. Some word graphs

Those special nodes will be called *role nodes*. Role node information will be added to every entry in the lexicon. It is important to note that this information belongs to the lexical entry instead of to the concept graph, since it is possible that two entries refer to the same concept but have different role nodes. An example of this is the transitive and intransitive use of the verb "to break": in the transitive version, the subject node is the person who causes the breaking, whereas in the intransitive version the subject is that which breaks.

Similar kinds of lexicalized syntactical and semantical information are used in many other theories. For example, these roles are reminiscent of the f-layer in Lexical Functional Grammars. That theory uses a distinction in a c-layer, where grammar describes the allowable word orders, and an f-layer that describes features and functions of words. Example 6 (taken from the LFG mailing list [1]) shows what kind of information that f-layer contains.

Example 6. Consider the verb "hand", as used in the following two sentences:

S_1: "Evan handed a toy to the baby."
S_2: "Evan handed the baby a toy."

The grammatical structure of both sentences is different, but their semantical content is the same. In the theory of Lexical Functional Grammars this correspondence is defined by providing templates for the functional structure, e.g. (hand (Agent) (Theme) (Goal)), and defining possible grammatical mappings to this structure, like (hand subj obj objDat) for the first sentence and (hand subj obj obj2) for the second sentence.

Role Types. Every theory using something like roles has its own lists of role types that are allowed. In this project the requirement was that the roles should only express syntactical information. This means that a role like "patient" cannot be used, since the patient of a verb is a semantical function that may be fulfilled by the subject or object in different situations. Such semantical functions should be represented by the structure of the word graph. These considerations led to the role types described below.

- The head of the graph: The head of the graph defines the central node to which external links from other words will be connected. In the noun "dragon" for example a node can be identified that stands for the actual dragon. All other nodes and relations in the graph are *aspects* of the dragon, expressing concepts like "a dragon has scales" (some SUB relation) or "a dragon likes gold". When a dragon is painted red (probably for camouflage in the fires of the burning village) it is the head-node of the graph to which the attribute "red" should be connected. Willems [7] also identifies head-nodes in a graph (p 59: "a terminological k-graph with head h").
- Subject and object in a verb: Figure 3 gives an example of how these roles could be used. Connecting the word graph of the object and the verb for example will be done by combining the head-node of the object graph with the object-node of the verb graph into one new node.
- Prepositions: Those are a special case and will be discussed in the following subsection.

Role Nodes and Prepositions. Prepositions are a special case in this discussion, as has already been recognized in many other projects. In one of the previous projects on knowledge graphs and language, Willems [7] maps propositions on arcs between noun and verb graphs. This project however follows the structural parsing process of Hoede and Zhang where prepositions are treated on the same level as other word types [4]. Prepositions must be discussed in two contexts: what roles they have and whether they call for extra roles for words like nouns and verbs.

The first issue is quite straightforward. Syntactically, a preposition connects two parts, be they a noun phrase and a verb phrase, or two noun phrases. This leads to the roles of "first part" and "second part".

The second question is harder to answer. When a word graph contains explicit information that could be expressed using a prepositional phrase you might want to identify this information using a role node. Examples of this are cases such as "breaking with" (instrumental) or "to bury under". The first case might for example result in a role "with" in the graph for "to break". This would make it possible to pinpoint the exact location in the graph of "to break" where the graph of the noun in the propositional phrase should be attached. This could improve the quality of the sentence graphs.

There are however drawbacks to this approach. The main reason for this is the fact that with any given verb one proposition might be used to express very different meanings. This would result in an explosion of role node information in the lexicon, adding a role node for every possible use of the proposition. This is a commonly known issue in lexical semantics. When the effect of propositions is (partly) included in the semantics of a noun or verb the amount of information in the lexical entry for that noun or verb grows explosively. Furthermore this type of propositional roles is not entirely syntactical in nature as was required. For these reasons we decided not to introduce any extra roles for this second aspect of prepositions.

4 Evaluating and Comparing Sentence Graphs

This section describes an algorithm that calculates a relative ranking for the different possible sentence graphs for one sentence in case of lexical or syntactical ambiguity.

When a sentence graph is created using the method presented in Sect. 3, there may be a lexical or grammatical ambiguity, resulting in graphs expressing different possible meanings. The algorithm calculates a measure expressing the amount of complementary or contradictory information in a graph. The relative values of the evaluation for the different alternatives give an indication for which alternative is the best. Example 7 shows how word graphs can complement or contradict each other.

The evaluation value has the following characteristics, related to the information expressed by the different word graphs:

- When the word graphs contradict each other with respect to a certain aspect the value will turn out more negative or less positive
- When the word graphs give the same information with respect to a certain aspect the value will turn out less negative or more positive
- When a link has a higher weight, its influence on the value of the judgement will be higher, either positively or negatively.
- When the different word graphs contain unrelated information this will have no influence on the judgement.

Example 7. In Sect. 3.1 the verb "kill" was discussed in relation to the sentence "Kennedy kills Oswald". Suppose that the lexicon contains two entries for "Oswald": one is an entry for the person Oswald and the other is an entry for a space ship named "Oswald". The syntactical unification process will create sentence graphs for both possibilities. The graph for "kill" might express among other things the fact that the victim of a killing is a living entity. The entry for the person Oswald would contain the information that it is indeed a person. The concept person would contain the information that a person can be alive. So indirectly the sentence graph for this alternative will contain the complementary information that Oswald was a living entity because he is a person and because he is the victim of the killing. The sentence graph with Oswald as a space ship contains the information that Oswald is a living entity because he is killed and the contradictory information that he is not a living entity because space ships are, generally speaking, not living entities.

4.1 The Algorithm

The algorithm is based on the observation that the only nodes where the complementary or contradictory information will occur are those that have just been created through the unification of two or more nodes. The algorithm starts with a syntactically unified (partial) sentence graph and a list of the syntactically unified nodes. For every syntactically unified node an evaluation value will be

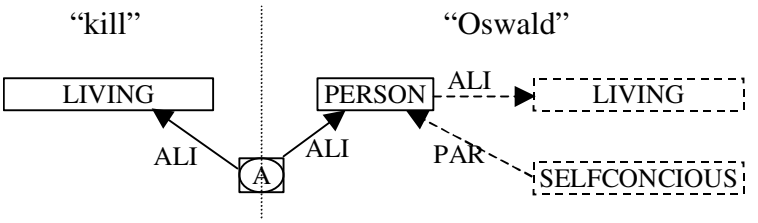

Fig. 4. Evaluating one unified node

computed. The operation $f_{\text{boundedAdd}}$ on all those partial evaluation values will yield the final judgement. The remaining part of this section explains how these partial evaluations are achieved.

Figure 4 shows the situation. The node marked A resulted from the unification of the head nodes for the victim and for Oswald. The essence of the evaluation is to find which nodes can be reached from this node A through paths out of the "kill" graph and through paths out of the "Oswald" graph. Whenever necessary labeled nodes are expanded (in the example the node for "Person" has been expanded resulting in the dotted extension) and nodes with the same label are considered to be the same node since they express exactly the same information. All different paths starting at node A through links out of the verb graph are collected, as well as for the "Oswald" graph. Only paths in which the product of all link weights is above a certain threshold are considered. When the product of link weights falls below this threshold the information in this path is considered not important enough to be included in the evaluation. Figure 5 shows what paths will be found in the example.

The next step is finding out what the relation is between the A node and the end node of every path. For this an associative link integration operation is defined, which defines for every path of length two the integrated path of length one, depending on the direction and type of the arcs. Two SUB arcs with the same direction for example integrate to one SUB arc, but a SUB and a PAR

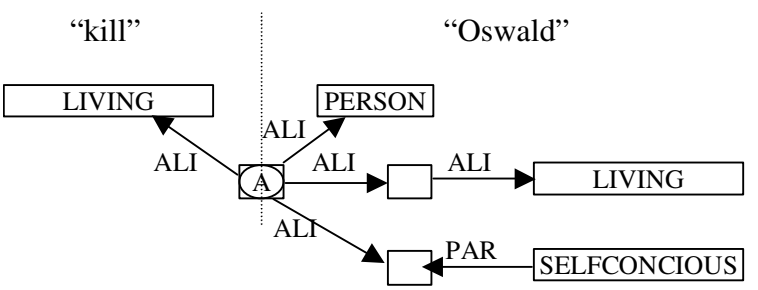

Fig. 5. The resulting paths

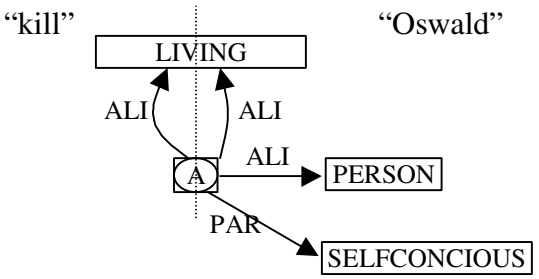

Fig. 6. The resulting links

arc in different directions do not integrate so a path containing that sequence does not integrate to a path of length one but is removed from the set. The weight of the integrated paths is the product of the individual link weights.

So now there is a set of links between the A node and a lot of other nodes, either resulting only from information in the “kill” graph or from information in the “Oswald” graph (see Fig. 6). The final step is to take every end node of those links and consider its relations to A: for every combination of a relation through the verb graph and a relation through the Oswald graph the effect on the evaluation measure is calculated. This is done using a table describing for every two relation types how much they contradict or complement each other. The value from this table is multiplied with the link weights of both links and added to the evaluation using the bounded add operator.

In the example, the labeled node “living entity” can be reached through both subgraphs, resulting in both cases in an *ALI* arc with positive weight. The table returns a value of 1 for two *ALI* arcs, so the positive influence of this complementary information on the evaluation is the product of the link weights of the two integrated paths. If the “Oswald as space ship” alternative was considered, the link between A and “living entity” created from the noun graph would be a negative *ALI* link since space ships are usually not living entities. That would result in a negative influence on the evaluation value: the product of a positive and a negative weight, expressing contradictory information.

5 Semantical Unification

Section 4 showed that the sentence graphs created through the method in Sect. 3 may contain redundant information. The algorithm presented in this section aims at unifying these redundant parts to keep the graphs clear and concise. When for example a node in the syntactically unified graph has both an *ALI* link to a node with the label “human” and an *ALI* link to a node with the label “man”, the link to the node labeled “human” could be removed, at the same time increasing the weight of the link to “man”. This way redundant information is unified but the impact of the removed information is kept intact. The algorithm presented here was also developed in this project and implemented in the working system.

The algorithm for semantic unification is very similar to the evaluation algorithm presented in the previous section. Once again the algorithm starts with a syntactically unified graph and a list of unified nodes. Those are exactly the nodes where the redundancy will occur [5]. So the semantic unification starts there: every combination of two neighbours from *different word graphs* of such a syntactically unified node are considered for further unification. Whenever two such nodes are unified, their neighbours are in turn considered in the same process.

Given two such neighbours from one of the unified nodes, the evaluation whether they should be unified is based on what is known about these two nodes. First of all a table is consulted to check whether the relations between the unified node and these two neighbours allow unification. Then all information connected to the two nodes is evaluated by creating exactly those same integrated paths that were used in the evaluation algorithm. This time however the starting points of the paths are the two neighbours. Finally, the following decisions are made:

- If, for one of the neighbours, for every link from the unified node to that neighbour some path can be found through the other neighbour that integrates to that same link, all links from the unified node to this first neighbour will be removed from the graph, since that information is also implicitly present in the other node.
- If the above is not true, it is possible that the two neighbours express the same information to such a large degree that they can be unified into one node. This is done using the same evaluation method as described in the previous section. When the evaluation of how much similar, non-contradictory information they express gives a value above a certain threshold, the nodes are unified into one node.
- When none of the above is true, the nodes will not be unified and no links will be removed.

6 Conclusions

The algorithms described in this paper have been implemented in a working system. Preliminary tests indicate that both the generation of sentence graphs and the evaluation of those graphs in cases of ambiguity can be done automatically. The quality of the resulting graphs is good. One of the main problems that has to be solved however is the fact that building a suitably large lexicon is not easy. At the moment it still takes a lot of time to create the word graphs, which means that large scale testing of the system has not yet been done. One of the goals for future research is finding a way to solve this, for example by automatic conversion of knowledge from other network lexicons or automatic lexicon generation from a starting lexicon and a corpus of text.

Another theme in ongoing research is the application of the system in a practical dialogue environment.

References

1. Mailing list item from the lexical fuctional grammar mailing list. http://clwww.essex.ac.uk/LFG/Burquest/. 159
2. L. T. F. Gamut. *Logic, Language and Meaning, Intensional logic and logical grammar*, volume 2. 152
3. C. Hoede and L. Zhang. Word graphs: The third set. Memorandum 1526, University of Twente. 152
4. C. Hoede and L. Zhang. Structural parsing. Memorandum 1527, University of Twente, 2000. 157, 160
5. D. Reidsma. Juggling word graphs, a method for modeling the meaning of sentences using extended knowledge graphs. Master's thesis, University of Twente, August 2001. 152, 153, 157, 164
6. H. van den Berg. *Knowledge Graphs and Logic, One of Two Kinds*. PhD thesis, University of Twente, 1993. 156
7. M. Willems. *Chemistry of Language, a graph-theoretical study of linguistic semantics*. PhD thesis, University of Twente, 1993. 152, 157, 158, 160