Generating under Global Constraints: the Case of Scripted Dialogue

Paul Piwek and Kees van Deemter

April 2007

Final Draft. The paper was published electronically 11 July 2007 by Springer in: Research on Language and Computation, DOI: 10.1007/s11168-007-9029-z

Abstract

Recently, the view of Natural Language Generation (NLG) as a Constraint Satisfaction Problem (CSP) has seen something of a revival. The aim of this paper is to examine the issues that arise when NLG is viewed as a CSP, and to introduce a novel application of constraint-based NLG, namely the Scripted Dialogue. Scripted Dialogue shares a number of crucial features with discourse, which make it possible to control the global properties of a computer-generated dialogue in the same way as those of a generated discourse. We pay particular attention to the use of soft constraints for enforcing global properties of text and dialogue. Because there has been little research into the formal properties of soft constraints in relation to generation, we start out with a theoretical exploration. We argue that, when multiple constraints are involved, it is important to define properly what is being optimised before proposing specific algorithms, and we argue that such definitions are often lacking in CSP-based NLG. We show that it can be difficult (and sometimes even impossible) to guarantee satisfaction of global constraints by following local strategies. Based on these difficulties, we propose a novel approach to the generation of discourse and dialogue which combines CSP solving with revision. Scripted Dialogue is used to illustrate this approach, which is compared with alternatives such as monitoring and estimation.

Keywords Natural Language Generation, Constraints, Scripted Dialogue

1 Introduction

Computational investigations into dialogue have typically focused on the communication *between* the dialogue participants. There are, however, many situations in which a dialogue fulfills a rather different purpose: dialogues in the

theatre, on television (in drama, sitcoms, commercials, etc.) and on the radio (in plays, commercials, etc.) are not primarily meant as vehicles for genuine communication between the interlocutors, but intended for an audience (in the theatre, in front of the TV). The purpose of these 'scripted' dialogues is primarily to affect that audience, e.g., by entertaining it, teaching it something,[1] making a point carefully, or conveying information within certain time limits. Typically, it is the effect that the dialogue *as a whole* has on the audience that counts. In short, when we look at dialogue from the perspective of an audience, the *global properties* of the dialogue are paramount.

This paper explores the consequences of viewing a scripted dialogue as something to be generated computationally, under global constraints. We start by sketching some of the ways in which global properties of generated *text* can be managed (section 2), and how these ideas carry over to the generation of scripted dialogue (section 3). At that point, it will have become clear that global constraints on generated text can be difficult to manage. In order to find the best perspective on this issue, section 4 will step back and examine the distinction between local and global decisions more closely. We conclude the section by discussing the problems posed by *soft* constraints. Having done this, we apply the lessons of section 4 to the type of scripted dialogues that are generated in the NECA system (Krenn et al, 2002), proposing a novel approach based on the management of soft constraints (section 5). We conclude with a Discussion section (section 6).

2 Controlling global properties of generated text

A Natural Language Generation (NLG) system has to make a large number of *decisions* concerning the way in which information is worded: aggregating information into paragraphs and sentences, choosing one syntactic pattern instead of another, deciding which words to use, and so on.[2] Often, such decisions can be made on the basis of local information. The choice between an active and a passive sentence, for example, usually does not take other decisions (such as the same choice involving another sentence) into account. In slightly more difficult cases, the decisions of the generator can be based on decisions that have been taken *earlier*. For example, the generator may inspect the linguistic context to the left of the generated item for deciding between a proper name and a personal pronoun. There are also decisions which require information about text spans that have not yet been generated. This happens typically when the generated text is subject to global constraints, that is informally, constraints on the text

[1] There is empirical evidence that learners benefit more from watching a dialogue, than from being presented with the same material in the form of a monologue (by a tutor). See, for example, Cox et al. (1999) and Craig et al. (2000).

[2] For a strong version of this claim, where generation is described as the problem of choosing between alternative expressions, while understanding/interpretation is essentially a problem of hypothesis management (i.e., trying to find the intended interpretation), see McDonald (1987:643).

as a whole. For example, suppose the text is generated subject to constraints on its length. Now, in order to decide whether it is necessary to use aggregation when generating some subspan (to stay below the maximum length), it is necessary to know the length of the text outside of the current subspan. This can lead to complications since at the point in time when the current span is generated, the rest of the text might not yet be available for inspection.

A nonlinguistic example may be helpful. Compare the constraint that a party has to *fit into one's living room*. Suppose the room can contain ten visitors, and ten invitations have gone out. Is it wise to send out another invitation? This depends on how many of the invitations will be accepted. 'Number of visitors' is a global constraint on the party. It is affected by more than one decision, in particular the host's decision whether or not to invite a given person, and an invitee's decision whether or not to accept. Generation decisions can be very similar.

One class of global NLG constraints involves the linguistic style of a text, as regards its degree of formality, detail, partiality, and so on (Hovy 1988). Whether a text can be regarded as *moderately formal*, for instance, depends on whether formal words and patterns are chosen *moderately often*, which makes this a quintessentially global constraint. Moreover, style constraints may contradict each other. For example, if a text has to be *impartial* as well as *informal* then the first of these constraints may be accommodated by choosing a passive construction, but the second would tend to be adversely affected by that same choice. Hovy argues that these problems make top-down planning difficult because it is hard to foresee what 'fortuitous opportunities' will arise during later stages of NLG. Perhaps more contentiously, he argues that these problems necessitate a **monitoring** approach, which keeps constant track of the degree to which a given text span satisfies each of a number of constraints (e.g., low formality, low partiality). After generating the first n sentences of the text, the remainder of the text is generated in a way that takes the degrees of satisfaction for all the style constraints into account, for example by favouring those constraints that have been least-recently satisfied. Monitoring addresses global constraints in an incremental fashion, which might make it a plausible model of spontaneous speech. It may be likened to steering a ship: when the ship is going off course, you adjust its direction. There is no guarantee that monitoring will result in a happy outcome, but it is a computationally affordable approach to a difficult problem.

Reiter (2000) discusses various ways in which the size of a generated text may be controlled.[3] We already saw that for certain generation decisions it might be necessary to know the length of the remainder of the text. Based on experiments with the STOP system, Reiter finds that the size of a text is difficult

[3]Reiter assumes that one way to meet a size constraint is to express less information, rather than to express the information more briefly. The difference with our perspective is largely inconsequential, but we believe the issues can be discussed more easily if content is left constant.

to *estimate* (i.e., predict) on the basis of an abstract document plan. He argues that *revision* of texts, based on measuring the size of an actually generated text, is the way to go: when the size measure indicates that the text is too large, the least important message in the sentence plan is deleted and the text regenerated. Revision has been claimed to be a good model of human *writing* (e.g., Hayes and Flower 1986).

Given the usual pipelined architecture for natural language generation, in which an initial semantic representation is converted, stepwise, into structures that are closer and closer to the final linguistic surface form (e.g., Reiter 1994), size is a particularly difficult kind of global constraint, since it applies to the surface form of a text, and this is why Reiter and colleagues let revision wait until a draft of the text has been generated and evaluated (i.e., after its size has been measured). In other respects, it is relatively simple, because it is one-dimensional and straightforward to define.

In this paper, we explore global constraints in the generation of scripted dialogue. The approach proposed was first introduced in Piwek & Van Deemter (2002). While we shall be building on the ideas in our earlier Piwek & Van Deemter (2002; 2003), the formal explorations in the first half of this paper will allow us to discuss the issues much more precisely than before.

3 Controlling global decisions in scripted dialogue

In this section, we compare scripted dialogue generation with one of the most widely used approaches to dialogue, the information state-based approach. We focus our comparison on the scope for controlling global properties of dialogue.

3.1 The Information State approach

Firstly, let us consider the currently prevalent approach to generating dialogue. The starting point is the use of two autonomous agents. Each of these agents is associated with their own information state (e.g., Traum et al., 1999; Traum & Larsson, 2003). The agent who holds the turn generates an utterance based on its information state. This leads to an update of the information states of both agents. Subsequently, whichever agent holds the turn in the new information states, produces the next utterance. In most implemented systems, one of the agents is a dialogue system and the other a human user; an approach akin to this one has, however, also been used for dialogue simulations involving two or more software agents (as pioneered by Power, 1979).

For our purposes, it is important to note that the agents only have access to their own information state, and can only use this state to produce contributions. This has repercussions for controlling global properties of dialogue. **Estimation** becomes especially difficult if it involves a span that is to be generated by the other agent. One agent has no access to the Information State of another

and will therefore find it more difficult to estimate what it is going to say. Furthermore, estimating one's own future utterances can become more difficult, since they may depend on utterances by the other agents.

Revision is also much more limited if the information state approach is strictly followed. Revision is only possible within a turn. The information state approach assumes that turns are produced according to their chronological ordering, and hence it is not possible to go back to a turn once it has been completed.

3.2 The Dialogue Scripting approach

An alternative to the information state approach is the dialogue scripting approach. According to Piwek & Van Deemter (2002) the main characteristic of this approach is that it involves the creation of a dialogue (script) by one single agent, i.e., the script author. Thus, the production of the dialogue is seen as analogous to single-author text generation.

The automated generation of scripted dialogue has been pioneered by André et al. (2000), and has subsequently been explored in the NECA and the CRAGS projects (Isard et al., 2006). We follow André et al. (2000) in distinguishing between the creation of the dialogue text, i.e., the script and the performance of this script. Of course the information state approach also lends itself for such a separation, but typically the authoring and performance functions are taken care of by the same agent (one of the interlocutors) and take place at the same time. In the scripting approach, the script for the entire dialogue is produced first. The performance of this script takes place at a later time, typically by actors who are different from the author.

There are at least two reasons why the scripting approach is better suited to creating dialogues with certain global properties than information state approaches. Firstly, in the scripted dialogue approach the information and control resides with one author. This makes estimation more reliable, assuming that it is easier to predict one's own actions than those of another agent. Secondly, the scripting approach does not presuppose that the dialogue is created in the same temporal order in which it is to be performed. Hence it is possible to revisit spans of the dialogue and edit them.

In between traditional information state and scripted dialogue approaches, *hybrid* approaches are possible. For instance, one might generate a dialogue according to the information state approach, and then edit this draft with a single author. The techniques described in the next section are presented in the context of pure dialogue scripting but they remain largely valid for more hybrid set-ups.

Despite the existence of hybrid approaches, it is important to keep in mind the different perspectives from which information state and scripted dialogue approaches arose: the information state approach focuses on the communication between the interlocutors in the dialogue, whereas the scripted dialogue approach focuses on the communication between the script author and the readers/audience of the dialogue; the communication between the interlocutors of

the scripted dialogue is only pretended communication.

3.3 Two Global Constraints on Dialogue

For concreteness, we focus on two global constraints. The first concerns the length of a generated dialogue. Length constraints have been discussed in Reiter (2000) and the considerations discussed there mostly apply to dialogue as well, though there are subtle differences. For Reiter, the problem was to fit information on a fixed number of pages. For dialogue, there are several options regarding the size measure. If the dialogue is to be presented on paper, the size constraint is likely to be similar to that of Reiter. On the other hand, if the dialogue is to be performed, the size is more likely to be calculated in terms of the duration of the performance. For our purpose, this difference will, however, be immaterial. Although we will use a specific size measure, nothing in our approach rests on the specific nature of this measure.

The second constraint concerns the amount of emphasized information in a dialogue. We will discuss this constraint in detail in section 5, where we explain how dialogue lends itself for a specific type of emphasis marking. For now, let us focus on the arguably simplest means of achieving emphasis, i.e., *verbatim* repetition (see Hovy 1988; another means that will not be considered here is the use prosody). For instance, compare 'this is a very fast car' with 'this is a very very fast car', where in the second sentence 'very' is emphasized through repetition. A side effect of this type of emphasis marking is that the overall utterance size is increased. As a result, a constraint on emphasis (e.g., emphasize as much as possible) interacts with a size constraint (e.g., keep your utterances as short as possible). This moves the problem of constraints in generation away from the setting discussed in Reiter, where only a single type of constraint was taken into account. We will discuss alternative ways of combining multiple constraints and identify an approach to multiple constraints that is specifically suited for the aforementioned two constraints on dialogue.

4 Constraints in Generation: Motivation, Limitations and Complications

Before we focus on specific constraints in dialogue generation, we shall look at constraints in generation from a more abstract point of view. First, we briefly motivate the constraint-based view of generation. Next, we show that even simple global constraints can often not be solved using local decisions. Finally, we lay out the various options that are available once multiple soft constraints are used.

Constraint-based approaches are particularly attractive for generation for a number of reasons. Firstly, a constraint-based approach liberates NLG from the narrow view of *mapping* (from non-linguistic representations to expressions in a natural language) as its primary concern. It acknowledges that faithful expression of some input representation is only one of many requirements that

can be imposed on the output of an NLG system; in many settings it is not even the most important one.[4] One of the first to employ the idea that language generation involves addressing multiple constraints at once is Appelt (1985). Although he speaks of 'goals' rather than 'constraints', his ideas are suggestive of a constraint-based approach:

> "One must constantly bear in mind that language behavior is part of a coherent plan and is directed toward satisfying the speaker's goals. Furthermore, sentences are not straightforward actions that satisfy only a single goal. The utterances that people produce are crafted with great sophistication to satisfy multiple goals at different communicative levels." (on pages 1–2 of Appelt, 1985)

Constraints enable a *declarative* approach to solving generation problems. The main advantage of this is that one can separate a clear formulation of *optimal* solutions to the generation problem from *heuristics-based* algorithms for finding solutions. (We will return to this point at various places in this paper.) Constraints are particularly well-suited for addressing problems that involve *interactions* between information of different types, granularity and levels of specification (see Langkilde-Geary, 2004).

4.1 Formalizing the notion of a constraint

The notion of a constraint can be formalized in many different ways. Here, we broadly follow the conception of a (soft) constraint as put forward in Bistarelli et al. (1997), Bistarelli et al. (2004), and Dubois et al. (1996), which – as pointed out by Dubois et al. – is rooted in early work on constraint satisfaction in computer vision (Waltz, 1975; Rosenfeld et al., 1976). The current formulation provides us with a uniform characterization of hard and soft constraints. Later in this paper, we make use of this when we formally examine local constraints and construct proofs that apply to both soft and hard constraints.

The notion of constraint is defined as follows, given a universe E of objects to which the constraint applies:

> CONSTRAINT A constraint c for a universe E is a function from E to a range S of satisfaction (or non-satisfaction) levels, i.e., $c : E \mapsto S$.

Now **hard constraints** can be characterized as constraints for which there are only two satisfaction levels, i.e., $S = \{1, 0\}$ with $0 < 1$. Usually (though not always), 1 is 'better' than 0. Thus, a hard constraint returns 1 for objects which satisfy it and 0 for objects which do not. A hard constraint allows only these two alternatives: satisfied and not satisfied. In contrast, **soft** or **flexible**

[4] In the generation of poetry, for example, factors such as rhyme and alliteration can be more important than faithful expression of content (see Manurung et al. 2000). In some generation systems, even content determination can go beyond straightforward mapping of information, when content itself is selected based on constraints (Van Deemter and Odijk 1997).

constraints may have more than two levels of satisfaction or non-satisfaction. If levels of *non*-satisfaction – rather than satisfaction – are taken as basic (for example, when 1 is 'worse' than 0), one often speaks of a *cost* function.

Two types of soft constraints can be distinguished: (1) the ones discussed here, which have more than two degrees of (dis)satisfaction (see, e.g., Dubois et al. 1996; Bistarelli et al. 1997); and (2) the ones which, while Boolean in nature, are combined in such a way that they do not all have to be satisfied; for example, one might legislate that a majority of constraints needs to be satisfied, or constraints may be preferentially ordered with respect to each other, causing some constraints to have 'right of way' over others. In Dubois et al. (1996), this type of softness is modelled by attaching a priority degree to each constraint, where the priority degree indicates to what extent it is imperative that the constraint is satisfied. The two types of softness can be treated simultaneously (as worked out in, e.g., Dubois et al., 1996); this is, however, beyond the scope of the current paper.

Levels of satisfaction in the sense discussed here are also known as measures of *desirability*, *feasibility*, *importance* or *preference*. Satisfaction levels can, for example, be represented by $\{0, \frac{1}{2}, 1\}$ with $0 < \frac{1}{2} < 1$ and $\frac{1}{2}$ representing partial satisfaction. This is, however, only one among many other possible structures, that include, for instance, the natural numbers with 0 as not satisfied and $1, 2, \ldots$ as increasing levels of satisfaction.

Given the notion of a constraint as defined above, we can now define the notion of a Constraint Satisfaction Problem(CSP):

> CONSTRAINT SATISFACTION PROBLEM (CSP) A CSP consists of (1) a constraint c, (2) a subset A of the universe E and (3) (optionally) a threshold $t \in S$. A *solution* to a CSP is an object $e \in A$ with $c(e) = s$ such that no other object in A is assigned a higher satisfaction level (or lower non-satisfaction level) by c than s, and $s > t$ (or $s < t$ if we are dealing with non-satisfaction, i.e., if c is a cost function).[5]

Note that e does not need to be unique: a CSP can have multiple solutions. In our formulation of a CSP, we deal with only a single constraint. Multiple constraints are modelled as a single constraint that is formed from multiple constraints using a combination operator (we discuss constraint combination in detail in section 4.4).

In constraint-based NLG, the above-mentioned *objects* are usually spans of text, while constraints can pertain to any aspect of such a span. Constraint-based NLG is no monolithic entity: there is a variety of approaches and architectures, each with its own advantages (Piwek & Van Deemter, 2006). In all these cases, however, the generator can be viewed as a structured network of choices.

[5]In this definition, *solutions* are *optimal* in the sense that no other members of the universe E are better than any of the solutions. Additionally, solutions need to meet the threshold, provided that there is one. Alternatively, we could have stipulated that meeting the threshold is by itself sufficient. This, however, assumes that there is a threshold. For many applications of soft constraints in NLG, a threshold is not available; e.g., none of the 25 papers on constraint-based NLG that are discussed in Piwek & Van Deemter (2006) uses thresholds.

4.2 A Generator as a network of choices

McDonald (1987:643) observes that generation is the task of choosing between alternative expressions. This view also underlies Systemic Functional Grammar (SFG), one of the most influential approaches to generation (see, e.g., Bateman, 1997). At the core of this approach is a very general formalism, the AND/OR-graph, for mapping out the space of choices that need to be made during generation. An AND-node with the children a and b indicates that both the decisions a and b need to be taken. An OR-node with children a and b indicates that a choice between a and b needs to be made. The formalism also allows explicit formulation of dependencies between nodes: e.g., 'if the choice for node X is a, then the choice for node Y is b or b''.

The network represents a paradigmatic grammar which lays out the permissible combinations of expressive choices. A number of alternative approaches have been proposed for making specific choices. Here we adopt the perspective developed in the Penman text generation system (Mann, 1985). OR-nodes are associated with *choosers* which select an alternative from the set of available alternatives. A chooser bases its decision on information obtained through a number of inquiries, e.g., inquiries on aspects of the concepts that need to be expressed and inquiries regarding the style or genre of the text to be generated. Note that the division of labour between the network and the choosers nicely separates grammar (syntactically correct ways of expressing oneself) from additional constraints (e.g., expressing a certain semantic content, or adhering to a particular style or other global constraint). This division of labour is also present in the Generative Grammar tradition, and may be traced back to Chomsky's separation of meaning from grammaticality, as illustrated by the grammatically correct, but meaningless 'colorless green ideas sleep furiously' (Chomsky, 1957:15).

4.3 Local decisions and their limitations

In this section, we investigate the notion of a local decision and ask under what circumstances a constraint can be enforced through taking local decisions.

4.3.1 Local decisions and local choosers

Following SFG, we view generation as involving choices that are represented by the OR-nodes in an AND/OR-graph. The daughters of each OR-node can be thought of as storing information about a separate, independent aspect of a generated object (e.g., discourse or dialogue). In other words, each OR-node represents a decision problem and its daughters denote decision values. The word 'decision' will sometimes denote a decision value, sometimes a decision problem.

We write '$u \in D_1$' to say that u is a daughter/decision value of OR-node/decision problem D_1. We write $P(v, w_1, w_2, \ldots, w_n)$ to say that the combination of decisions $\{v, w_1, w_2, \ldots, w_n\}$ fulfills all the relevant constraints. (In many cases

there is only one constraint.) If we assume that a decision value bears its decision problem on its sleeve then the order of P's arguments does not matter.

We now define a notion of locality, assuming that the complete set of all the relevant decision problems involved in a given generation task is $D, D_1, \ldots, D_n$, where D is the decision of interest.

> DEFINITION: The decision D can be taken locally $\Leftrightarrow_{Def}$ $\exists u \in D$ such that $\forall w_1, \ldots, w_n \in D_1, \ldots, D_n$, if $\exists v \in D$ such that $P(v, w_1, \ldots, w_n)$ then $P(u, w_1, \ldots, w_n)$.

Informally: The decision D can be taken locally if and only if there is a decision value u for this decision problem such that regardless of all the other decisions $D_1, \ldots, D_n$, either these other decisions, let us call them, $w_1, \ldots, w_n$, make it *impossible* to fulfill the constraints (i.e., $\neg\exists v \in D$ such that $P(v, w_1, \ldots, w_n)$), or you can fulfill them while also making the decision u. If the decision D can be taken locally then the success of the decision does *not* depend on the $D_1, \ldots, D_n$.

Note that the fact that a decision can be taken locally does not mean that there necessarily exists a local chooser (i.e., an algorithm) for making this decision, because some local decisions may be uncomputable. If, however, there exists no local decision, then of course no local chooser can be constructed. In what follows, we will mainly be interested in theorems concerning the existence of a local decision. Algorithms will be discussed in section 5, where we focus on decisions involving scripted dialogue.

4.3.2 Constraints that satisfy Monotonicity

Let us explore what kinds of global constraints can be solved with local decisions, whilst continuing to focus on networks as generators of text (rather than objects of a different kind). We begin by examining a simple class of networks, which fulfill the following conditions: (P1) there are no dependencies between decisions, i.e., there are no explicitly formulated dependencies, and (P2) no part of the generated text is affected by more than one decision (OR-node). In many cases of interest, these conditions are, of course, not met. (Penman's choosers, for example, are notoriously dependent on each other.) For now, however, it will be useful to make these simplifying assumptions. In section 5, where we return to Scripted Dialogue, they will be abandoned.

Let S' be a (possibly non-contiguous) substring of S, in which case we write $S' \sqsubseteq S$. Suppose a soft constraint c assigns a preference to both S and its substrings (for instance assigning a number in the interval $[0, 1]$ to each substring, with 0 standing for non-satisfaction and 1 for complete satisfaction of the constraint). Then the following is a property that c might or might not have:

> *Full Monotonicity of Satisfaction (FMoS) of constraint c*: For all conceivable strings S_1 and S'_1 such that $S'_1 \sqsubseteq S_1$, if we replace S'_1 with S'_2 to obtain S_2 then:

(1) if $c(S'_2) < c(S'_1)$, then $c(S_2) \leq c(S_1)$;
(2) if $c(S'_2) = c(S'_1)$, then $c(S_2) = c(S_1)$;
(3) if $c(S'_2) > c(S'_1)$, then $c(S_2) \geq c(S_1)$.

FMoS is useful mainly for the following reason: If c satisfies FMoS then there exist local decisions that guarantee that the degree to which c is satisfied is maximised. Before stating and proving this theorem more precisely, some new terminology will be useful. Given condition P2 and the fact that there are exactly i OR-nodes, the text region R which is to be generated (and which can still take different textual forms!) can be divided into i non-overlapping regions R_i, each of which is 'filled' by the decision D_i alone. (Some regions of the text may not be affected by any decisions, which means that their form is fixed.) We write S_i, S'_i, etc., for the different texts that may result from the decision D_i, so S_i and S'_i are different ways in which the region R_i may pan out; similarly, S and S' are different ways in which the region R as a whole may take shape. By letting $c(R_i)$ denote the degree to which the region R_i satisfies the constraint c given a certain way in which R_i is filled, we shall be able to speak conveniently about maximising $c(R_i)$ (i.e., choosing an S_i such that $c(S_i)$ is maximal).

THEOREM FMoS locality: Let R be a text region governed by the decisions $D_1, \ldots, D_n$ while conditions P1 and P2 hold. Then if c fulfils FMoS, the decisions $D_1, \ldots, D_n$ can be taken locally in such a way that satisfaction of $c(R)$ is maximised.

PROOF: Suppose each decision D_i is taken in such a way that $c(R_i)$ is maximised, and that the result of these decisions is the text S; then FMoS guarantees that $c(R)$ itself is also maximised. For suppose there exists a text S' such that $c(S') > c(S)$. S' can only differ from S as a result of different decisions for some of $D_1, \ldots, D_n$. Given FMoS, $c(S') > c(S)$ implies that there exists at least one i for which $c(S'_i) > c(S_i)$, contradicting the assumption that each decision D_i is taken in such a way that $c(S_i)$ is maximised. If the number of each D_i's decision-values is finite, and if there exists an algorithm for calculating $c(X)$ (for an arbitrary text span X) then there is an algorithm for maximising each $c(R_i)$, and hence there exists an algorithm for maximising $c(R)$. In other word, in that case, there exist *local choosers* for maximising $c(R)$.

4.3.3 Constraints that do not satisfy Monotonicity

Let us illustrate the abstract issues relating to local strategies using some extremely simple examples, which involve the generation of a sentence rather than a dialogue (*cf.* section 5). As long as we stick to the conditions P1 and P2 above, we shall use a simple notational convention whereby the subscript on a decision

indicates to which of the (non-overlapping) parts of the text the decision applies. One easy way of thinking about this is to imagine that D_1 generates the leftmost part of the text, after which D_2 generates the part immediately to its right, and so on. We write $D_i :== A_1 \mid \ldots \mid A_n$ to signify that the decision (OR-node) D_i involves a choice between the alternatives $A_1, \ldots, A_n$. Consider:

Network N_1
$D_1 :== \mathit{The\ Porsche} \mid \mathit{The\ Ferraris}$
$D_2 :== \mathit{is} \mid \mathit{are}$
$D_3 :== \mathit{very} \mid \epsilon$
$D_4 :== \mathit{very} \mid \epsilon$
$D_5 :== \mathit{very} \mid \epsilon$
$D_6 :== \mathit{fast}$

Suppose this network is used to generate text that accurately expresses some given semantic content. The chooser for D_1 will need to make an inquiry about this content in order to determine what type of car is discussed. Similarly, the chooser for D_2 will need to investigate whether we are dealing with one or multiple objects.[6] The third, fourth and fifth decision involve the amount of emphasis. Now, let us assume that no specific value is provided for this (i.e., the degree of emphasis is not specified), and thus for each decision both *very* and the empty string ϵ are available. However, let us also assume that a soft *constraint* c_{S1} is in place, which values strings with more than 5 words more highly than ones with 5 or fewer words. In particular, for strings s with 6 or more words, we have $c_{S1}(s) = 1$, whereas if there are fewer than 6 words, then c_{S1} decreases with string length. This constraint obeys FMoS, and therefore we can now apply local decisions for maximising $c_{S1}(R_i)$to obtain a globally optimal solution (see theorem FMoS locality): for D_3 to D_5 we chose *very* since $c_{S1}(\mathit{very}) > c_{S1}(\epsilon)$. The resulting text (assuming an underlying semantics has guided the other decisions) is *The Porsche is very very very fast.*

Although, in this paper the focus is on soft constraints, note that the local chooser for the soft constraint also works for the corresponding hard constraint c_{H1} such that $c_{H1}(x) = 1$ if $c_{S1}(x) = 1$ and else $c_{H1}(x) = 0$. This hard constraint can be paraphrased as *the number of words is at least 6.* In general, if a soft constraint can be solved with local choosers, the corresponding hard constraint can also be solved with those very same local choosers.

Let us now investigate the mirror image of this constraint, that is, *the number of*

[6]The validity of these choosers follows from the constraint that the input semantics should be realised accurately. Note that such a constraint can be dealt with in the current framework. (If we want to generate a string S whose semantics is characterised by some formula f, then the totality of choices must lead to a string with the semantics f. In other words, we have a hard constraint that the semantics of the generated string equals f.) In this paper, we do not investigate how to enforce such semantic constraints in detail. Our main interest lies with constraints that differentiate between realizations, regardless of whether the underlying semantics is more or less the same. By using a declarative constraint-based approach, these issues can be treated separately from each other.

words is at most 6. Again, we first set up a soft constraint c_{S2}. This constraint values strings with 6 or fewer words as 1, and strings with 7 or more words with decreasing values. c_{H2} is obtained from c_{S2} in the same way as c_{H1} from c_{S1}. Unfortunately, this constraint does not satisfy $FMoS$, though it does satisfy a 'partial' variant of it:

> *Partial Monotonicity of Satisfaction (PMoS)* For all conceivable strings S_1 and S'_1 such that $S'_1 \sqsubseteq S_1$, if we replace S'_1 with S'_2 to obtain S_2 then:
> (1) if $c(S'_2) < c(S'_1)$, then $c(S_2) \leq c(S_1)$;
> (2) if $c(S'_2) > c(S'_1)$, then $c(S_2) \geq c(S_1)$.

The cause of the asymmetry between c_{S1} and c_{S2} is that local choosers manipulate 'small' substrings, and these have a different status in the two kinds of constraints. Thus, local optimization does not work for c_{S2}. There is, however, an alternative chooser, based on the observation that degree of satisfaction associated with a soft constraint fails to distinguish between different strings (e.g., a string of length 1 and one of length 0), each of which satisfy a constraint completely (e.g., length is at most 6), but which can have different effects on a larger string when they are embedded into it. In the case of the constraints of interest, one can define a reasonable notion of 'distance to non/partial- satisfaction' (dn) in terms of standard edit distance (ld, i.e., Levenshtein distance): Given a constraint c and assuming that $c(S) = 1$, then $dn(S)$ is the edit distance $ld(S, S')$ between S and some S' for which $c(S') \neq 1$, and such that there is no $S'' \neq S'$ with $c(S'') \neq 1$ such that $ld(S, S'') < ld(S, S')$. Thus, $dn(S)$ denotes the smallest number of edit operations that will transform S into some S' such that $c(S') \neq 1$.

For example, for c_{S2}, strings of length 4 are closer to non-satisfaction than those of length 3. Strings which satisfy c_{S2} have the following property in addition to *PMoS*.

> *Monotonicity of Distance to Non-satisfaction (MoDNS)*: For all conceivable strings S_1 and S'_1 such that $S'_1 \sqsubseteq S_1$, if we replace S'_1 with S'_2 to obtain S_2, and $c(S'_1) = c(S'_2) = 1$ then the following implications hold:
> (1) If $dn(S'_2) < dn(S'_1)$ then $c(S_2) \leq c(S_1)$;
> (2) If $dn(S'_2) = dn(S'_1)$ then $c(S_2) = c(S_1)$.

A chooser for optimal satisfaction of constraints that obeys PMoS and MoDNS is easily derived from this: based on PMoS we still maximise c, but now for those strings x and y where $c(x) = c(y)$, we also maximise dn (which according to MoDNS, leads to an optimal value for c globally).

Wherever there are contrasting requirements (e.g., containing more than x and fewer than y words, where $x < y$), being safer with respect to one of the two means being less safe with respect to the other. This is illustrated by taking the combination $c_{S3}(z) = min(c_{S1}(z), c_{S2}(z))$ of c_{S1} and c_{S2}. It can be paraphrased

as *the number of words is as close as possible or equal to 6*. The corresponding hard constraint c_{H3} (*the number of words is exactly 6*) is another example.

Neither c_{S3} nor c_{H3} satisfies any of FMoS, PMoS and MoDNS, and consequently there exist networks for which no local chooser can guarantee optimal results. Note that this means that given a hard constraint, even if we can find a corresponding soft constraint, that soft constraint is not guaranteed to be solvable with local choosers. This happens whenever, for any chooser for D, every choice that it makes can be 'spoilt' by the chooser for another decision. The following network is a simple case in point, in combination with a constraint c_{H4} that says *'the number of words is exactly* 3*'*:

Network N_2
D_1 :== $He \mid The\ man$
D_2 :== $left \mid went\ away$

Consider the chooser for D_1 in this network, for example. If D_1 chooses 'He' then this *can* lead to a sentence that fulfills the constraint, but only if D_2 chooses 'went away'; if D_1 chooses 'The man' then this too can lead to a sentence that fulfills the constraint, but only if D_2 chooses 'left'. (The wisdom of one choice depends on what other choice is made.) Thus, there is no local decision for D_1 that *guarantees* satisfaction of c_H4.

So far, we have worked with networks that satisfy the properties P1 (there are no dependencies) and P2 (each decision pertains to a single span of text). Even for such simple networks, local choosers are not always available, as we have seen. But if we abandon these properties, e.g., P2, then things get even worse. Consider the following artificial example:

Network N_3
D_1 :== Choose a real number x between 0 and 1
D_2 :== Choose two subsequent natural numbers (e.g., n and m).

Suppose the effect of these choices is to choose the n'th and m'th digit in the decimal development of x (not counting the initial 0). For example, suppose D_1 chooses the number 0.53453534485463840463. Now if D_2 chooses 'first and second' (i.e., $n = 1, m = 2$) then the string 53 is generated; if D_2 chooses '13th and 19th' then the string 66 is generated. Now suppose a hard *constraint* c requires that the string should contain at least one occurrence of the digit 5. Now the following hold:

1. The constraint c satisfies FMoS.
2. D_1 can be decided locally. A safe choice is $0.5 \ldots 5$.
3. But D_2 cannot be decided locally. Consider e.g. the decision D_2 = 'third and fourth'. This is not a safe choice, for consider the decision $D_1 = 0.537432 \ldots 2$ (among many other possible choices). This decision *can* fulfil c (e.g. if we decide D_2= 'second and third'), but not in combination with D_2 = 'third and fourth'. All other choices for D_2 are equally unsafe.

The point is simple but important: if several constraints *conspire* to determine a particular part of the generated text (as is prohibited by condition P2) then whether a given choice is optimal may depend on the others. Clearly, in such situations, local choosers are of limited value. The same is true if choices interact (as is prohibited by P1).

4.3.4 Strategies for enforcing non-local decisions

In light of this discussion, let us examine the different strategies that have been proposed for dealing with non-local decisions:

1. Monitoring. Clearly, in some situations, monitoring is too weak a strategy. Consider the network N_4,

Network N_4
$D_1 :== a \mid aa \mid aaa$
$D_2 :== a \mid aa$

with the constraint that the generated string contain exactly four characters. Now suppose the generator starts with D_1, choosing a. This represents a 'false start': an error has been made that later decisions cannot repair. Monitoring is a computationally affordable method that is sometimes not powerful enough to guarantee that some perfectly satisfiable constraints are satisfied.

2. Estimation. Estimation is more powerful than monitoring, since it works on the basis of (imperfect) predictions concerning decisions that may follow. That estimation is sometimes too weak to fulfil a given constraint can be seen if we consider network N_5:

Network N_5:
$D_1 :== a|aa|aaaaaa$
$D_2 :== a|aaaaaaaaa$,

while the constraint c specifies that *the total number of characters is* 7. If estimation operates by counting the average number of characters that result from the different alternatives for D_2 then the *prediction* is that rewriting D_2 will introduce 5 (average of 1 and 9) terminals. Estimation would suggest that to arrive at a total of 7 terminals, as specified by c, D_1 would need to choose aa. If this is done, however, then the two available alternatives for D_2 introduce 1 and 9 characters respectively, leading to a total of either 3 or 11 characters, both of which are wide off the mark, whereas a choice by D_1 for $aaaaaa$, and then by D_2 for a would have produced a total of exactly the seven required terminals. Estimation strategies may of course use other methods (i.e., other than using the average of all possible future rewritings), but as long as they do not list all possibilities separately, estimation can cause a system to miss out on a perfectly achievable solution.

3. Revision. Revision may be defined in different ways. Informally speaking,

we take it to mean that some or all rewritings are applied tentatively, the result of which may be modified if they are in breach of a constraint. Revision can be performed at the end of the generation process, or earlier. Also, revision might take the shape of some simple changes to the generated structure (e.g., Inui et al. 1992) or it might replace generative decisions by alternative ones. In this case, revision may even amount to full backtracking, which means that, potentially, all decisions (at all points where decisions were taken) can be tried out. When revision can examine all possible decisions, it becomes similar to a full *overgenerate-and-test* approach, in which all possible decisions are carried out and compared. Methods of this kind amount to an exhaustive search and are therefore much stronger than monitoring and estimation. For these reasons, they are a natural candidate for enforcing global constraints, which is why, in section 5, we shall explore how revision can be applied to the problem of enforcing global constraints in the generation of scripted dialogue. Before doing this, we shall briefly examine what happens when *soft* constraints enter the picture.

4.4 Combining Soft Constraints

So far, we have considered one example of multiple constraints (combining c_{H1} and c_{H2} and the corresponding c_{S1} and c_{S2}). The constraint derived from combining two hard constraints was simply the *min* operation (i.e., logical conjunction): the value of a state under the combined constraint is 1 if and only if it was 1 under both individual constraints. For combining the soft constraints c_{S1} and c_{S2} we also used *min*. In general, however, matters are less straightforward when it comes to soft constraints, since these can be combined in different ways. Once again, a simple abstract network will illustrate the point:

Network N_6:
D_1 :== $a \mid aa \mid aaa$
D_2 :== $b \mid bb$

The following two soft constraints are in place:

CL. *The length of the string should be as close as possible to 2.*
CN. *The number of a's in the string should be as high as possible.*

Variants of the first constraint have already been discussed in the context of Reiter's work (Reiter, 2000); an application of the second type of constraint will be described in section 5.

Let us specify each of the soft constraints in a manner that will allow us to illustrate what can happen if different constraints have been satisfied to a limited degree. For CL, the maximum score should be returned when the length of the string is 2. The function we create assigns 100(%) for string length 2. As we move away from this ideal, the value decreases to 0 and for any strings with length equal to 4 or more, we also obtain 0: for the interval [0,2] we use the linear function $f(x) = 50x$, and for [2,4] we use $f(x) = -50x + 200$. Finally, for

[4,∞) we use $f(x) = 0$. For CN, the maximum score is associated with strings containing three occurrences of the character a, i.e., maximum of a's which a string generated by G_4 can contain. The minimum score, that is 0, is returned for strings that contain no a's. For the interval [0,3] we use the linear function $f(x) = \frac{100}{3}x$.

Table 1: Strings Generated by N_6

string	CL	CN	multiplication	addition	min
ab	100.00	33.33	3333.00	133.33	33.33
abb	50.00	33.33	1666.50	83.33	33.33
aab	50.00	66.66	3333.00	116.66	50.00
aabb	0.00	66.66	0.00	66.66	0.00
aaab	0.00	100.00	0.00	100.00	0.00
aaabb	0.00	100.00	0.00	100	.00

Table 1 lists the set of strings generated by N_6 and shows the values for CL and CN for each string, along with some different ways of combining these values: multiplication, addition and minimum. For each of these operators, the maximum value is different: for multiplication we have a tie between *ab* and *aab*; for addition, *ab* is the winner; and for min, *aab* is the winner.

Where a plurality of soft constraints play a role, they can be balanced in different ways. Which of these is best will depend on the setting in which the constraints are applied: sometimes, for example, it may be better to maximise one constraint at the cost of others, while in other situations a more even-handed approach is preferable. Current practice in NLG is, however, often to use one of these alternatives without further motivation or, even worse, simply to implement some search strategy for finding a solution without specifying at all which solutions it is intended to approximate.[7] In our view, the best way to design a generation system is to first specify what the optimal solutions are and then to devise an algorithm for finding or approximating such solutions. Too often, an algorithm is designed without a clear specification of what would count as an optimal solution.[8] In the remainder of this paper, we discuss these issues in more detail by means of a concrete example of a generation system for scripted dialogue.

[7] In Piwek and Van Deemter (2006), we survey 25 approaches to constraint-based NLG of which 8 deal with multiple soft constraints. Of these only 3 discuss the motivation of the combination operator that was used (in the survey, the following combination operations were found: least satisfied precedence, weighted addition, multiplication and bidirectional superoptimality).

[8] Of course the limitations of monitoring, estimation, and limited revision may sometimes be outweighed by their computational advantages, especially when the number of rewriting rules is large. In this paper, however, we will address two logically prior questions: *What is an ideal solution?*, and *How might such a solution be achieved algorithmically?* See also section 5.1.2.3.

5 Scripted Dialogue Generation in NECA

We describe here an implemented system called NECA (Krenn et al., 2002), in which our approach to dialogue scripting, and the use of global soft constraints in particular, has been explored.

The NECA system generates dialogue scripts that are performed by animated characters. The ideas of this paper have been implemented in SICSTUS Prolog, as a stand-alone variant of one of the two NECA prototypes, called *eShowroom*, which focusses on the generation of car sales dialogues. By taking a concrete dialogue system as our point of departure we were able to try out our ideas in an existing system that was not built with the problem of controlling global properties in mind.

As we discussed in the introduction to this paper, scripted dialogue has many different potential applications, including entertainment and education. The eShowroom demonstrator is intended to both entertain and educate. It was developed as the front-end to a car sales portal: users browse a database of cars, select a car, select two characters and their attributes, and subsequently view an automatically generated film of a dialogue between the characters about the selected car. The dialogues are meant to be be entertaining (to to attract visitors to the portal) as well as informative. The dialogues go beyond the recital of technical information about the car: for the benefit of non-expert car buyers, the dialogues link technical information with broad categories such as luxury, friendliness for the environment, etc.

The eShowroom system has the following inputs:

- A database with facts about the selected car (maximum speed, horse power, fuel consumption, etc.).
- A database which correlates facts with value dimensions such as 'sportiness', 'environmental-friendliness', etc. (e.g., a high maximum speed is good for 'sportiness', high gasoline consumption is bad for the environment).
- Information about the conversational characters (e.g., role and personality).

This input is processed in a pipeline that consists of:

1. A Dialogue Planner, which produces an abstract description of the dialogue (the dialogue plan).
2. A multi-modal generator which specifies linguistic and non-linguistic realizations for the dialogue acts in the dialogue plan.
3. A Speech Synthesis Module, which adds information for Speech.

4. A Gesture Assignment Module, which controls the temporal coordination of gestures and speech.

5. A Player, which plays the animated characters and the corresponding speech sound files.

Each step in the pipeline adds information to the dialogue plan/script until finally a player can render it. A single XML-compliant representation language, called NECA Rich Representation Language (RRL), has been developed for representing the Dialogue Script at its various stages of completion (see Piwek et al., 2002). The following is a transcript of a dialogue fragment which the system currently generates. (Note that this is only the text. The system actually produces spoken dialogue accompanied by gestures of the embodied agents which perform the script):

SELLER: Hello! How can I help?
BUYER: Can you tell me something about this car?
SELLER: It is very comfortable.
SELLER: It has leather seats.
BUYER: How much does it consume?
SELLER: It consumes 8 liters per 60 miles.
BUYER: I see.
ETC.

Here, we focus on the representation of this dialogue after it has been processed by the Dialogue Planning module. The RRL dialogue script consists of four parts:

1. A representation of the initial common ground of the interlocutors in terms of a discourse representation structures (DRSs; Kamp & Reyle, 1993). This representation provides information for the generation of referring expressions.

2. A representation of each of the participants of the dialogue. It contains information on the name, sex, appearance, role (seller or buyer), etc. of the characters.

3. A representation of the dialogue acts. Each act is associated with attributes, some of which are optional, specifying its type, speaker, addressees, semantic content (in terms of DRSs), what it is a reaction to (in terms of conversation-analytical adjacency pairs) and the emotions with which it is to be expressed.

4. The fourth component of the RRL representation of the dialogue script records the temporal ordering of the dialogue acts.

5.1 Enforcing global constraints in scripted dialogue

Let us now examine how to adapt the system so that it can take global constraints into account. We have seen in section 3.2 that the *dialogue scripting* approach is best suited for the control of global dialogue properties. The NECA system is based on dialogue scripting: each module in the pipeline operates as a single author/editor who creates/elaborates the Dialogue Script. Within the Dialogue Scripting approach various methods for controlling global properties can be employed. Earlier on, we discussed various strategies that have been employed in text generation. In this section, we limit our attention to a revision approach.

Our reasons for using revision are as follows. Constraint Satisfaction as defined in Van Hentenryck (1989) does not appear to be applicable, since it is based on a set of variables whose values are optimized given a set of constraints. In the situations discussed in the present paper, it is difficult to see how the relevant variables could be assigned values directly, since they can only arise indirectly.[9] Secondly, revision will allow for a straightforward division of labour between system developers.More generally, it has been argued that a single-pass approach, as opposed to revision-based approaches, can complicate the design and maintenance of a generation system (see Callaway & Lester 1997, Reiter 2000, Robin & McKeown 1996).[10] A consequence of this is that revision often scales up better to new data (see Robin and McKeown 1996). Thirdly, as explained in section 2, various authors have claimed that revision is a more effective method for satisfying constraints (Reiter 2000, Robin and McKeown 1996). Fourthly, we favour revision over monitoring because the latter is tailored to left-to-right processing, whereas the dialogue scripting approach is not constrained in this way. Monitoring can, of course, only work well if the number of decisions relating to each constraint is large, since this gives the system many opportunities for 'changing course'.

We add a new revision module to the system described in section 5, which will operate immediately after the Dialogue Planner. Dialogue-oriented revisions will take a dialogue plan as input, and deliver another possible dialogue plan as output. A revised dialogue plan will express basically the same information as the original dialogue plan, consistent with the traits and preferences of the characters. The degree to which it satisfies other global constraints on the dialogue might, however, differ from that of the original dialogue plan.

Our approach to revision differs from other approaches. Firstly, our revisions are carried out on the abstract dialogue plan, before linguistic realization. Although Callaway & Lester also carry out their revision operations on abstract

[9]Here one can think of variables like TURN and EMPH (see section 5.1.1), where the problem is that their values only arise as a result of generation decisions involving INSERT and AGGR. Note, however, that a clever encoding is sometimes possible. For instance, Power (2000), in the context of text structuring, uses a CSP solver to produce a partial description of the solution which is then expanded by further structural rules.

[10]We could, for instance, have included the aggregation and insertion operations (see below) directly in our dialogue manager, but this would have complicated the dialogue planner rules.

representations of sentences, these are obtained by first generating concrete sentences and then abstracting again over irrelevant details. Instead of first fully generating and then abstracting, we follow an approach of partial generation. Secondly, Reiter and Callaway & Lester focus on a single type of constraint. In this respect, our work is more similar to that of Hovy, where different, potentially conflicting constraints are considered. To our knowledge, we are the first to propose revision operations on *dialogue* structure as opposed to discourse or sentence structure. Arguably, the different types of revision ought at some point be addressed through one common approach.

5.1.1 Two global constraints and two revision operations for dialogue

To illustrate the issues, let us consider four possible global constraints on dialogue:

- (TURN) Number of turns in the dialog: maximal (MAX) or minimal (MIN)
- (EMPH) Degree of Emphasis: maximal (MAX) or min (MIN)

For the moment, we keep the constraints as simple as possible and assume that they can only take extreme values (MAX or MIN).

Furthermore, we introduce two revision operations on the output of the dialogue planner: aggregation and insertion. In the definitions of the two operations we use the notion of an *adjacency pair* which is common in Conversation Analysis. The idea is that the first and second part of the pair are connected by the relation of conditional relevance (e.g., a pair consisting of a question and an answer): 'When one utterance (A) is conditionally relevant to another (S), then the occurrence of S provides for the relevance of the occurrence of A' (Schegloff, 1972:76).

ADJACENCY PAIR AGGREGATION (AGGR)

Operation: Given the adjacency pairs $A = (A_1,A_2)$ and $B = (B_1,B_2)$ in the input, create $A + B = (A_1+B_1,A_2+B_2)$.

Preconditions: A and B are about the same value dimension, and A_2 and B_2 entail answers that have the same polarity (i.e., for YES/NO questions both entail *yes* or both entail *no*).

Example: A = (Does it have airbags? Yes),
B = (Does it have ABS? Yes),
$A + B$ = (Does it have airbags and ABS? Yes)

Comment: The shared value dimension is *security*.

ADJACENCY PAIR INSERTION (INSERT)

Operation: Given adjacency pair $A = (A_1,A_2)$ in the input, 1. create adjacency pair $B = (B_1,B_2)$ which is a clarificatory subdialogue

about the information exchanged in A and 2. insert B after A, resulting in $(AB) = (A_1,A_2)(B_1,B_2)$. 3. Remove the emphasis marker on the information exchanged in A (see Precondition).

Precondition: The information exchanged in A is marked for emphasis.

Example: A = (Does it have leather seats? Yes). Assume that *comfort* is positively correlated with having leather seats and that the user has indicated that the customer prefers comfortable cars. On the basis of this, the information exchanged in A is marked for emphasis. The text after revision is: (AB) = Does it have leather seats? Yes. Real leather? Yes, genuine leather seats.

Comment: This operation was inspired by examples, discussed in Piwek & Van Deemter (2002), of how human authors of scripted dialogue appear to use sub-dialogues for emphasis.

One might ask how we decide which operations are included in the conventional topdown planner and which ones are deemed revision operations. To answer this question, it will be useful to elaborate a bit on our underlying assumptions about dialogue.

According to Clark (1996), dialogue acts belong to different 'tracks' depending on how directly they contribute to the purpose of the dialogue. The acts on track 1 contribute directly.

Meta-communication about the communication on track 1 takes place at the level of track 2. This includes monitoring the success of the communication, attempting to fix communication problems, etc. If the utterances on track 2 are omitted, the remaining dialogue script still makes sense (see Piwek & Van Deemter, 2002), whereas removing utterances from track 1 does not have the same effect.

1. BUYER: How much does the car cost?
2. SELLER: 15.000 Euro.
3. BUYER: 15.000?
4. SELLER: Yes, only 15.000.

The acts on track 1 of this dialogue (utterances 1. and 2.) make sense on their own, whereas those on track 2 (3. and 4.) do not. For this reason, acts on track 1 are dealt with by the dialogue planner, while acts on track 2 are inserted at the revision stage, by means of the operation INSERT.

AGGR introduces a new type of aggregation, on the dialogue level. Aggregation operations are typically dealt with using revision: two or more structures are merged/revised into one new structure. Our AGGR allows us to reorganize the location of dialogue acts. It does not add or remove any dialogue acts, though it can reduce the size of their realization. The precondition on AGGR, which stipulates that only dialogue acts which deal with the same value dimen-

sion can be aggregated, guards against erratic reorganizations of the dialogue, destroying smooth shifts from one topic (value dimension) to another.[11]

5.1.2 The revision problem and how to solve it

Let us now describe our revision problem. We have an initial dialogue plan dp_1, produced by the dialogue planner. Before it is passed on to the multi-modal natural language generator we want to apply the revision operations AGGR and INSERT in such a way that the resulting dialogue plan dp_2 optimally satisfies the constraints for TURN and EMPH. In total, there are four possible constraint settings:

TURN = MAX and EMPH = MAX.
TURN = MAX and EMPH = MIN.
TURN = MIN and EMPH = MIN.
TURN = MIN and EMPH = MAX.

Sequential revision Before we look at pairs of constraints, let us start with examining a single constraint, i.e, EMPH = MAX. For this constraint, a local strategy, as defined in section 4.3.1, exists; we simply apply INSERT as often as possible. In other words, whenever we need to make the decision whether to insert or not, we choose for INSERT. By choosing for INSERT, we make a decision which is locally optimal (it maximises emphasis locally), and, since emphasis for the entire dialogue is obtained by summing of local emphasis, global emphasis is also maximised. Note that the maximum number of INSERTs is determined by how many bits of information in the dialogue are marked for emphasis. The current strategy means that for each of the marked bits of information, a subdialogue that emphasizes it is inserted.

When we introduce multiple constraints, the picture is less straightforward. Let us assume that in addition to EMPH = MAX we also want to minimize the number of turns, i.e., TURN = MIN. We have seen that maximization of emphasis is associated with INSERT. Building on this, let us try a rather simplistic approach to deal with the additional constraint that TURN = MIN. We observe that AGGR brings down the number of turns, and therefore given TURN = MIN, we opt to apply AGGR as often as possible. So our strategy could be to first apply AGGR as often as possible, and subsequently INSERT as often as possible.

This solution, however, overlooks the fact that the INSERT and AGGR operations are not independent from each other. In particular, the INSERT operation can create new opportunities for applying the AGGR operation. This type of problem can usually be finessed by finding an 'optimal' ordering between operations: if insertion precedes aggregation, both constraints of our example situation can be satisfied. Unfortunately, however, there is another type of problem which cannot be finessed so easily.

[11] More generally, the generation of texts with smooth topic shifts can be seen as a constraint satisfaction problem. See, for instance, Kibble & Power (2000).

So far, we have overlooked the fact that application of INSERT positively affects the number of turns as well as the degree of emphasis, affecting the two constraints TURN = MIN and EMPH = MAX in opposite ways. In such a case it is unclear what the best strategy is: the algorithm might either maximise the number of insertions, trying to maximise emphasis, or minimise them, trying to minimise the number of turns. To tackle problems of both kinds, an approach is needed that is able to make *trade-offs* between conflicting constraints.

Stepping back: What is an optimal solution? Starting with an ad hoc approach to our revision problem, we have run into serious problems. Most importantly, we have not yet specified what would count as an optimal solution. Therefore, it is difficult to judge how well the algorithm is actually doing. So, let us take a different tack, and first try to specify what would count as an optimal solution to the problem at hand. For this, we draw upon the framework which we introduced in section 4, where we discussed alternatives ways of defining what counts as an optimal solution given a set of soft and/or global constraints. Firstly, we need to characterize the set of all candidate solutions. For current purposes, this set consists of those plans that can be obtained by applying the operations INSERT and AGGR zero or more times, in any order, to the plan which is produced by the dialogue planner. Let us call this call this set of candidate dialogue plans DP_{cand}. Members of DP_{cand} can satisfy the TURN and EMPH constraints to different degrees. Each constraint C is given as a function from DP_{cand} to [0-100], which maps candidate solutions to the degree of satisfaction of the solution:

- For TURN = MIN, $c_{turn=min}$ returns 100% for the shortest plans and 0% for the longest plans in DP_{cand}. We assume that the function is linear between these extremes.
- For TURN = MAX, we construct $c_{turn=max}$ by interchanging the extremes of $d_{turn=min}$ and adjusting satisfaction for the intermediate plans accordingly.
- For EMPH = MAX, $c_{emph=max}$ returns 100% for the plans with the highest number of emphasis subdialogues in DP_{cand} and 0% for those with the least number. The function is linear in between.
- For EMPH = MIN, we interchange the extremes of $c_{emph=max}$ and adjust satisfaction for the intermediate plans accordingly.

Having set up each constraint as a function which returns the degree of satisfaction for specific dialogue plans, we still need to say how the functions for different constraints are combined. In section 4.4 we discussed three possible combination operators (multiplication, addition and min) and argued that a specific operator should be selected on the basis of characteristics of the application in question.

Let us elaborate on the intended application of our method. We assume that a person (or program) selects a constraint setting (MIN or MAX) for EMPH and TURN. The person values each of the constraints equally and seeks a solution that satisfies each of the resulting constraints as much as possible. In principle, this informal idea can been made precise in a number of ways, while taking certain general principles into account. Perhaps the best known such principle is the idea that a solution must always fulfill Pareto Optimality. A pair of degrees of satisfaction for the turn and emphasis constraints ($\langle S_T, S_E \rangle$) is Pareto Optimal if it is impossible to improve one of its elements without making the other element worse off.

> PARETO OPTIMALITY: A pair $\langle S_T, S_E \rangle \in DP_{cand}$ is Pareto Optimal iff it is impossible to find another pair $\langle S'_T, S'_E \rangle \in DP_{out}$ such that:
> (1) $S'_T = S_T$ and $S'_E > S_E$ or
> (2) $S'_E = S_E$ and $S'_T > S_T$ or
> (3) $S'_E > S_E$ and $S'_T > S_T$.

Unfortunately, Pareto Optimality alone does not help us to identify a unique solution: For example, if DP_{cand} contains only two pairs: $dp_1 = \langle 100, 10 \rangle$ and $dp_2 = \langle 50, 50 \rangle$ then both pairs are Pareto optimal. Making use of a number of other axioms that any fair solution must obey, Nash (1950) argued that only one arbitration plan counts as treating the two constraints evenhandedly. According to the Nash arbitration plan, the optimal solution is simply the solution $\langle S_T, S_E \rangle$ with the highest value for $S_T \times S_E$. For example, if DP_{cand} contains only two solutions with the pairs: $dp_1 = \langle 100, 10 \rangle$ and $dp_2 = \langle 50, 50 \rangle$, Nash arbitration causes dp_2 to win. Although other reasonable arbitration plans are conceivable, Nash's notion of fairness appears to be reasonable in the situations of interest to us. [12]

Searching for optimal solutions Ideally, one would like to guarantee that an optimal solution is always found. The most straightforward way of guaranteeing this is to employ an overgenerate-and-test approach: simply generate all possible dialogue plans (DP_{cand}), then find the one with the highest score (based on combining the satisfaction function for individual constraints through multiplication). Such an approach is not limited to revision-based generation. It could also have been applied if we had decided to keep hold of multiple solutions in the generation process at any other stage in the generation process.

The overgenerate-and-test approach does have one major drawback: it is computationally expensive. Take, for example, our application. Assume an initial dialogue plan consisting of a sequence of adjacency pairs. Some of these pairs will lend themselves to aggregation. Let us denote each set of such adjacency pairs with x_i, and assume that there are k such sets, i.e., $x_1, \ldots, x_k$. Additionally, assume that n adjacency pairs have been marked for emphasis. Now the lower bound on the number of candidate dialogue plans to be generated from the initial dialogue plan is given by:

[12] For an empirical study of several arbitration plans in a another area, see Masthoff (2004).

$$(\prod_{i=1}^{k} B_{|x_i|}) \times 2^n$$

Here, $B_{|x_i|}$ stands for the *Bell* number of the set x_i with cardinality $|x_i|$. It returns the number of possible partitions of the set x_i (i.e., ways to aggregate members of x_i). The formula 2^n represents the number of possible ways that the INSERT operation can deal with adjacency pairs that have been marked for emphasis (e.g., if there is only one pair we have $2^1 = 2$ ways: add a subdialogue expressing emphasis or not).

The formula we give provides only a lower bound since application of revision operators can introduce new opportunities for applying these operators (e.g., if we insert multiple subdialogues these subdialogues themselves become candidates for aggregation).

To get an impression of the running times incurred in situations typical for the NECA system,[13] we have tested the overgenerate-and-test dialogue revision module with eight different inputs that represent a random sample of six initial dialogue plans (DP_1 to DP_6) that were generated with the eShowroom system, and two initial dialogue plans (DP_7 and DP_8) with more extreme values (than any of the dialogues produced by the eShowroom system).

The outcomes are given in Table 2, suggesting that, in situations of typical size,'overgenerate and test' is usually quite feasible. Because we apply revisions at the level of dialogue plans, the computation of the candidates themselves is relatively cheap (e.g., it does not involve potentially expensive tactical and surface generation).

In the long run, it might very well turn out that overgenerate-and-test is not feasible, in particular when further constraints and corresponding revision operators are considered. The point of this paper is, however, not to advocate a particular algorithm for solving this problem. Rather, our aim was to argue for two stages in the design of generation systems which address multiple constraints: 1. specification of optimal solutions independent of specific algorithms and 2. construction of an algorithm for finding or approximating an optimal solution. Once step (1) has been performed, an algorithm can often be found for the problem at hand from the repository of existing search and optimization algorithms.[14]

6 Discussion and Conclusions

We started this paper by introducing the notion of a fully generated *scripted dialogue*, and explaining how it is often essential that certain global and soft

[13] Our dialogue revision module has been implemented in Sicstus 3.12.2. We ran the tests on a Pentium M 1.8 MHz laptop.

[14] See Piwek & Van Deemter (2006) for an overview ranging from genetic algorithms (Mellish et al. 1998, Manurung et al. 2000), generate-and-test (Callaway & Lester, 1997) and Linear Integer Programming (Marciniak & Strube, 2005), to static analysis (Paiva & Evans, 2004;2005). Particularly relevant here is the work of Germann et al. (2001) who explicitly compare *optimal* algorithms for translation with *fast* algorithms and use the former to evaluate the latter.

DP	AP	EMPH	AGGR CAND	GEN CAND	GEN SOL	GEN TIME
DP_1	3	1	2	4	1	0.0s
DP_2	2	2	2	11	1	0.0s
DP_3	4	4	2,2	121	6	0.1s
DP_4	3	1	-	2	2	0.0s
DP_5	5	4	2,2	121	6	0.11s
DP_6	3	1	-	2	2	0.0s
DP_7	6	5	2,2,2	484	7	6.81s
DP_8	6	2	6	1198	2	3.866s

Table 2: Results of running our dialogue revision module on eight different initial dialogue plans. In this table DP stands for Initial Dialogue Plan, AP for the number of Adjacency Pairs in the Initial Dialogue Plan, EMPH for the number of Adjacency Pairs that were marked for Emphasis, AGGR CAND for the cardinality of the sets of adjacency pairs that lend themselves to aggregation, GEN CAND for the total number of Candidate Dialogue Plans that the revision module produced, GEN SOL for the number of solutions that were identified in the set of all Candidates (with the constraints set to TURN = MIN and EMPH = MAX, and finally, GEN TIME for the time it took the revision module to generate all candidates and find the optimal solutions.

constraints on the generated dialogues be obeyed, in order that these dialogues have the desired effect on an audience. The remainder of the paper has proposed a specific way in which global soft constraints on scripted dialogues can be formulated and enforced.

Soft constraints have often played an important role in Natural Language Generation but never, to the best of our knowledge, in connection with scripted dialogue. Moreover, we believe that there is insufficient understanding of the properties of soft constraints, particularly in connection with global properties of the texts (or dialogues) generated. We have therefore devoted a substantial part of this paper to a formal exploration of a number of issues in this area, focusing on the question what types of constraints can be enforced using a *local strategy*. This exploration is largely independent of the subject matter of the constraints, and is as relevant to the generation of text as it is to the generation of scripted dialogue. The main outcome of the theoretical explorations in section 4 is that it is difficult, and in some cases impossible, to guarantee the satisfaction of global constraints by following simple 'local' strategies: in some cases, global constraints really do necessitate the inspection of the generated text (or dialogue) as a whole.

When multiple soft constraints are involved, one of the main problems is, of course, to find a principled way in which to balance these constraints against each other. Although there is a significant amount of work on constraints in generation (see Piwek & Van Deemter, 2006, for a survey), detailed comparisons of alternative ways of combining soft constraints are typically missing, as

we have argued, and it is often unclear what motivates a particular approach to constraints in NLG (see e.g. section 4.4).Taking our inspiration from the literature on decision and game theory – which is finding more and more applications in many areas including linguistics (e.g. Rubinstein 2000) – we have argued that combination strategies should be heavily dependent on the situation in which the constraints are applied, but that it is possible to formulate some general principles to narrow down the options.

Throughout the paper, we have emphasised that it is important that the details of how constraints interact are worked out *before* a concrete algorithm is designed. This approach – which, once again, is applicable far beyond the generation of Scripted Dialogue – is helpful in two ways: it tells us what we would like to achieve, and it helps us to identify the best optimization strategy. These and other issues discussed in this paper are illustrated using a stand-alone extension to the NECA system, which generates dialogues between animated conversational agents, and where a *revision* approach is taken to search for a dialogue that satisfies a set of constraints in the best possible way.

Acknowledgements We would like to thank the reviewers of Research on Language and Computation, and Chris Mellish, Daniel Paiva Richard Power and Graeme Ritchie for their helpful comments on earlier versions of this paper. Needless to say tha any remaining erros are the sole responsibility of the authors.

References

Appelt, D.E. (1985). *Planning English Sentences*, Cambridge University Press, Cambridge.

André, E., Rist, T., van Mulken, M., and S.Baldes (2000). The automated design of believable dialogues for animated presentation teams. In J.Cassell, J.Sullivan, S.Prevost, and E.Churchill (Eds.), *Embodied Conversational Agents*, MIT Press, 220-255.

Bateman, J. (1997), 'Sentence generation and systemic grammar: an introduction'. *Iwanami Lecture Series: Language Sciences*, Iwanami Shoten Publishers, Tokyo.

Bistarelli, S. Montanari, U. & Rossi, F. (1997). 'Semiring-based constraint solving and optimization'. *Journal of the ACM*, **44**(2), 201–236.

Bistarelli, S., Frühwirth, T., Marte, M. & Rossi, F. (2004). 'Soft Constraint Propagation and Solving in Constraint Handling Rules', *Computational Intelligence*, **20**(2), 287–307.

Callaway, C. and J. Lester (1997). 'Dynamically Improving Explanations: a Revision-Based Approach to Explanation Generation'. In: *Proceedings. of IJCAI97 conference*, Nagoya, Japan.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague/Paris.

Clark, H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Cox, R., McKendree, J. Tobin, R., Lee, J. and Mayes, T. (1999). Vicarious learning from dialogue and discourse: A controlled comparison. *Instructional Science*, **27**, 431–458.

Craig, S., Gholson, B., Ventura, M., Graesser, A. and the Tutoring Research Group (2000).Overhearing Dialogues and Monologues in Virtual Tutoring Sessions: Effects on Questioning and Vicarious Learning. *International Journal of Artificial Intelligence in Education*, **11**, 242–253.

Dubois, D., Fargier, H. and Prade, H. (1996). 'Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty'. *Applied Intelligence*, **6**, 287–309.

Germann,U., Jahr, M., Knight, K., Marcu, D. and Yamada, K. (2001). 'Fast Decoding and Optimal Decoding for Machine Translation'. *Proceedings of the 39th Conference of the Association for Computational Linguistics (ACL)*, Toulouse, France, 228–235.

Hayes, J. and L. Flower (1986). 'Writing research and the writer'. *American Psychologist* **41**, 1106–1113.

van Deemter, K. and Odijk, J. (1997). Context Modeling and the Generation of Spoken Discourse. *Speech Communication* **21**, 101–121.

van Hentenryck, P. (1989). *Constraint Satisfaction in Logic Programming*, The MIT Press, Cambridge, Massachusetts.

Hovy, E. (1988). *Generating Natural Language Under Pragmatic Constraints*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Inui, K., Tokunaga, T. and Tanaka, H. (1992). 'Text revision: A model and its implementation'. In R. Dale et al. (eds.), Aspects of Automated Natural Language Generation: Proceedings of the Sixth International Natural Language Generation Workshop, pp. 215–230, Springer-Verlag, Berlin.

Isard, A., Brockmann, C. and Oberlander, J. (2006). 'Individuality and Alignment in Generated Dialogues'. In: *Proceedings of INLG 2006, the International Natural Language Generation Conference*, July 2006, Sydney, Australia.

Kamp, H. & Reyle, U. (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht.

Kibble, R. & R. Power (2000). 'An integrated framework for text planning and pronominalisation'. In: *Proceedings of The First International Natural Language Generation Conference (INLG'2000)*, 77–84.

Kibble, R. & R. Power (2004). 'Optimizing Referential Coherence in Text Generation'. *Computational Linguistics*, **30**(4), 401–416.

Krenn B., H. Pirker, M. Grice, S. Baumann, P. Piwek, K. van Deemter, M. Schröder, M. Klesen, E. Gstrein (2002). 'Generation of multimodal dialogue for net environments', in: Busemann S. (ed.), *KONVENS 2002*, DFKI, Saarbrücken, Germany, 91–98.

Langkilde-Geary, I. (2004). 'An Exploratory Application of Constraint Optimization in Mozart to Probabilistic Natural Language Processing'. *International Workshop on Constraint Solving and Language Processing – CSLP 2004*, September, Roskilde University.

Mann, W. C. (1985). 'An introduction to the Nigel text generation grammar'. In: J. D. Benson & W. S. Greaves (eds.), *Systemic Perspectives on Discourse: Selected Theoretical Papers from the 9th. International Systemic Workshop*, Ablex Pub. Corp., Norwood, N.J., 84-95.

Manurung, H., Ritchie, G. and Thompson, H. (2000). 'Towards a computational model of poetry generation'. In: G.A. Wiggins (ed.), *Proceedings of the AISB00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, SSAISB, 79–86.

Marciniak, T. and Strube, M. (2005). 'Discrete Optimization as an Alternative to Sequential Processing in NLG'. *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG 2005)*, Aberdeen, UK.

Masthoff, J. (2004). 'Group modeling: Selecting a sequence of television items to suit a group of viewers'. *User Modeling and User Adapted Interaction*, **14**, 37–85.

McDonald, D. (1987). 'Natural-Language Generation'. In: S. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, Volume 1.* John Wiley and Sons, New York.

Mellish, C., Knott, A., Oberlander, J. and O'Donnell, M. (1998). 'Experiments using stochastic search for text planning'. *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-lake, Ontario.

Nash, J. (1950). 'The Bargaining Problem'. *Econometrica*, **18**, 155–162.

Paiva, D. and Evans, R. (2004). 'A framework for stylistically controlled generation'. In: A. Belz, R. Evans and P. Piwek (eds.), *Natural Language Generation: Third International Conference (INLG 2004)*, LNCS 3123, Springer, Berlin, 120–129.

Paiva, D. and Evans, R. (2005) 'Empirically-based control of natural language generation', In: K. Knight, H.T. Ng and K. Oflazer (eds.), *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, Ann Arbor, USA, pp. 58-65.

Piwek, P. and K. van Deemter (2002). 'Towards Automated Generation of Scripted Dialogue: Some Time-Honoured Strategies'. In: Bos, J., M. Foster and C. Matheson (Eds.), *Proceedings of EDILOG: 6th workshop on the semantics and pragmatics of dialogue*, Edinburgh, September 4-6, 2002, pp. 141 - 148.

Piwek, P., B. Krenn, M. Schröder, M. Grice, S. Baumann and H. Pirker (2002). 'RRL: A Rich Representation Language for the Description of Agent Behaviour in NECA'. In: *Proceedings of the AAMAS workshop "Embodied conversational agents - let's specify and evaluate them!"*, Bologna, Italy, 16 July 2002.

Piwek, P and van Deemter, K. (2003). 'Dialogue as Discourse: Controlling Global Properties of Scripted Dialogue'. In: *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, AAAI Technical Report SS-03-06. Menlo Park, California: AAAI Press, pp. 118 - 124.

Piwek, P. and van Deemter, K. (2006). 'Constraint-based Natural Language Generation: A Survey'. *Technical Report* 2006/03, Computing Department, The Open University.

Power, R. (1979). 'The Organization of Purposeful Dialogues'. *Linguistics*, **17**, 107–152.

Power, R. (2000). 'Planning texts by constraint satisfaction'. In *Proceedings of COLING 2000*, pp. 642–648, Saarbrücken, Germany.

Reiter, E. (1994). 'Has a consensus NL Generation architecture appeared, and is it psycholinguistically plausible?', In *Proceedings of the Seventh International Workhop on Natural Language Generation*, pp. 95–105, Leiden, The Netherlands.

Reiter, E. (2000). 'Pipelines and Size Constraints'. *Computational Linguistics.* **26**, 251–259.

Robin, J. & K. McKeown (1996). 'Empirically Designing and Evaluating a New Revision-Based Model for Summary Generation', *Artificial Intelligence.* **85**(1–2).

Rosenfeld, A., Hummel, R.A., & Zucker, S.W. (1976). 'Scene labeling by relaxation operations', *IEEE Trans. on Systems, Man and Cybernetics*, **6**, 420–433.

Rubinstein, A. (2000). *Economics and Language*, Cambridge University Press, Cambridge.

Schegloff, E. (1972). 'Notes on a Conversational Practice: Formulating Place'. In: D. Sudnow (ed.), *Studies in Social Interaction*, The Free Press, New York, 75–119.

Traum, D. J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio (1999). 'A model of dialogue moves and information state revision'. TRINDI Project Deliverable D2.1, 1999.

Traum, D. and Larsson, S. (2003). 'The Information State Approach to Dialogue Management'. In: J. van Kuppevelt & R.W. Smith (eds.), *Current and New Directions in Discourse and Dialogue*, Kluwer Academic Publishers, Dordrecht, 325–353.

Waltz, D. (1975). 'Understanding line drawings of scenes with shadows'. In: P.H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 19–92, 1975.