Research on Language and Computation manuscript No.
(will be inserted by the editor)

Performing aggregation and ellipsis using discourse structures

Mariët Theune[1], Feikje Hielkema[2], Petra Hendriks[3]

[1] Human Media Interaction, Department of Computer Science, University of Twente, The Netherlands, e-mail: M.Theune@ewi.utwente.nl
[2] Department of Computing Science, University of Aberdeen, Scotland, UK, e-mail: fhielkem@csd.abdn.ac.uk*
[3] Center for Language and Cognition, University of Groningen, The Netherlands, e-mail: P.Hendriks@rug.nl

Received: date / Revised version: date

Abstract This article describes the generation of aggregated and elliptic sentences, using Dependency Trees connected by rhetorical relations as input. The system we have developed can generate both hypotactic and paratactic constructions with appropriate cue words, and various forms of ellipsis such as Gapping and Conjunction Reduction. We contend that Dependency Trees connected by rhetorical relations are excellent input for a generation system that has to generate ellipsis, and we propose a taxonomy of the most common Dutch cue words, grouped according to the kind of discourse relations they signal. Finally, we argue that syntactic aggregation should be performed in the Surface Realizer of a language generation system, because it requires access to language-specific syntactic information.

Key words aggregation, dependency trees, discourse structure, ellipsis, language generation

1 Introduction

Ellipsis and coordination are key features of natural language. For a Natural Language Generation (NLG) system to produce fluent, coherent texts, it must be able to generate coordinated and elliptic sentences. The generation of such sentences is part of a process called aggregation, which is one of the basic tasks of any NLG system (Reiter and Dale, 2000). However, there is no consensus on the definition of aggregation. It is an amalgam of processes

* Feikje Hielkema carried out this work while she was at the University of Groningen.

that can be performed in different modules of a language generation system (Cahill and Reape, 1999). Aggregation has been broadly defined as the combination of two or more linguistic structures into one linguistic structure by Reape and Mellish (1999), who distinguish several kinds of aggregation: conceptual, discourse, syntactic, semantic, lexical and referential aggregation. Shaw (2002) distinguishes four different types: interpretative, referential, lexical and syntactic aggregation. Dalianis (1999) also distinguishes four types: syntactic aggregation, elision, lexical and referential aggregation.

In this paper we focus on syntactic aggregation, which is the most common form of aggregation according to Reape and Mellish (1999). Shaw (2002) defines syntactic aggregation as the combination of propositions using syntactic constructions, with hypotactic and paratactic aggregation as the two main types. Dalianis (1999) equals syntactic aggregation with removing redundant information at the syntactic level, while leaving at least one item in the text to carry the meaning explicitly. We define syntactic aggregation as the process of combining two clauses using any kind of syntactic structure such as coordination, subordination, and also the absence of overt syntactic structure in the case of ellipsis.

We describe how syntactic aggregation is implemented in a story generation system called the Virtual Storyteller. The focus lies on the generation of coordinated and elliptic structures for the Dutch language. Although our work aims in the first place at improving the texts produced by our story generation system, we believe that our approach to syntactic aggregation and ellipsis is sufficiently general to be relevant for all kinds of language generation systems. New aspects of our approach, compared to earlier work on aggregation and ellipsis, are our use of Dependency Trees (Mel'cuk, 1988) in combination with rhetorical relations, and the fact that we perform aggregation in the Surface Realizer rather than in the Microplanner module of our NLG system. Another contribution is our design of a taxonomy of the most common Dutch cue words[1] for use in the aggregation process.

This paper is structured as follows. In Section 2 we briefly describe the Virtual Storyteller system, which forms the application context for our work. The goals we set out to achieve are discussed in Section 3. In Section 4 we describe the design of our language generation module (the Narrator), and argue that syntactic aggregation should be located within the last stage of the language generation process. In Section 5, we present the cue word taxonomy that we developed for use in the aggregation process. In Section 6, we discuss how we perform aggregation and ellipsis in our system, using this taxonomy. In Section 7 we show some results, and we end with a brief discussion and conclusions in Sections 8 and 9.

2 Background: The Virtual Storyteller

The Virtual Storyteller is a system that automatically creates fairy tales, expressed in Dutch natural language. Story generation in the Virtual Storyteller takes place in three stages, each handled by specialized components.

The first stage is *plot generation*, which is based on the actions of semi-autonomous character agents in a virtual story world. These character agents are able to reason logically and can make plans to achieve their personal goals. In addition, they are equipped with an emotion model. In reaction to events and objects, the characters can experience emotions such as joy and distress, love and hate, and their subsequent actions are influenced by these emotions (Theune et al., 2004). What happens in the story world is recorded in a structure called the *fabula*, a causal network that expresses the relations between character actions and the goals and emotions that motivate them. The fabula also records temporal information.

The second stage is *narration*, the generation of a text that expresses the plot in natural language (in our case, Dutch). Originally, narration in the Virtual Storyteller involved a simple mapping of character actions to fixed sentence templates. To improve on this, we have developed a sophisticated Narrator component, which takes a fabula structure as input and converts it into a natural language text using knowledge about discourse structure and Dutch syntax and morphology. In the rest of this paper, the Narrator will be discussed in more detail, focusing on the way aggregation is handled. For information on other aspects of the Narrator, we refer to Slabbers (2006).

The third generation stage is *presentation*: presenting the text generated by the Narrator by means of speech synthesis. To make the speech produced by standard text-to-speech systems more suitable for storytelling, we have developed a method for converting 'neutral' speech into storytelling speech by adapting the prosody (Theune et al., 2006).

3 Goal

So far, much of our work on the Virtual Storyteller system has focused on plot creation and the development of believable characters. However, the quality of a story depends not only on the actions and emotions of the characters, but also on how these are expressed in natural language. However high the quality of the plot may be, if it is badly expressed the result will not be an enjoyable story. This can be illustrated by the text fragment given below, which is an example of the kind of texts originally generated by the Virtual Storyteller system. In this fragment, given together with its translation, two character agents accidentally meet in the desert (a location in our virtual story world).

Diana gaat naar de woestijn.	*Diana goes to the desert.*
Brutus gaat naar de woestijn.	*Brutus goes to the desert.*
Diana is bang voor Brutus.	*Diana is afraid of Brutus.*
Diana gaat naar het bos.	*Diana goes to the forest.*
Brutus gaat naar het bos.	*Brutus goes to the forest.*

As can be seen in this example, presenting only the bare facts of the story using fixed sentences results in a monotone, uninteresting narrative. Some

obvious improvements would be the use of pronouns, more suitable word choice (e.g., having Diana *flee* rather than *go* to the forest) and variation in the length and complexity of sentences. In this paper, we focus on the last aspect: the aggregation of simple sentences into more complex ones.

Experimental research by Callaway and Lester (2001) has indicated that aggregation (called revision in their system) is an important aspect of the generation of narrative prose. When presented with different versions of the same story, their test subjects showed a clear preference for narratives on which revision had been performed, over narratives without revision. Therefore we believe that having our Narrator perform syntactic aggregation should help enormously to improve the liveliness of the generated narratives. Our goal in designing an aggregation component for the Narrator has been to allow for the production of at least the following structures:

- **Paratactic constructions**: these are constructions where two clauses of equal status are coordinated. Example: 'Diana verliet de woestijn en Brutus betrad het bos' (*Diana left the desert and Brutus entered the forest*)

- **Hypotactic constructions**: these are constructions where one of the combined clauses is subordinated to the other. Example: 'Diana verliet de woestijn, omdat ze Brutus zag' (*Diana left the desert, because she saw Brutus*)[2]

- **Conjunction Reduction**: these are paratactic, elliptic constructions where the subject of the second clause is deleted. They only occur with the cue words 'en' (*and*) and 'maar' (*but*). Example: 'Diana betrad de woestijn en zag Brutus' (*Diana entered the desert and saw Brutus*)

- **Right Node Raising**: these are paratactic, elliptic constructions where the rightmost string of the first clause is deleted. The ellipted string can be a direct object, but it can also be a locative, or any other string, as long as it is in the rightmost position of the first and second clause. Example: 'Diana schopt en de prins slaat Brutus' (*Diana kicks and the prince hits Brutus*)

- **Gapping**: these are paratactic, elliptic constructions where the main verb of the second clause is deleted. In the following example, this is the verb 'verliet' (*left*). Example: 'Diana verliet de woestijn en Brutus het bos' (*Diana left the desert and Brutus the forest*)

- **Stripping**: these are paratactic, elliptic constructions where all constituents but one are deleted from the second clause, and replaced by the word 'ook' (*too*). This can happen with any constituent. Example: 'Diana betrad de woestijn en Brutus ook' (*Diana entered the desert and so did Brutus*)[3]

All possible combinations of these structures should be generated as well, for example sentences such as 'Diana gaf Brutus een schop en de prins een kus' (*Diana gave Brutus a kick and the prince a kiss*), which is both gapped and conjunction-reduced, as both verb and subject are deleted in the second conjunct. We would also like to be able to coordinate single constituents, e.g., 'Diana schopte en vervloekte Brutus' (*Diana kicked and cursed Brutus*). Different cue words should be available to express different rhetorical relations between the clauses. If more than one cue word is available for each relation, this results in variety in the output.

Note that several linguists have proposed a unified analysis of various types of reduced coordinate structures (Tai, 1969; van Oirsouw, 1987; Hartmann, 2000); see Shaw (1998, 2002) for an NLG framework based on these insights. However, in this paper we analyze Conjunction Reduction, Right Node Raising and Gapping as related but different types of syntactic constructions that can be combined with each other. (See Harbusch and Kempen (2006) for a similar approach.) This is motivated by the frequently made observation that Right Node Raising is insensitive to syntactic islands and does not respect syntactic constituency (see Zwarts (1986), for Dutch), whereas Gapping is sensitive to syntactic islands and does respect syntactic constituency (see Neijt (1979), for Dutch). For this reason, it does not seem wise to collapse them.

4 Design of the Narrator

In this section we present the design of the Narrator component of the Virtual Storyteller. First we provide a global description of the Narrator architecture. Then we discuss the placement of the aggregation component in this architecture. Finally we zoom in on Rhetorical Dependency Graphs, the linguistic representation we use as a basis for aggregation and ellipsis.

4.1 Architecture

The design of the Narrator is based on the pipe-lined NLG architecture described by Reiter and Dale (2000), who distinguish three stages in the NLG process:

- **Document planning**: this involves determining the content and the global structure of the information to be presented. The outcome is an abstract message specification.
- **Microplanning**: at this stage, the message specification is fleshed out further. This involves the generation of referring expressions, lexicalization (word choice), and aggregation.
- **Surface realization**: here, the abstract message specification is converted into real text, using knowledge about syntax, morphology, etc. In addition, mark-up may be added for use by external components.

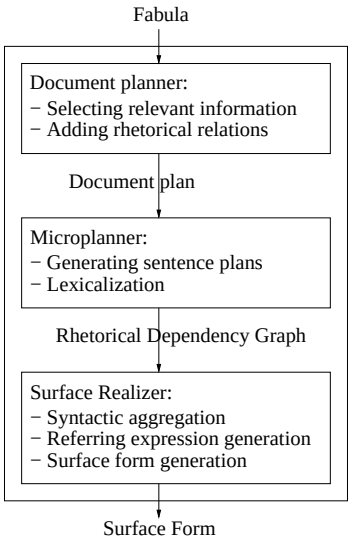

Fig. 1 Architecture of the Narrator

The Narrator consists of three modules that correspond to these three stages: a Document Planner, a Microplanner and a Surface Realizer. The input for the Narrator consists of a plot representation (the fabula network) specifying the actions of each of the characters, their emotions and goals, linked together by causal relations. The Document Planner turns this into a document plan by mapping the links from the network to appropriate rhetorical relations, while removing irrelevant information and adding background information when necessary. For instance, the Document Planner may add information about the properties of characters and objects that play a role in the story. These are added to the document plan using an Elaboration relation.

The Microplanner then converts the document plan into what we call a *Rhetorical Dependency Graph* (see Section 4.3) by mapping the plot elements at its leaf nodes to Dependency Trees, while maintaining the rhetorical relations between them. We call this process *sentence plan generation*. Currently this is done in a fairly simple fashion, using templates for Dependency Trees expressing various actions, events and states. The Microplanner also performs lexicalization, mapping the concepts from the document plan to Dutch words. These words are still uninflected, as morphology is taken care of in the Surface Realizer. References to characters and objects are not lexicalized yet at this stage, because this is part of referring expression generation, which is also performed in the Surface Realizer.

Finally, given a Rhetorical Dependency Graph as input, the Surface Realizer decides which of the Dependency Trees making up its leaf nodes should be aggregated, which cue words should be used to express the relations between the trees, and whether ellipsis should be applied to the aggregated

trees (and if so, which type of ellipsis). How this is done, is discussed in detail in Section 6. As the next step, referring expressions are generated for the remaining (non-ellipted) references to characters and objects, e.g., using a proper name, a descriptive noun phrase, or a pronoun, depending on factors such as recency of mention and the presence of alternative antecedents. Finally, the words in the (possibly aggregated) Dependency Trees are inflected and ordered, obeying rules that dictate the order of the child nodes, using their dependency labels. Punctuation is added when the Surface Form is complete. The global architecture of the Narrator is shown in Figure 1.

4.2 Where to perform aggregation

As can be noted from the above description, the architecture of the Narrator deviates from the NLG architecture presented by Reiter and Dale (2000) with respect to the location of two important NLG tasks: aggregation and the generation of referring expressions, which we have placed in the Surface Realizer rather than the Microplanner module of our system.

Despite its deviation from the 'standard', our decision to perform aggregation in the Surface Realizer is not unprecedented. The RAGS-project (Cahill and Reape, 1999), in which the architecture of nineteen NLG systems was investigated, showed a lack of consensus on the location of the aggregation process in the NLG pipe-line. Instead, the situation varied widely over different NLG-systems, with a number of systems performing aggregation in the Surface Realizer. This divergence is partly caused by the fact that many, quite different processes are gathered under aggregation (see Reape and Mellish (1999) for an overview). However, the Narrator only deals with syntactic aggregation. Because syntactic aggregation involves grammatical processes (coordinating sentences and deciding which elements can be deleted without rendering the sentence ungrammatical), in our view it should be situated in the Surface Realizer together with the other grammatical processes. For example, one of the structures we focus on (viz. Gapping) is sensitive to syntactic island restrictions such as the Complex NP Constraint, respects syntactic constituency, and must target the main verb (Neijt, 1979). To decide whether Gapping is possible or not, we therefore need access to syntactic information. The same holds for Right Node Raising, which applies to the rightmost material in a clause and therefore requires information about linear word order.

In short, we perform aggregation in the Surface Realizer module of our system because the syntactic information it should have access to, is available there. A consequence of this decision is that referring expression generation also has to be performed in the Surface Realizer (after aggregation), to avoid generating referring expressions that will be deleted during ellipsis. Still, while being placed in the same module and (potentially) making use of the same syntactic resources, aggregation and the generation of referring expressions are handled separately from the actual surface form generation that, naturally, also takes place in the Surface Realizer.

4.3 Rhetorical Dependency Graphs

The Surface Realizer receives as input what we call a Rhetorical Dependency Graph: a graph with Dependency Trees expressing simple propositions as nodes, connected by rhetorical relations.

Dependency Trees are a prominent feature of Meaning-Text theory (Mel'cuk, 1988). They are constructed on the basis of predicates and arguments. There is no dependence on linear word order, and no limit on the number of children a node can have. This means the trees are able to handle variation in word order easily, so that they translate well over different languages. In fact, Dependency Trees have been used with success in Machine Translation (Lavoie et al., 2000). The fact that Dependency Trees are largely language independent means that a generation system using Dependency Trees can be adjusted to another language quite easily; only the rules specific to the generated language have to be replaced, the generation algorithm remains the same. In addition, the independence of word order, and the dependency labels that specify which role a node performs in its parent syntactic category, make Dependency Trees easy to manipulate, especially for the purpose of generating ellipsis. These two properties make Dependency Trees an attractive formalism for representing sentence plans in the Narrator.

A standard of Dependency Trees is set by the Spoken Dutch Corpus (van der Wouden et al., 2002). The Alpino parser (Bouma et al., 2001), a computational analyzer for Dutch, follows this standard, albeit with some practical adaptations. In the Narrator we adopt the Alpino format for Dependency Trees with some minor changes: a tag for morphology has been added, and the tags that indicate the position of a word or constituent in the Surface Form are left out initially (these are added during linearization). An example Dependency Tree for the sentence 'Diana vlucht naar de bergen' (*Diana flees to the mountains*) is given in Figure 2.

Dependency Trees expressing basic plot elements (i.e., simple facts about the current state of the story world) are connected by *rhetorical relations* to form a so-called Rhetorical Dependency Graph. The use of such graphs was inspired by Rhetorical Structure Theory (RST) (Mann and Thompson, 1987). This theory was originally developed as a descriptive framework for the analysis of text structure, but it has also been used in several NLG applications (for an overview, see Hovy (1993)). In RST, rhetorical relations have been defined independently from the lexical and grammatical forms of a text. Thus it can be argued that these rhetorical relations are language-independent, as is illustrated by the fact that RST has been applied successfully to various languages including English, Dutch, Portuguese and Japanese (see Taboada and Mann (2006) for an overview). However, rhetorical relations do have an influence on syntactic structure and lexical choice. Scott and de Souza (1990) describe the relation between rhetorical relations and syntactic structure. They propose a number of heuristics to guide the generation process, such as "Embedding can only be applied to the ELABORATION relation" (Scott and de Souza (1990), p.57) and "The parat-

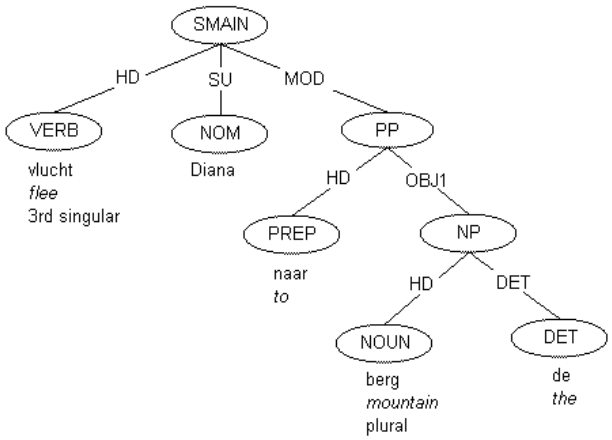

Fig. 2 Dependency Tree for 'Diana vlucht naar de bergen' (*Diana flees to the mountains*)

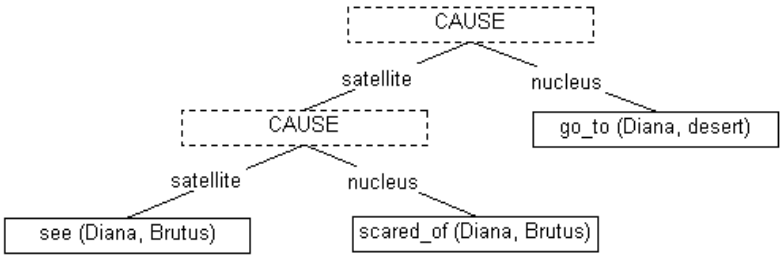

Fig. 3 Example Rhetorical Dependency Graph. (The internal structure of the Dependency Trees is not shown.)

actic marker *and* must only be applied to SEQUENCE and LIST, *but* to CONTRAST, and *or* to ALTERNATIVE" (Scott and de Souza (1990), p.67). Shaw (2002) points out the relation between rhetorical relations and hypotactic constructions: using rhetorical relations, one can decide which hypotactic structure is appropriate to use. According to Hendriks (2004), rhetorical relations also influence the use of certain elliptic structures, such as Gapping: a gapped sentence cannot have a Causal relation between its clauses, but only a Resemblance relation, such as Additive or Contrast. All this means that we need information about rhetorical relations as input for the aggregation process, to be used for the selection of a suitable syntactic, possibly elliptic, structure when combining two Dependency Trees. (However, see White (2006) for a sophisticated method to generate elliptic structures without making use of rhetorical relations.)

The basic set of rhetorical relations that we currently use in the Narrator consists of Cause, Contrast, Temporal and Additive relations (see Section 5.2). These relations are based on, but not identical to, those from Rhetorical Structure Theory. RST distinguishes far more, and far more spe-

cific, rhetorical relations, most of which are simply too detailed for the relatively simple stories created by the Virtual Storyteller. Moreover, some of the distinctions made in RST are less useful from the perspective of the Narrator. In text analysis, Cause and Consequence can be seen as two different relations, distinguished by the roles of the nucleus and the satellite. If the nucleus (the important clause, which the satellite supports) expresses the cause and the satellite the consequence, we see a Consequence relation. Alternatively, if the satellite is the cause, we see a Cause relation. The Virtual Storyteller makes no such distinction. The Document Planner first determines that two plot elements are important enough to mention, and then that there is a cause-consequence relation between the two. Because it has no way to distinguish between a Cause and a Consequence relation, and because in practice these two relations can be realized at the surface level in the same way, we have only defined a Cause relation, in which the satellite is always the cause and the nucleus the consequence. In temporal relations, the satellite always corresponds to the event that is first in chronological order. So unlike RST, where the nucleus/satellite distinction indicates relative importance, in our Rhetorical Dependency Graphs we use the distinction to indicate the different roles of the elements in the relation. This is illustrated by the Rhetorical Dependency Graph shown in Figure 3, which shows that Diana's seeing Brutus causes her to be afraid, and the combination of the two events causes Diana to go (flee) to the desert.[4] Another difference with RST is that (for practical reasons) we do not have multi-nuclear relations. Instead, in relations that are considered multi-nuclear in RST, such as Contrast, a nucleus and a satellite are arbitrarily assigned.

5 Cue word taxonomy

Cue words are a natural language's resources to signal different rhetorical relations, and as such are a vital part of syntactic aggregation. They have great influence on the syntactic structure of an aggregated sentence. Sanders and Noordman (2000) show that rhetorical relations (which they call coherence relations) play an important part in human text processing, and that cue words cause faster processing of coherence relations between two segments.

Dalianis (1999) describes a system in which the aggregation and the cue word system are interleaved. Where aggregation has taken place, the clauses involved are marked with a cue primitive (e.g., joint or disjunct) to disambiguate the aggregated sentence. During surface generation the cue primitives are then translated into cue words. In our system, cue words are selected based on the rhetorical relation between two Dependency Trees, and the selected cue word determines if and how the trees are aggregated. In this section we describe the taxonomy of Dutch cue words we have developed for this purpose. First we discuss related work on cue word taxonomies for English and Dutch.

5.1 Taxonomies for English and Dutch cue words

Knott and Dale (1994) collected a corpus of English cue words and classified them according to their function in discourse, using a substitutability test. Put simply, this test is used to determine whether two cue words signal (partly) the same features, by checking whether one can be substituted by the other in a particular context. For instance,

Kate and Sam are like	whereas	Kate is only
chalk and cheese. Sam	+ on the other hand	interested in
lives for his books;	* then again	martial arts.
I don't know where to	then again,	we had curry
eat tonight; The Star of	+ on the other hand	just the
India is always good;	* whereas	other night.

(examples are taken from Knott and Dale (1996)). A plus sign means that a cue word is a suitable substitute, whereas an asterisk means it is not. The example shows that *on the other hand* signals only those features that *whereas* and *then again* have in common, while *whereas* and *then again* signal opposing features. On this basis Knott and Dale created a taxonomy of English cue words. This taxonomy is hierarchical, as some cue words signal more features than others.

Following the same method, Knott and Sanders (1998) created a similar taxonomy for Dutch cue words, using the cognitive primitives that were proposed by Sanders et al. (1992) to differentiate between the classes. A drawback of this taxonomy is that it is rather complex, and will presumably be hard to implement in a practical NLG system. In addition, Knott and Sanders admit that their taxonomy was created using only those cue words that were easiest to classify; other cue words will be even harder to classify and may cause the taxonomy to be even more complex. For these reasons we have decided to create a less convoluted taxonomy for our own purposes (see Figure 4). This taxonomy is presented in the next section.

5.2 A cue word taxonomy for syntactic aggregation

For the purpose of syntactic aggregation in our storytelling system, a small taxonomy charting only the most prevalent cue words in Dutch has been constructed, using a variant of the substitutability test described by Knott and Dale (1994). Because our taxonomy is meant to be used *before* the words in a Dependency Tree are ordered to produce the surface form (linearization), unlike Knott and Dale we paid no attention to any changes a cue word might make in the word order in the clauses. We also allowed substitutions that changed the clause order, as long as this did not influence the meaning of the sentence. In short, we only looked at substitutability with respect to meaning, regardless of surface form.

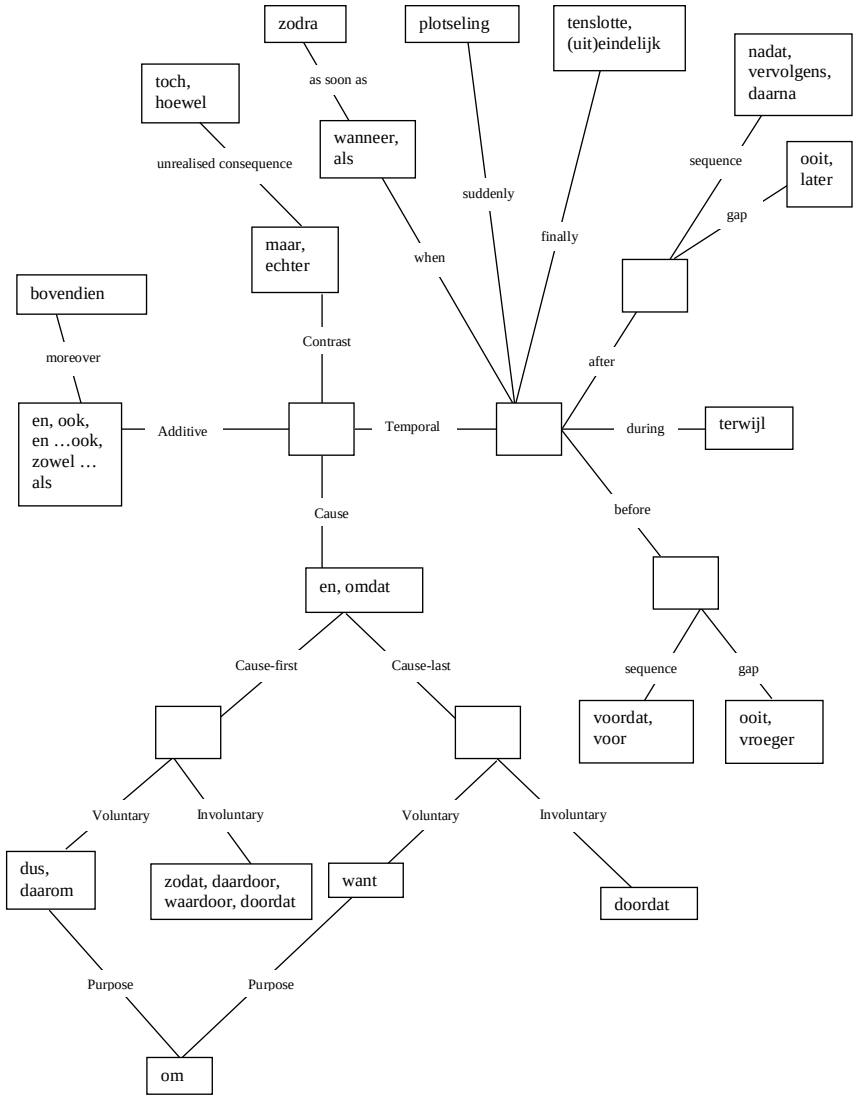

Fig. 4 Taxonomy of Dutch cue words

The test was used on two types of data: sentences taken from a fairy tale book (Andersen, 1975), and sentences based on the original output of the Virtual Storyteller (see Section 3). The tested cue words all had a frequency of >100 in a representative sample from the Spoken Dutch Corpus (van der Wouden et al., 2002), to exclude rare cue words. Only cue words

that seemed appropriate for narrating a story (not too difficult, because the obvious target group are children) were included.

The set of cue words is divided into four main classes: cue words signaling Cause, Temporal, Contrast and Additive relations. Each of these classes has been subdivided into smaller subsets to allow for finer distinctions in meaning, to be discussed below. A cue word in a small subset can always be replaced by a more general cue word, i.e., a cue word included in a superset from the same category. The resulting taxonomy is given in Figure 4. All features used in the taxonomy are in principle available to the story generation system. For instance, temporal information can be retrieved from the fabula structure. Empty boxes indicate 'missing cue words', i.e., no cue words were found to express a relation with the indicated properties.

For Cause relations, one feature denotes whether the cause is expressed in the first or second clause in a complex sentence. A further distinction is whether the consequence was voluntary (consciously willed by an actor) or not. The cue words expressing a Purpose relation turned out to be the same as the cue words signaling a voluntary Cause relation, with one addition: 'om' (*to*, as in *Diana picked up the sword to kill Brutus*).

The cue words signaling a Temporal relation are divided into subsets according to the order of events expressed by the first and second clause: before, after, sequence, finally, during, suddenly and once (for the specific storytelling expression 'Er was eens' (*Once upon a time*)).

In the Contrast relation, there is one subset named 'unrealized consequence'. This subset encompasses the cases where one clause expresses something that is the direct opposite of what you would expect it to be, based on the 'default' consequence of what is expressed in the other clause. For instance in *Although Diana was scared, she did not flee* we would have expected the second clause to express the consequence of the first clause (e.g., *Diana was scared, so she fled*).

The most general Dutch cue word, which seems to be appropriate in the case of underlying Causal, Additive, Temporal and (in some contexts) Contrast relations is 'en' (*and*).[5] In principle, it can be substituted for any other cue word, so it might be argued that this cue word should occupy the empty box in the middle of our taxonomy. However, because this cue word strongly suggests an Additive relation and leaves any other relations very much implicit, we decided not to use it in this general fashion in our system, as we prefer a more explicit signaling of the rhetorical relations in the generated stories.

6 Performing aggregation and ellipsis

The cue word taxonomy is used in the Surface Realizer during the syntactic aggregation process, which consists of three steps. First, based on the rhetorical relation between two Dependency Trees, an appropriate cue word is selected that expresses this relation. Then, depending on the properties

of the selected cue word, the two Dependency Trees may be joined together using a specific syntactic construction. Finally, the joined Dependency Trees are checked for repeated elements that can be ellipted. Below we go through each of these steps in more detail, but first we briefly discuss how pairs of Dependency Trees are selected for aggregation.

6.1 Selecting Dependency Trees for aggregation

The Rhetorical Dependency Graph representing a story structure has the form of a binary branching tree, with Dependency Trees representing simple propositions as its leaves and relations as its non-terminal nodes. See Figure 3 for a very simple example; a full story will have a much larger Graph. The syntactic aggregation algorithm goes through the Rhetorical Dependency Graph depth-first, looking for relations that have two Dependency Trees as their children. If it finds one, such as that shown in Figure 5, it is passed through the aggregation steps described below to transform it, if possible, into a complex Dependency Tree combining its nucleus and satellite. If aggregation succeeds, the Rhetorical Dependency Graph is updated with the new, complex Dependency Tree replacing the original relation, and the algorithm continues looking for relations to transform.

To keep the resulting sentences from getting too complex, at most three simple Dependency Trees can be combined. In cases where this restriction prohibits aggregation it may still be possible to express the relation between two Dependency Trees by adding an adverb such as *then*, *however*, etc. to the second tree. To express the maximum of relations, after the initial traversal and transformation of the Rhetorical Dependency Graph, the algorithm makes another pass through it and expresses some final relations by adding adverbs to non-aggregated sentences.

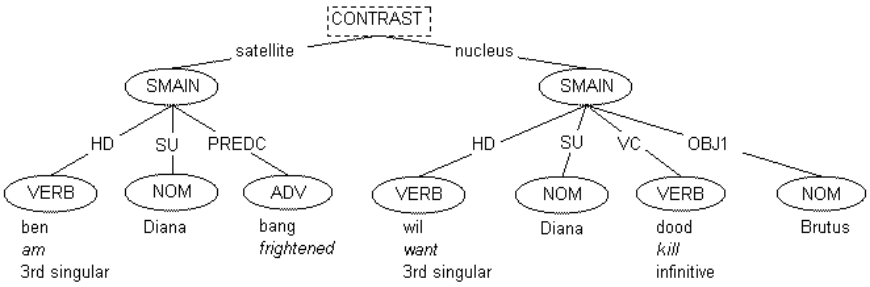

Fig. 5 Example input for aggregation.

6.2 Cue word selection

When two Dependency Trees are selected for aggregation, the first step is to choose an appropriate cue word to express their relation. A relation can have features that correspond to certain subclasses in the cue word taxonomy from Figure 4. If the values for these features are known, a highly specific cue word can be selected. If the relation has no specific features, a more general cue word is chosen. It is not necessarily the most specific applicable cue word that gets selected; discourse history plays a part as well: if a cue word has been recently used, it is less likely to get chosen again. The cue word selection algorithm first tries to find an appropriate coordinating cue word, i.e., a cue word that combines two trees. If coordinating cue words are unavailable or disallowed, an adverb is selected and added to either the nucleus or the satellite, without aggregation. Coordinating cue words are disallowed if the resulting aggregated tree would get too complex (see Section 6.1). When a cue word is selected, a node is created for it in the tree of the relation and the result is passed on to the next step, where either the dependency trees are combined, or the cue word is added to one of them.

As an example, assume that the Contrast relation from Figure 5 is selected for aggregation. Its satellite is a Dependency Tree expressing that Diana is scared and its nucleus a Dependency Tree expressing that Diana wants to kill Brutus. The relation is not specified for the 'unrealized-consequence' feature (see Figure 4) so the choice of cue words is limited to 'maar' (*but*) and 'echter' (*however*). Of these, 'maar' is a coordinator, and 'echter' is an adverb. Since both nucleus and satellite of the relation are simple Dependency Trees there is no reason to prefer a non-coordinated construction, and we assume that 'maar' gets selected.[6]

6.3 Aggregation

The next step is to create a grammatical, aggregated Dependency Tree from the nucleus and satellite of the relation, given the selected cue word. Only entire trees are combined at this stage; the conjunction of single constituents (i.e., nodes in the Dependency Trees) is handled in the next step, Ellipsis. The structure of the aggregated Dependency Tree depends on the properties of the selected cue word. If the cue word is a coordinator, a paratactic structure is created. A new Dependency Tree is constructed with a root labeled CONJ (conjunction). Its child nodes are a coordinator (the cue word) and two conjuncts (the nucleus and satellite of the old relation). Other than in Alpino-trees, the conjuncts do not have the same label: the distinction between nucleus and satellite is kept, because in some relations this is important for linearization (for instance, in a paratactic structure expressing a causal relation, we do not want to put the consequence before the cause). If the selected cue word is a subordinator, a hypotactic structure is created. If the cue word is an adverb, the cue word is added to either the nucleus or the satellite of the relation (depending on the cue word), and the separate trees

are added to the result, the unmodified tree first. In our example, the cue word 'maar' is a coordinator, so it requires the creation of a paratactic tree. A new Dependency Tree is built, with as its root node a Conjunction. The nucleus and satellite of the tree become its conjuncts and its coordinator is 'maar'. This new tree is then passed on to the next stage, Ellipsis.

6.4 Ellipsis

The final step is to perform ellipsis, i.e., to remove superfluous nodes or branches from a Dependency Tree. This step only applies to paratactic constructions. First both conjuncts of the aggregated Dependency Tree are checked for identical nodes or branches. This search is restricted to constituents and direct children of the root of the Dependency Tree. Like Dalianis (1999) we use unique identifiers to distinguish different instances of the same concept, so that ellipsis is only applied to nodes with identical referents. If the conjunction is nested, the nodes that the nested conjuncts share are compared to the nodes in the other conjunct. Any identical nodes found in the Dependency Tree are marked. When all identical nodes have been found and marked, it is determined which operations are suitable.

If no nodes are marked at all, no Ellipsis will take place. If one or more nodes are marked (but more than one node has been left unmarked), the labels of the nodes that are identical determine which operation is performed: Gapping (where the verb of the second conjunct is removed), Right Node Raising (removal of the rightmost part of the first conjunct) or Conjunction Reduction (removal of the subject of the second conjunct). These three operations can be performed in combination. For example, if only the subjects of the coordinated clauses are marked as identical, Conjunction Reduction is selected, but when both the subjects and the main verbs are marked, then both Conjunction Reduction and Gapping are performed. An example would be the sentence *Diana is in love with the prince and afraid of Brutus*, formed by aggregating the simple sentences *Diana is in love with the prince* and *Diana is afraid of Brutus* and applying both Conjunction Reduction and Gapping. Right Node Raising and Gapping cannot be applied in all cases: they can only be used to express additive or contrast relations, because these constructions rule out a causal interpretation (Hendriks, 2004). Conjunction Reduction is feasible with all relations. Right Node Raising can be applied to multiple constituents. For example, in a sentence such as *Diana kicks and the prince hits Brutus in the desert* Right Node Raising has applied to both *Brutus* and *in the desert*.

If all nodes are marked except one, as for instance in the tree representing 'Diana gaat naar het bos en Brutus gaat naar het bos' (*Diana goes to the forest and Brutus goes to the forest*), where both conjuncts are identical except for the subject, the transformation is more radical. Two constructions can be chosen: 1) Stripping and 2) coordination of the unmarked nodes. With Stripping, all the identical nodes of the nucleus are deleted and replaced by a node representing the discourse particle 'ook' (*too*), to realize

a construction such as 'Diana gaat naar het bos en Brutus ook' (lit. *Diana goes to the forest and Brutus too*). As an alternative, in Constituent Coordination the non-identical nodes are combined into one, and the second conjunct is deleted in its entirety, thus realizing constructions such as 'Diana en Brutus gaan naar het bos' (*Diana and Brutus go to the forest*).

The conjunct that is ellipted is marked CNJ-ELLIPTED, unless the operation was Right Node Raising. In that case the conjunct is marked CNJ-RAISED, because with Right Node Raising something is deleted from the first conjunct of the Surface Form, while with the other operations a node from the second conjunct is removed. So different labels are needed to determine which conjunct comes first in linearization.[7] Superfluous nodes are deleted, but their parent node receives a connection to the remaining twin. This connection is marked 'borrowed' to show that the node should not appear in the Surface Form. This means that the elliptic conjunct has the same structure as the intact conjunct, but is ellipted in the Surface Form.

In our example, the Dependency Tree created by the Surface Realizer was paratactic. The conjuncts have one node in common, i.e., their subject, *Diana*. When only the subject is identical, Conjunction Reduction is the suitable elliptic structure. The subject node of the second conjunct (the nucleus) is removed, and replaced by a connection with the subject of the first conjunct (the satellite). The result is shown in Figure 6. This Dependency Tree could be realized as 'Diana was bang, maar wilde Brutus doden' (*Diana was scared, but wanted to kill Brutus*). This concludes our description of how aggregation and ellipsis are performed in the Virtual Storyteller; more details can be found in Hielkema (2005).

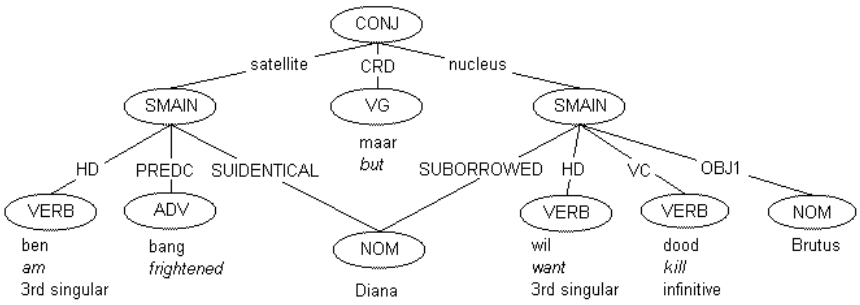

Fig. 6 Output of the Elliptor.

7 Results

In this section we take a look at the results that are actually produced by the syntactic aggregation algorithm. Because a reimplementation of the plot creation component of the Virtual Storyteller is being carried out, the Narrator currently has no fabula input available (see Section 2). Therefore,

the algorithm has been tested with hand-made document plans. First, let us consider the original output of the Narrator as presented in Section 3. This text is repeated in Figure 7 as text (1). The original version of the Narrator made no use of rhetorical relations, but for the sake of comparison let us assume that Causal relations hold between the facts expressed in sentences 3-5, and Additive relations between the rest. Given these relations, the new version of the Narrator (using aggregation, ellipsis and pronominalization) would express the story fragment as text (2) in Figure 7.

(1)	1	Diana gaat naar de woestijn	*Diana goes to the desert.*
	2	Brutus gaat naar de woestijn.	*Brutus goes to the desert.*
	3	Diana is bang voor Brutus.	*Diana is afraid of Brutus.*
	4	Diana gaat naar het bos.	*Diana goes to the forest.*
	5	Brutus gaat naar het bos.	*Brutus goes to the forest.*
(2)	1+2	Diana en Brutus gaan naar de woestijn.	*Diana and Brutus go to the desert.*
	3+4	Diana gaat naar het bos, want ze is bang voor Brutus.	*Diana goes to the forest, because she is afraid of Brutus.*
	5	Daarom gaat Brutus ook naar het bos.	*Therefore Brutus goes to the forest too.*

Fig. 7 Old and new version of Narrator output

We see that in text (2), the first two sentences of text (1) are combined using Constituent Coordination, that the third and fourth sentence are coordinated using the causal cue word 'want' (*because*) and that the causal adverb 'daarom' (*therefore*) has been added to the last sentence. It is clear that text (2) flows more smoothly and appears more coherent than text (1). However, the coherence of text (2) is still not optimal. This is not a problem of the aggregation algorithm, but of the limited input to the algorithm: the Document Plan, based on the original example, does not include all the information that is relevant for the story. In particular, apart from emotional states, information about the motivations of the characters is lacking. For instance, in text (2) it is still not clear why Brutus goes to the forest too. (In fact he is following Diana, because his goal is to capture her.) This kind of information will be present in the new fabula structures to be used as input for the Narrator. If we add information about the personal goals of the characters to the input document plan, we get text (3) in Figure 8. Note that in this version of the example text we use past tense, because it is more appropriate for fairy tales. The morphology component of the Narrator allows for the use of different tenses, but originally, when the Virtual Storyteller still used fixed sentences for language generation, everything was expressed in present tense. Therefore, in text (2) we used present tense to keep it similar to text (1).

(3)
Diana ging naar de woestijn en Brutus ging ook naar de woestijn, want hij wilde de omgeving verkennen. Daardoor zag Diana Brutus. Diana wilde Brutus vermoorden, maar was bang voor hem, dus zij wilde vluchten. Daarom ging zij naar het bos. Brutus ging ook naar het bos, want hij wilde Diana gevangen nemen.

Diana went to the desert and Brutus went to the desert too, because he wanted to explore the environment. Because of that, Diana saw Brutus. Diana wanted to kill Brutus, but was afraid of him, so she wanted to flee. Therefore she went to the forest. Brutus went to the forest too, because he wanted to capture Diana.

Fig. 8 Example output of the Narrator with character goals added.

Text (3) expresses why Brutus went to the desert, how this caused Diana to see him (an Involuntary Cause relation), why Diana went to the forest and why Brutus followed her there. Still, some flaws can be observed in the way the story is expressed. For instance, in the first and last sentences the actions of Brutus are similar to Diana's, and the adverb 'ook' (*too*) is used to express this parallelism, but not in connection with Stripping. Instead, the Narrator has added this adverb to the Dependency Trees prior to aggregation, based on the semantic similarity with the previous plot element being expressed. However, in the first sentence of text (3) Stripping would have been more appropriate, so we would get 'Diana ging naar de woestijn en Brutus ook, want hij wilde de omgeving verkennen' (*Diana went to the desert, and so did Brutus, because he wanted to explore the environment*).[8] Currently, such a construction cannot be produced by the aggregation algorithm, because the example first expresses the causal relation between Brutus' action (going to the desert) and his goal (exploring the environment). The result is an aggregated Dependency Tree that is not eligible for Stripping when it is combined with the Dependency Tree that expresses Diana's action, because the two trees are only partly identical to each other. This means we need a more permissive definition of Stripping to allow it in cases like this.

Another problem of the current aggregation algorithm is illustrated by the first sentence of text (2): *Diana and Brutus go to the desert.* This sentence is ambiguous between a distributive reading, where Diana and Brutus went to the desert separately (which is correct in this case) and a collective reading, where they went together (which is incorrect). In real life, such ambiguous sentences are quite common, but in systems like the Virtual Storyteller it seems preferable to avoid ambiguity as much as possible. So when two Dependency Trees that are identical apart from one node (and thus are eligible for both Gapping and Constituent Coordination, see Section 6.4), express separate, non-simultaneous events, the elliptic construction that should be used is Stripping (*Diana went to the desert and so did Brutus*) rather than Constituent Coordination, which should only be used to express collective events. Adding such a restriction on the use of Constituent Coordination should be no problem, because the required

knowledge about Events is already available in the input of the Narrator. An even better solution would be to use additional cue words such as *both, each* or *separately* to disambiguate these cases, as in Dalianis (1999).

Our goal in designing the aggregation component of the Narrator was to generate at least the six different constructions listed in Section 3. Three of these constructions are present in the example texts: the first sentence of text (3) combines a paratactic and a hypotactic construction, and the third sentence has Conjunction Reduction as well as a combination of two paratactic constructions. Right Node Raising, Gapping and Stripping do not occur in the example texts, because the input Document Plans did not contain any instances to which these constructions would apply. However given applicable input, these elliptic constructions can be generated. For instance, if Brutus' initial action had been to go to the forest instead of the desert, the first sentence of text (3) would have been *Diana went to the desert and Brutus to the forest, because he wanted to explore the environment*, which is an instance of Gapping.

8 Discussion

In the design of the Narrator and the algorithm for aggregation and ellipsis we have made some important choices. Here we discuss the choices we made and add some critical remarks. We also discuss some of the remaining problems with our algorithm, and possible solutions.

8.1 Syntactic aggregation in the Surface Realizer

As we discussed above, there is no consensus on the proper location for aggregation to take place in the architecture of an NLG system. This is at least partly caused by the use of different definitions of aggregation, which comprise processes quite different in level. Our focus is on syntactic aggregation, which includes ellipsis. Since there is evidence that ellipsis is language dependent (not all forms of ellipsis are permissible or realized similarly in all languages) we have decided to situate aggregation in the Surface Realizer, the module which traditionally deals with language-specific grammatical processes. Here, all syntactic information that is relevant for ellipsis is available. For instance, Gapping is sensitive to syntactic islands, and Right Node Raising depends on linear word order, so it seems that the aggregation process must have access to this type of syntactic information. A consequence of locating aggregation in the Surface Realizer is that referring expression generation also has to be located there (after aggregation), to avoid generating referring expressions that will be deleted during ellipsis. A positive side-effect of this is that it allows easy access to syntactic information to check the binding constraints that hold for certain referring expressions.

It might be argued[9] that placing both aggregation and the generation of referring expressions in the Surface Realizer, in addition to its 'core' surface generation tasks, may make the module unnecessarily complicated and inefficient. However, apart from the (potential) sharing of syntactic resources, the various processes included in the Surface Realizer work quite independently from each other. The resulting architecture is as efficient as one where aggregation is situated in the Microplanner, but retrieves the syntactic information required for aggregation from the Surface Realizer.

Finally, it should be noted that although our argument for placing aggregation in the Surface Realizer rests on relevant syntactic information being available there, in practice, our aggregation algorithm does not make use of this information yet. Currently, the Dependency Trees to be aggregated only represent simple facts (*Diana is afraid, Diana flees, etc.*), so aggregation of these trees is relatively uncomplicated and does not require much syntactic knowledge. However, in the future we would like to use more advanced constructions, for which access to detailed syntactic information is indispensable.

8.2 Dependency Trees

Are Dependency Trees language independent? Mel'cuk (1988) designed Dependency Trees to be free of word order, to allow for languages where the word order is vastly different from English. But is it only word order that makes languages differ from one another? In languages such as Spanish or Italian (so-called pro-drop languages), it is not always necessary to mention the subject of a sentence - and that certainly shows in a Dependency Tree. And all languages have some concepts that do not translate well, though these might not crop up often in the telling of a simple fairy tale. Still, even if the Dependency Trees that the Surface Realizer gets as input are not completely language independent, the methods we use to process them and turn them into Surface Forms are. Substituting the cue words and grammatical rules should be sufficient to enable the Surface Realizer to process Dependency Trees lexicalized to a different language. For this reason alone, we think Dependency Trees are excellent input for a Surface Realizer that tries not to commit itself to one language. Another advantage of using Dependency Trees in syntactic aggregation is that they can easily be manipulated, because the role a word or a constituent plays in a sentence is given by a label (e.g., subject deletion is realized by deleting the node labeled 'subject').

8.3 Rhetorical relations and cue words

Rhetorical relations can be used to determine which cue words should be selected. They are a suitable mechanism to carry a certain meaning across to the level when it is finally of use. And since the relations even influence the

possibility or impossibility of ellipsis, the Surface Realizer certainly should have access to them.

However, the relations that were used in this project are not the set that was given by Mann and Thompson (1987). Only a few relations were selected for the moment: those deemed of the most importance to the narrating of a fairy tale. Cause and Contrast are very basic concepts, and Temporal relations are vital for any narrative. These relation classes were then divided into subclasses that correspond to groups of cue words, derived from the small cue word taxonomy that was created for this purpose. The properties that distinguish the subclasses are molded in terms of information that is available to the story generation system. This way, rhetorical relations can easily be added by the Document Planner, because the information is already there. The rhetorical relations that we currently distinguish were selected based on linguistic evidence, i.e., on the grouping of cue words that we established using the substitution test. In the future, our cue word grouping should be validated based on experiments with naive subjects, along the lines of the experiments performed by Sanders et al. (1992). If the taxonomy is not confirmed, it should be adapted according to the outcome of the experiments. After all, as Reape and Mellish (1999) have claimed, NLG systems should be based on linguistic theories and linguistic evidence to be truly successful.

8.4 Aggregation and ellipsis

Although our results for syntactic aggregation and ellipsis are promising, and certainly are a big improvement over the results of a system without aggregation, the syntactic aggregation algorithm we use is still too simple and inflexible in several respects. In Section 7 we have already discussed some problems with Stripping and Constituent Coordination. For these problems, fairly straightforward solutions are available, but they have not been incorporated in the system yet.

In our system, Gapping only applies to the main verb. However, in certain contexts it should be allowed to ellipt additional material along with the verb. An example is Kuno's *John hit Mary with a stick and Bill with a belt*, where the ellipted sentence could be interpreted as *and Bill hit Mary with a belt* in a context like the question *With what did John and Bill hit Mary?* (Kuno, 1976). As shown by the example, this extended form of Gapping is subject to several subtle contextual constraints: preferably, the ellipted material should be contextually given, the remaining material should be new or contrastive, and the non-Gapping interpretation should be less plausible. These constraints (especially the latter) are currently impossible to check in our system, so for the moment we leave this form of Gapping aside.

A more serious problem occurs with Right Node Raising. Adhering to the standard NLG pipe-line (Reiter and Dale, 2000), in our system linearization is performed after aggregation and ellipsis. This means that the final

ordering of the constituents in a sentence is still unknown when ellipsis is performed. However, the Right Node Raising construction relies on word order information, as it is the rightmost material that should be raised. In the absence of this information, Right Node Raising is performed on nodes that are *expected* to end up rightmost in the sentence. Unfortunately, this expectation is not always met, occasionally leading to unwellformed cases of ellipsis. An obvious solution to this problem would be to perform linearization after aggregation, but before ellipsis. However, this would require a substantial change in our algorithm, which we have not been able to make yet.

Finally, the general strategy of our aggregation component is to find a balance between keeping the generated texts from being 'choppy' and repetitive on the one hand, and ensuring that they remain easy to read on the other hand. We try to achieve this by applying coordination and ellipsis when this is possible while respecting certain constraints on complexity (see Section 6.1). However, it is not clear if our strategy always results in texts that are optimally suited for processing by a reader or listener. In the Virtual Storyteller, texts are presented to the listener using text-to-speech. Since understanding synthetic speech requires some concentration from the listener, long and complex sentences may be less suitable for this mode of presentation. This may also be the case for texts that are presented to younger readers. (See Siddharthan (2006), who presents a method to reduce the syntactic complexity of texts while retaining cohesion, which is essentially the opposite process of aggregation.) To investigate which is the optimal balance between fluency and comprehensibility, reading and listening experiments should be performed in which processing speed is measured for texts that differ with respect to the depth of aggregation.

9 Conclusion

In this article, we have described the generation of aggregated and elliptic sentences in a story generation system, using Dependency Trees connected by rhetorical relations ('Rhetorical Dependency Graphs') as input. For use in our aggregation component we have developed a taxonomy of the most common Dutch cue words, grouped according to the kind of discourse relations they signal. Using this taxonomy, we are able to generate several different sentences on the basis of a given Rhetorical Dependency Graph. Each relation has several cue words by which it can be expressed. These cue words lead to the use of different syntactic structures. The current system can produce paratactic and hypotactic constructions, add an adverb to individual trees, and use ellipsis to omit repeated constituents from the surface structure. We were able to produce all the desired forms of ellipsis (see Section 1), including combinations of different structures, such as Gapping and Conjunction Reduction simultaneously (e.g., *Diana wants to hug the prince but hit Brutus*). Although many improvements to our aggregation component are still possible, in its current form it already allows us

to generate a variety of sentences far greater than the boring sequence of fixed, simple sentences that were generated before.

We have claimed that the most appropriate place for syntactic aggregation is at the level of the Surface Realizer, and that the combination of Dependency Trees and rhetorical relations is excellent input for such a Surface Realizer, because Dependency Trees are easily manipulated and rhetorical relations can determine the syntactic constructions that can be used. Moreover, Dependency Trees and rhetorical relations are assumed to be largely language independent, which means that it should be relatively easy to adjust our generation system to another language; only the grammar rules specific to the generated language have to be replaced, the generation algorithm remains the same. Currently, our system only generates texts in Dutch, but we intend to put the claim of language-independence to the test by porting our Surface Realizer to English in the near future.

Acknowledgements The authors would like to thank Nanda Slabbers for her useful suggestions and for her work on improving the syntactic aggregation algorithm. Thanks are also due to our two anonymous reviewers for their helpful comments on the first version of this paper, and to Rieks op den Akker and Dennis Reidsma for their help with the implementation of the aggregation algorithm and the Rhetorical Dependency Graphs. Mariët Theune and Petra Hendriks gratefully acknowledge the Netherlands Organisation for Scientific Research, NWO (grant numbers 532.001.301 and 015.001.103 respectively). The work of Mariët Theune was partly carried out within the IMIX project, sponsored by NWO.

Notes

[1]We use the term 'cue word' to refer both to single words and to cue phrases that consist of more than one word. An alternative term often used in linguistic literature is 'discourse marker'.

[2]Lit.: *Diana left the desert, because she Brutus saw.*

[3]Lit.: *Diana entered the desert and Brutus too.*

[4]Given this Rhetorical Dependency Graph as input, a possible output of the Surface Realizer is 'Diana ziet Brutus en wordt bang. Daarom gaat ze naar de woestijn.' (*Diana sees Brutus and gets scared. Therefore she goes to the desert.*)

[5]Examples: *Mary fell and broke her leg* (Causal), *Mary is an architect and John is a lawyer* (Additive), *Mary got into the car and drove off* (Temporal), *Mary was pleased and Sue was angry* (Contrast).

[6]Currently, when the recency and complexity constraints allow more than one cue word, one of the permissible cue words is selected at random. Obviously, this is not a perfect solution since there will always be subtle differences in meaning and usage between the cue words. However, making a proper choice between possible cue words would require more semantic and pragmatic knowledge than is currently available in our system.

[7]Shaw (2002) uses directional constraints to achieve the same effect: if a recurrent element appears at the beginning or in the middle of the aggregated clauses,

Shaw's system performs 'forward deletion', removing the element from the second conjunct; if the element is at the end of the clause, 'backward deletion' is performed, removing the element from the first conjunct.

[8]Lit.: *Diana went to the desert, and Brutus too, because he wanted to explore the environment.*

[9]This argument was raised by one of our reviewers.

References

H.C. Andersen. *Sprookjes en Vertellingen*. Van Holkema en Warendorf, Bussum, 1975. Translated by W. van Eeden.

G. Bouma, G. van Noord, and R. Malouf. Alpino: Wide coverage computational analysis of Dutch. In W. Daelemans, K. Sima'an, J. Veenstra, and J. Zavrel, editors, *Computational Linguistics in the Netherlands 2000*, pages 45–59. Rodopi, 2001.

L. Cahill and M. Reape. Component tasks in applied NLG systems. Technical Report ITRI-99-05, Information Technology Research Institute, Brighton, UK, 1999.

C. Callaway and J. Lester. Evaluating the effects of natural language generation. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society (CogSci 2001)*, pages 164–169, August 2001.

H. Dalianis. Aggregation in natural language generation. *Computational Intelligence*, 15(4):384–413, 1999.

K. Harbusch and G. Kempen. ELLEIPO: A module that computes coordinative ellipsis for language generators that don't. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 115–118, April 2006.

K. Hartmann. *Right Node Raising and Gapping: Interface Conditions on Prosodic Deletion*. John Benjamins, Philadelphia, 2000.

P. Hendriks. Coherence relations, ellipsis, and contrastive topics. *Journal of Semantics*, 21(2):133–153, 2004.

F. Hielkema. Performing syntactic aggregation using discourse structures. Master's thesis, Artificial Intelligence, University of Groningen, Groningen, The Netherlands, 2005. Available at http://www-home.cs.utwente.nl/~theune/VS/.

E. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1-2):341–385, 1993.

A. Knott and R. Dale. Using linguistic phenomena to motivate a set of rhetorical relations. *Discourse Processes*, 18(1):35–62, 1994.

A. Knott and R. Dale. Choosing a set of coherence relations for text generation: A data-driven approach. In G. Adorni and M. Zock, editors, *Trends in Natural Language Generation: an Artificial Intelligence Perspective*, pages 47–67. Springer-Verlag, Berlin, 1996.

A. Knott and T. Sanders. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30:135–175, 1998.

S. Kuno. Gapping: A functional analysis. *Linguistic Inquiry*, 7:300–318, 1976.

B. Lavoie, R. Kittredge, T. Korelsky, and O. Rambow. A framework for MT and multilingual NLG systems based on uniform lexico-structural processing. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP/NAACL 2000)*, pages 60–67, April-May 2000.

W.C. Mann and S. Thompson. Rhetorical Structure Theory: A theory of text organization. Technical Report ISI/RS-87-190, ISI: Information Sciences Institute, Los Angeles, USA, 1987.

I. Mel'cuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany, 1988.

A. Neijt. *Gapping: A Contribution to Sentence Grammar*. Foris Publications, Dordrecht, 1979.

M. Reape and C. Mellish. Just what is aggregation anyway? In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 20–29, May 1999.

E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, 2000.

T. Sanders and L. Noordman. The role of coherence relations and their linguistic markers in text processing. *Discourse Processes*, 29(1):37–60, 2000.

T. Sanders, W. Spooren, and L. Noordman. Toward a taxonomy of coherence relations. *Discourse Processes*, 15:1–35, 1992.

D. Scott and C.S. de Souza. Getting the message across in RST-based text generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, New York, 1990.

J. Shaw. Segregatory coordination and ellipsis in text generation. In *Proceedings of the 17th COLING and the 36th Annual Meeting of the ACL*, pages 1220–1226, August 1998.

J. Shaw. *Clause Aggregation: An Approach to Generating Concise Text*. PhD thesis, Columbia University, New York, NY, USA, 2002.

A. Siddharthan. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4:77–109, 2006.

N. Slabbers. Narration for virtual storytelling. Master's thesis, Human Media Interaction, University of Twente, Enschede, The Netherlands, 2006. Available at http://wwwhome.cs.utwente.nl/~theune/VS/.

M. Taboada and W.C. Mann. Applications of Rhetorical Structure Theory. *Discourse Studies*, 2006. Accepted for publication.

J.H. Tai. *Coordination Reduction*. PhD thesis, Indiana University, Bloomington, IN, USA, 1969.

M. Theune, S. Rensen, R. op den Akker, D. Heylen, and A. Nijholt. Emotional characters for automatic plot creation. In S. Göbel, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix, and U. Spierling, editors, *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2004)*, Lecture Notes in Computer Science 3105, pages 95–100. Springer-

Verlag, 2004.

M. Theune, K. Meijs, D. Heylen, and R. Ordelman. Generating expressive speech for storytelling applications. *IEEE Transactions on Audio, Speech and Language Processing*, 14(4):1137–1144, 2006.

T. van der Wouden, H. Hoekstra, M. Moortgat, B. Renmans, and I. Schuurman. Syntactic analysis in the Spoken Dutch Corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, 2002.

R. van Oirsouw. *The Syntax of Coordination*. Croom Helm, London, 1987.

M. White. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4:39–75, 2006.

F. Zwarts. *Categoriale Grammatica en Algebraïsche Semantiek*. PhD thesis, Groningen University, Groningen, The Netherlands, 1986.