ISSN 1744-1986

The Open University

Technical Report N° 2006/03

Constraint-based Natural Language Generation: A Survey

Paul Piwek
Kees van Deemter

25th March 2006

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

Constraint-based Natural Language Generation: A Survey

Paul Piwek
Centre for Research in Computing
The Open University, UK

Kees van Deemter
Department of Computer Science
University of Aberdeen, UK

March 2006

Abstract

This paper contains a survey of 25 papers that deal with constraint-based approaches to Natural Language Generation (NLG). Amongst other things, we discuss the nature of the constraints that have been modelled in NLG (stylistic, syntactic, etc.), their type (hard versus soft), the units to which the constraints are applied (words, sentences, rhetorical structure, etc.) and the approaches to solving constraint satisfaction problems in NLG. The survey starts with a brief introduction to the constraint-based view of NLG and a section that aims to clarify the many-faceted notion of a constraint. The aim of this survey is to provide the reader with an overview of constraint-based approaches to NLG; it does *not* present an exhaustive list of all the literature in this area. What it does aim for, is to give a reasonably representative sample of the various approaches to constraint-based NLG.

Table of Contents

Section 1

Introduction

The mainstream characterization of Natural Language Generation (NLG) is as addressing the problem of mapping non-linguistic representations to expressions in natural language (e.g., [53]).[1] NLG researchers have dealt with generation from a variety of non-linguistic representations including database entries, formulae in logical calculi and expressions of knowledge representation formalisms. Generation of linguistic expressions of various sizes and types has been investigated, including subsentential units (e.g., noun phrases), individual sentences and multi-sentence discourse. Additionally, some have investigated generation of multimodal output which combines natural language expressions with non-verbal signs such as gestures, graphics, layout, etc. (see, e.g., [55] for an overview of recent work).

In this paper, we survey work on NLG that adopts a specific perspective on the generation. Under this perspective, the emphasis is shifted from NLG as a *mapping problem*, to the problem of the automated production of natural language expressions that satisfy a set of *constraints*. Such constraints can pertain to surface properties of the expressions (e.g., their length), but also to the underlying syntactic properties, and their semantic or even pragmatic content. The mapping problem can now be seen as just one possible constraint amongst many, i.e., a constraint which requires the generated expressions to express no more and no less than some given (semantic) content.

Arguably, the constraint-based perspective is not sufficiently restrictive, since it also brings summarization, paraphrasing and machine translation (MT) into the fold: all of these can be conceived of as involving the production of linguistic expressions under a set of constraints. To keep this survey manageable, we have decided to focus the survey itself on constraint-based NLG that more or less fits the traditional conception of NLG, i.e., which starts with non-linguistic representations.

Before we delve further into constraint-based approaches to NLG, let us first briefly set out the arguments in favour of such approaches. Firstly, as we already

[1] But see [54] for a good overview of alternative approaches to natural language generation.

pointed out, a constraint-based approach liberates NLG from the narrow view of mapping as its primary concern. It acknowledges that verbatim 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 (in particular, settings where entertainment rather than informing is the overarching goal).[2] The idea that language generation is more than simple mapping of information and involves addressing multiple constraints at once is certainly not new. One of the first to express it very concisely and work it out in great detail is Appelt ([3]). Although he speaks of 'goals' rather than 'constraints', his ideas resonate with the approaches we discuss in this survey:

> "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 [3])

Secondly, constraints enable a *declarative* approach to solving generation problems. The main advantage of this is that we 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.

Thirdly, constraints are well-suited for addressing problems that involve *interactions* between information of different types, granularity and levels of specification.

Constraint-based NLG is by no means a monolithic entity. If anything, what this survey shows is that there is variety of approaches and architectures for constraint-based NLG, each with its own advantages and disadvantages.

[2] In many generation systems, content determination is an excellent example of generation which goes beyond straightforward mapping of information: typically content has to be selected based on one or more criteria/constraints. For example, in the DYD system ([58]), which generates brief informative spoken presentations about music recordings, information about the age of a composer is only included if it is noteworthy, e.g., if the composer was very young when s/he composed the music.

Section 2

What is a constraint?

Constraints are part of everyday life. Consider the predicament of someone who wants to organize a birthday party. One constraint which she will need to take into account when inviting people is that party has to *fit into her 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. Another constraint may be that the party has to *be pleasant*. For example, if Mr. X comes to the party then perhaps it would be better if Mrs. Y stayed at home.

The problem for the party organizer is to make sure that the actual party satisfies such a set of constraints. Constraints allow one to differentiate among party situations: some will be more preferred than others and some might even be ruled out altogether.

In this example, we have seen that situations and ways of distinguishing between them are central to the notion of a constraint. In fact, the notion of a constraint is more general: situations are just one type of *object* that constraints can apply to. Other, more tangible, objects include for example artifacts — such as houses, cities, roads, cars, etc. — which are typically designed with a set of constraints in mind.

2.1 Formal definition of the notion of a constraint

Let us now move to a formal characterization of constraints. Our exposition roughly follows the formalizations of the notion of a constraint as given in [7] and [13].

We assume that a universe E of objects is given to which a constraint or set of constraints applies.

> Constraint A constraint c for a universe E is a function from E into a range S of satisfaction levels, i.e., $c : E \mapsto S$.

For classical constraints, $S = \{1, 0\}$ with $0 < 1$. A classical constraint returns 1 for objects which satisfy it and 0 for objects which do not satisfy it. In other words, we have two satisfaction levels: satisfaction and no satisfaction. Combining classical constraints amounts to taking the minimum (min) of the satisfaction levels assigned by the individual constraints. If we write $c_1 \otimes c_2$ for the combination of the classical constraints c_1 and c_2, then for all $e \in E$: $c_1 \otimes c_2(e) = min(c_1(e), c_2(e))$. Thus, an object e only satisfies two or more classical constraints if it satisfies (i.e., returns 1) for each one of them. If one of them returns 0, the minimum of their combination will automatically be 0.

Generally speaking, a Constraint Satisfaction Problem (CSP) consist of a set of constraints $c : E \mapsto S$, given an instantiation of the following three components:

1. A UNIVERSE E. Typically, E is characterized in terms of a sequence of variables $V = X_1, X_2, \ldots, X_n$ that range over non-empty domains $D_1, D_2, \ldots, D_n$. Objects of E are then tuples $\langle x_1, x_2, \ldots, x_n \rangle$ such that $x_1 \in D_1, x_2 \in D_2, \ldots, x_n \in D_n$. Such a tuple contains instantiations for the variables in the sequence V.[1]

2. A structure S to represent SATISFACTION LEVELS. **Classical** constraints can be modelled using a total ordering over the set $\{1, 0\}$, with 0 as the bottom element and 1 as the top element. They require only two levels: not satisfied (0) and satisfied (1). **Soft** or **flexible** constraints allow for intermediate levels. Their satisfaction levels can, for example, be represented by a total ordering on a set that still has 0 as the bottom element and 1 as the top element, but also other elements representing satisfaction levels between 0 and 1. Levels of satisfaction are also known in the literature as *measures of desirability* or *levels of feasibility*, *importance* or *preference*. They can be related as ratio variables (so that it makes sense to say that one level is twice as high as another) or mereley as ordinal variables (where one can only say that one level is higher than another).

3. A COMBINATION OPERATION $\otimes$ for combining multiple constraints into a single constraint. The satisfaction level that is assigned by a composite constraint is normally a function of the satisfaction levels assigned by individual constraints. We have already seen that min is used for classical constraints. Other functions are, for example, *multiplication* and *weighted addition*. Note that for weighted addition, 0 is typically interpreted as full satisfaction. Whereas any elements above zero are interpreted as representing *costs*.

[1] Often, in the formulation of CSPs sets of variables and assignments of values to them are used instead of sequences of variables and tuples. Here we chose for the latter formulation, because we want our universe E to be a set of objects. Tuples qualify as complex objects, whereas intuitively assignments are not really objects (they are functions). In most of the remainder of this section, the exact shape and form of the members of E is not relevant. All we need is the assumption that there is a set of objects E to which the constraints are applied.

A solution to a CSP is an object $e \in E$ such that there are no other objects in E with a higher satisfaction level than e and the satisfaction level of e is not total non-satisfaction (e.g., when the level of satisfaction is 0 for a classical hard constraint). Note that e does not need to be unique: a CSP can have multiple solutions.

Different constraint satisfaction problems can be obtained by varying the components listed above, for example (based on [4]):

- Fuzzy Constraint Satisfaction Problem 1. Variables ranging over finite domains. 2. Satisfaction levels are represented as real numbers between 0 and 1. 3. Constraints are combined using *min* ($c_1 \otimes c_2(e) = min(c_1(e), c_2(e))$).
- Weighted Constraint Satisfaction Problem 1. Variables ranging over finite domains. 2. Satisfaction levels are represented by non-negative real numbers. 0 is identified with maximal satisfaction. The higher the value, the lower the satisfaction level. In other words, the non-negative real numbers represent *costs*. 3. Constraints are combined using addition ($c_1 \otimes c_2(e) = c_1(e) + c_2(e)$).

The notion of a constraint that we have adopted here follows [7] and [13]. We decided to adopt this approach, because it brings out very clearly the relationship between optimization and satisfaction of hard constraints: satisfaction of hard constraints is basically a special case of optimization (which covers both hard and soft constraints).

An older and more restrictive notion of constraints can be found in, for example, pages 3–4 of [23]: "Formally, a CSP can be defined in the following way. Assume the existence of a finite set I of variables $\{X_1, X_2, \ldots, X_n\}$, which take respectively their values from their finite domains $D_1, D_2, \ldots, D_n$ and a set of constraints. A constraint $c(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$ between k variables from I is a subset of the Cartesian product $D_{i_1} \times D_{i_2} \times D_{i_k}$, which specifies which values of the variables are compatible with each other. In practice this subset does not need to be given explicitly, but can be defined by equations, inequalities, or programs whatsoever. A solution to a CSP is an assignment of values to all variables, which satisfies all the constraints. The task is to find one or all solutions."

According to this definition, optimization is not included by CSP. CSP is limited to satisfaction of hard constraints, i.e., constraints that are either satisfied or not. We have chosen for a broader definition of CSP (where classical hard CSP are a special case of optimization). This more general definition of CSP reflects better the common theme that underlies the constraint-based approaches that we discuss in the remainder of this survey.

2.2 Hard constraints and priorities

We have seen that we can view a constraint as imposing an ordering in terms of satisfaction levels on the universe of objects E. By allowing constraints to

assign satisfaction levels different from 0 and 1, it is possible to model situations in which we need to find a solution that satisfies the constraints to the highest degree possible, but not perfectly. Thus we can deal with situations where a perfect solution does not exist and a choice among less than perfect alternatives needs to be made. Thus we avoid the problem of classical hard constraints, i.e., that objects which do not fit all the constraints perfectly are not differentiated; they are all treated as equally bad.

There are also other ways to circumvent the crispness of classical constraints. For example, if not all hard constraints can be satisfied, one could simply attempt to find the solutions that satisfy the maximum number of hard constraints. Taking this a bit further, one can assign an ordering to constraints, indicating which constraints are more important in some respect than others. The 'priority' of a constraint can then influence whether we allow it to be dropped or not. Thus we obtain the notion of an ordered set of constraints:

> A set of ORDERED CONSTRAINTS consists of a set of classical hard constraints C and a function π from C to levels of priority P (that is $i : C \mapsto P$).

There are alternative formalizations for priority levels P and also for the way the priority level of a constraint influences whether it can be dropped. Two important instantiations of ordered constraints — optimality theoretic ([51]) and probabilistic — can be characterized as follows:

- OPTIMALITY THEORY Each constraint $c \in C$ is assigned a natural number representing its position in the priority ranking $\pi(c)$ (lower numbers indicating higher priority). Roughly speaking (i.e., disregarding ties) $e \in E$ is a solution if and only if: $\exists c(\forall c'.\pi(c') \leq \pi(c) \rightarrow c'(e) = 1$ and $\forall e'.e' \neq e \rightarrow \exists c''.\pi(c'') \leq \pi(c)$ & $c''(e') = 0)$. Thus, solutions are those $e \in E$ which satisfy all constraints down to a level of importance which is lower or equal to that for all other $e' \in E$. Perhaps most striking here is that *numbers* of constraint violations are not taken into account (except when there is a tie between two would-be solutions).

- PROBABILISTIC CONSTRAINT SATISFACTION PROBLEMS (description following [4]) We use π to assign to each constraint $c \in C$ its probability $\pi(c) = p$ independent from the probability of the other constraints. The probability is defined as the probability of the class of situations that the constraint characterizes.

  One way to characterize solutions of Probabilistic CSP is in terms of a comparison of the probability of the sets of constraints that are satisfied. In particular, for a given situation, we calculate the probability of all the sets of constraints which the situation satisfies. The probabilities for each of these sets is obtained by multiplying the probabilities of the constraints in the set. We then sum the resulting probabilities. The situation that receives the highest sum is a solution. More precisely, $e \in E$ is a solution

to a Probabilistic CSP with constraints C and $\pi : C \mapsto P$ if and only if for all $e' \in E$: $\sum\{\prod\{\pi(c)|c \in S\} \mid S \subseteq C$ such that $\forall c \in S : c(e) = 1\} \geq \sum\{\prod\{\pi(c)|c \in S\} \mid S \subseteq C$ such that $\forall c \in S : c(e') = 1\}$.[2]

2.3 Solving Constraint Satisfaction Problems

Let us conclude this section with some pointers to implementations of solvers for classical and non-classical Constraint Satisfaction Problems.

- Classical CSP: most contemporary ProLog implementations include a library for solving CSPs. There are also C++ and Java based libraries. See, e.g., [2] for a list of resources.
- OTSoft for Optimality Theory ([21]).
- Problems involving soft constraints are a form of optimization problems. A survey of optimization methods can be found in [19]. A particularly relevant field is linear Programming (LP). LP addresses optimization problems in which the objective function and the constraints are linear ([1]).[3] There exist various (integer) linear programming packages and libraries ([46]).

[2]Given function f and property r, we use the notation $\sum\{f(x) \mid r(x)\}$ to denote the sum of all $f(x)$ such that $r(x)$*holds*. The notation for products ($\prod$) works along the same lines.

[3]A linear programming problem consists of 1. a linear function (e.g., $c_1x + c_2y$) that is to be maximized, 2. a set of constraints, e.g., $g_1x + h_1y \leq a_1$ and $g_2x + h_2y \leq a_2$., and 3. the requirement that the variables (x and y) have non-negative values. If the variables are integers, we speak of an Integer Linear Programming (ILP) problem and if only some of them need to be integers we have a Mixed Integer Programming (MIP) problem.

Section 3

Constraints in NLG: Approaches and Architectures

In this section, we summarize the findings of our survey on constraint-based NLG. A one-by-one description of the papers that we collected for our survey can found in the next section. The various classifications that we provide are neither exhaustive nor are the categories always mutually exclusive. Rather, what we aim to provide is a quick overview of the different approaches to constraint-based NLG together with pointers to the relevant literature.

3.1 Nature of constraints

Morphological [45]

Lexical Choice [15, 24, 36, 44, 59]

Syntactic [6, 24, 25, 10, 30, 36, 39, 44, 45, 43]

Syntactic (aggregation) [9]

Semantic [10, 12, 57, 16, 39, 45, 43]

Semantic Ambiguity [26]

Punctuation [59]

Discourse [28, 36, 38, 49, 59]

Affect/Opinion [24, 25]

Style [24, 25, 26, 35, 34, 47, 48]

Size [52, 48]

Statistical/Probabilistic/Corpus-based [31, 30, 29, 36, 10, 47, 59]

3.2 Units to which constraints apply

Phonetic [35, 34]

Punctuation markers [59]

Words [24, 10, 30, 36, 44, 59]

Mappings from conceptual input to lexical items [15]

Phrases [24, 44, 42, 43]

(Content of) Noun Phrases/Referring expressions [16, 12]

Clauses [9]

Sentences [10, 29, 39, 45, 42, 43]

Syntactic Surface Structure [36, 26]

Trees [6]

Paragraphs [24]

Discourse: Text Structures/RST Trees [28, 36, 38, 49]

Entire text [47, 52]

Dialogue script plan [48]

3.3 Types of constraints

Hard [6, 9, 12, 15, 16, 39, 44, 42, 43]

Soft [10, 57, 29, 35, 34, 38, 47, 48]

Hard and Soft [52, 59, 24, 28, 30, 36, 49, 45]

3.4 Approaches for dealing with multiple soft constraints

Here we list those papers in which the problem of combining multiple soft constraints is addressed. Note that some of the papers dealing with soft constraints listed above only deal with a single constraint. There were also some papers that do not explicitly state how multiple soft constraints are combined. Below we provide the means by which the scores assigned by different constraints to potential solutions are combined (addition versus multiplication) or dealt with in an incremental approach (always satisfy the constraint that is currently least satisfied):

Least satisfied precedence [24]

Weighted addition [28, 10, 36, 38, 49]

Multiplication [48]

Bidirectional Superoptimality [57] (based on [8])

3.5 Approaches to satisfying constraints

In this section we organize the papers covered by this survey using a taxonomy. The taxonomy systematizes a variety of approaches for satisfying constraints in NLG. The main division is between approaches that guarantee an optimal solution with respect to the set of constraints that are in force, and those that can only approximate such solutions. Of course, usually optimization has to be traded in for efficiency. For example, [38] describe their stochastic search approaches specifically to address the problem that 'Constraint satisfaction in general is intractable, and having weighted constraints seems to make matters worse.'. Connectionist approaches are missing from our survey. In recent years they seem to have had few followers in the NLG community. We point this out because some constraint-based approaches specifically refer to connectionist models as implementations (Optimality Theory; see [51]).

- Optimal Solutions
  - **Produce a set of potential solutions and rank these** A set of potential solutions P (or a compact representation thereof) is generated and these are then ranked (sometimes only the first stage is implemented). This allows us to guarantee that we find the optimal solution relative to P, because we can compare all members of P with each other. Thus we have two stages: 1. generation of a set of (potential) solutions and 2. ranking. In stage 1., we either apply *hard constraints* which all solutions will need to satisfy using CSP satisfaction ([12, 16, 28, 49, 59]) or we use traditional symbolic generation, rewriting or revision rules to obtain a set of potential

solutions ([29, 30, 48]). Typically, there are pre- and post-processing steps which tailor the input and output of this stage (see [49] on the problems of directly applying CSP satisfaction to configuration tasks). In stage 2., the set of potential solutions that has been produced in stage 1 is ranked according to one or more soft constraints. Sometimes, step 2 is omitted altogether and solutions are only subject to a set of hard constraints (e.g., [6, 39]).

- **Integer Linear Programming (ILP)** In ILP, hard constraints are directly integrated with a soft constraints that are expressed in terms of a linear function which needs to be maximized/minimized (see section 2.3). This approach has been adopted in [36]. See also [18] for an application of Linear Programming to Machine Translation.

- Search using Heuristics

  - **Building a solution** A solution is *constructed* stepwise. The construction can be carried out and controlled in a number of different ways:

    * Incremental using monitoring regarding the extent to which various constraints are being satisfied ([24, 25]).
    * Construction using a unification-based grammar (e.g., [15, 44]).
    * Construction as a sequence of decisions, where alternatives for each decision problem are ranked based on scores which have been obtained during off-line (corpus-based) training ([47]).

  - **Searching complete states/revision** Here we distinguish between searching through a sequence of single potential solutions and searching through a sequence of sets of potential solutions. By a complete state, we mean a complete representation of the expression that is to be generated on a certain level of abstraction (e.g., this can be the discourse plan level or the level of lexical realizations, or both).

    * *Single solutions* Beginning from of a single complete representation of a possible solution, we apply revisions sequentially to this representation to improve it.
      · Applying revision operations triggered by their pre-conditions ([9, 26, 45]).
      · Estimation: we make local decisions regarding which revision operation to apply based on estimations of their global effects ([52]). Estimation is required if the representation is complete on a certain level (i.e., discourse plan), but not on levels which it dominates, e.g., lexical realization. For example, a discourse plan allows us to *estimate* the length of the final document, but the actual length will only be known after it has been lexically realized.

- Full expansion: revisions operate on a representation of the actual text such that we do not need estimates to choose between revision operations that are sensitive to details of the actual realization ([52]).

* *Multiple solutions* We maintain a set of potential solutions and revise members of this set. For this purpose, stochastic algorithms can be used (e.g., genetic algorithms) to minimize the chance of getting stuck in local maxima ([38, 35, 34]).

Section 4

Papers on Constraints in NLG

In this section, we describe each of the papers that we collected for our survey in terms of a list of 9 questions:

1. Name of the constraints (if available).
2. Brief informal description of the constraints.
3. Type of constraints: Hard, soft (e.g., Preferences), or other (specify).
4. Units to which the constraints apply: e.g., sentences, words, entire texts, dialogue turns, or other (specify).
5. Nature of the constraints: syntactic, stylistic, size, or other (specify).
6. Theoretical work or implemented: Implemented/Partially Implemented/Not Implemented
7. Approach to addressing constraints in generation: As a classical constraint satisfaction problem, or other (specify, e.g., revision-based, monitoring, overgenerate-and-test, ...).
8. Single or many constraints? How are constraints combined?
9. Any other comments on the paper.

For some of the papers that are surveyed below, we obtained answers to these questions from the authors of the papers. We have used these answers from the authors verbatim. For all papers, were we used the authors' answers, this is stated explicitly at the beginning of list of answers for the paper. Everywhere else, the answers were filled out by ourselves after careful study of the papers in question.

The heading for each paper contains the author names, references to the bibliography at the end of this paper, year of main publication and title of main publication.

4.1 Becker and Lockelt [6], 2000, 'Liliput: A Parameterisable Finite-Domain Constraint Solving Framework and Its Evaluation with Natural Language Generation Problems'

1. *Name of the constraint(s) (if available)* –
2. *Brief informal description of the constraint(s)* This paper describes the Liliput framework for solving finite-domain hard constraints, and its application to syntactic realization and microplanning.
3. *Type of constraint(s)* Hard.
4. *Units to which the constraint applies* Tree structures, feature structures, semantic representations and syntactic dependency structures.
5. *Nature of constraint(s)* Syntactic.
6. *Theoretical work or implemented* Implemented in Common LISP.
7. *Approach to addressing constraints in generation* Propose algorithm for solving finite-domain hard constraints.
8. *Single or many constraints? How are constraints combined?* –
9. *Any other comments on the paper* Beale et al. [5] is a precursor to work along these lines. They apply a special purpose constraint-directed control architecture called *Hunter-Gatherer* (HG) to sentence planning.

4.2 Callaway and Lester [9], 1997, 'Dynamically Improving Explanations: A Revision-Based Approach to Explanation Generation'

1. *Name of the constraint(s) (if available)* –
2. *Brief informal description of the constraint(s)* Constraints are used to reign in the application of revision operators for clause aggregation. For example, focus constraints are used to prevent inappropriate aggregation of adjacent clauses. Discourse constraints can prevent reorganization of discourse plans if the reorganization crosses boundaries (in the organizational structure of the discourse).

3. *Type of constraint(s)* Hard.

4. *Units to which the constraint applies* Clauses.

5. *Nature of constraint(s)* For controlling clause aggregation revisions.

6. *Theoretical work or implemented* Implemented system called Revisor.

7. *Approach to addressing constraints in generation* (1) The system produces an initial draft of a text. (2) Details that are irrelevant for clause aggregation are removed, to obtain an abstract representation of the discourse. (3) A depth-first exploration of the abstract discourse space is undertaken, through the iterated application of revision operators (for aggregation and permutation of clauses). Operators are tried if their preconditions are satisfied. The result of the application is tested for the satisfaction of discourse, global, and focus constraints. If one of these is violated an alternative operator is chosen, or the system backtracks. The process continues until an acceptable revision has been obtained or all possible revisions have been exhausted (the notion of an acceptable revision is not defined in this paper).

8. *Single or many constraints? How are constraints combined?* See above; constraints are used as guards on revision operations rather than goals in themselves.

9. *Any other comments on the paper* The approach is compared with how human writers produce text. They refer to [22], who hypothesize that human writers produce multiple drafts to cope with the complexity of writing text.

4.3 Dale & Haddock [12], 1991, 'Content Determination in the Generation of Referring Expressions'

1. *Name of the constraints (if available)* –

2. *Brief informal description of the constraints* In this approach to the generation of relational referring expressions, constraints derive from the properties and relations ascribed to objects in a referring expression. For example, the expression 'the cup on the table' corresponds with the constraints cup(x), on(x,y), table(y), together with a bookkeeping of the remaining values that each variable can take on the basis of these constraints. For example, [x=c1, y=b1] expresses that x and y both have a unique value. Crucially, this uniqueness arises from an inferential combination of the two constraints, which is something that a simpler approach to GRE would not be able to do.

3. *Type of constraints* Hard.

4. *Units to which the constraints apply* Properties and relations expressed in a referring expression.

5. *Nature of the constraints* Semantic (i.e., logical).

6. *Theoretical work or implemented* Implemented (e.g., in the EPICURE system, see [11]).

7. *Approach to addressing constraints in generation* Classical constraint satisfaction.

8. *Single or many constraints? How are constraints combined?* Any finite number of constraints (as expressed in a referring expression). Dale and Haddock use A.K. Mackworth's ([32]) notion of consistency in constraint networks (also applied in Mellish [37]).

9. *Any other comments on the paper* Essentially the same approach was later used for the generation of (distributive or collective) references to sets by Matthew Stone ([56]). Gardent's ([16]) use of constraints in generation of referring expressions (GRE) is different because there, the constraints correspond with general GRE principles.

4.4 van Deemter [57], 2004, 'Towards a Probabilistic Version of Bidirectional OT Syntax and Semantics'

1. *Name of the constraint(s) (if available)* One constraint is called (degree of) fluency or brevity; the other is called (vicious) ambiguity.

2. *Brief informal description of the constraint(s)* Fluency can be formalised in different ways; the simplest way is to define the most fluent formulation to be the shortest. Ambiguity revolves around the question whether the intended interpretation of the sentence is "by far" the most plausible of all its interpretations. If this is not the case then the sentence is viciously ambiguous. The paper explores how vicious ambiguity might be measured by making use of statistical parsing.

3. *Type of constraint(s)* Both fluency and ambiguity are soft constraints, in the sense that a generator should try to find the best balance between the two when the most fluent formulation happens to be viciously ambiguous.

4. *Units to which the constraint applies* In principle, these constraints can apply to any of these levels.

5. *Nature of constraint(s)* The constraints are relevant for selecting the 'best' among a number of possible (i.e., grammatically correct) formulations. Fluency can be viewed as a stylistic constraint (possibly involving the length of the generated sentence). Ambiguity is a semantic constraint.

6. *Theoretical work or implemented* Theoretical. Efforts to implement a simple version of the idea (involving either lexical choice or referring expressions generation) are at an early stage.

7. *Approach to addressing constraints in generation* Overgenerate-and-test. The starting point of the paper is bidirectional superoptimality (Blutner-style, [8]), but various changes to this concept are necessary.

8. *Single or many constraints? How are constraints combined?* If a formulation exists that is optimal for both constraints then this formulation is chosen. If not then a formulation may be chosen that is optimally fluent but viciously ambiguous (i.e., nice but unclear), or one that is neither optimally fluent nor viciously ambiguous (i.e., ugly but clear).

9. *Any other comments on the paper* This paper asks how bidirectional Optimality Theory can be applied to NLP, focussing primarily on the question of avoidance of ambiguity in NLG (as earlier addressed by e.g., [26] and [45]). It is a programmatic paper in which a number of possible approaches are discussed and compared. The general outlook aspires to be neutral between generation and interpretation.

4.5 Elhadad, McKeown and Robin [15], 1997, 'Floating Constraints in Lexical Choice'

1. *Name of the constraint(s) (if available)* –

2. *Brief informal description of the constraint(s)* Floating constraints are addressed. These are basically parts of the conceptual input structure to a generator for which there is no straightforward mapping to lexical items. Rather, they can appear at various places in the resulting linguistic structure. Examples for the constraints time and manner are given on page 197 of [15]:

   (a) Wall street indexes *opened* **strongly**. (*time* in verb, **manner** as adverb)

   (b) Stock indexes *surged* **at the start of the trading day**. (*time* as PP, **manner** in verb)

3. *Type of constraint(s)* Hard.

4. *Units to which the constraint applies* Mappings from conceptual input to lexical items.

5. *Nature of constraint(s)* Constraints on lexical choice.

6. *Theoretical work or implemented* Implemented in ADVISOR-II system using the FUF/SURGE package ([14]).

7. *Approach to addressing constraints in generation* FUF ([14]) Unification based top-down control regime with dependency-directed mechanism to make processing of floating constraints efficient.

8. *Single or many constraints? How are constraints combined?* Multiple constraints. See above.

9. *Any other comments on the paper* –

4.6 Gardent [16], 2002, 'Generating Minimal Definite Descriptions'

1. *Name of the constraints (if available)* No names provided. The algorithm focuses on the content determination aspect of Generation of referring Expressions. The algorithm builds a description (conceived as a set of positive and negative properties) of a target set. Here we concentrate on the simple case where disjunctions (as in 'the cat and the dog') are not used.

2. *Brief informal description of the constraints* The first constraint (1) says that the positive properties in the generated description must be true of all the elements of the target set. The second constraint (2) says that the negative properties in de description must be false of all the elements of the target set. The third constraint (3) says that every distractor (i.e., every domain element not in the target set) must be excluded by one of the (positive or negative) properties in the description. (This is what makes the description *distinguishing*.)

3. *Type of constraints* Hard.

4. *Units to which the constraints apply* Noun Phrases. More precisely (because the program focusses on the content of the NP): sets of properties to be expressed in a referring expression.

5. *Nature of the constraints* Semantic.

6. *Theoretical work or implemented* Implemented using the generator InDiGen ([17]) which uses the concurrent constraint programming language Oz.

7. *Approach to addressing constraints in generation* Classical constraint satisfaction.

8. *Single or many constraints? How are constraints combined?* Three constraints, which must all be fulfilled. The system (basically by using a version of Dale's Full Brevity algorithm) finds the shortest distinguishing description if a distinguishing description of the target set exists.

9. Any other comments on the paper. –

4.7 Hovy [24, 25], 1988, 'Generating Natural Language under Pragmatic Constraints'

1. *Name of the constraint(s) (if available)* –

2. *Brief informal description of the constraint(s)* Hovy proposes that there are three levels at which constraints operate in generation. At the top level there are *interpersonal communicative goals* and other *pragmatic aspects of the situation*. [25] provides the following example (on page 156):

   - Time: *some*.
   - Tone of interaction: *informal*.
   - Speaker's opinion's: *neutral*.
   - Depth of acquaintance: *strangers*.
   - Goal to affect hearer's opinion's: *none*.

   At the bottom level there are *syntactic decisions* which the generator has to take. He proposes to bring the two together through an intermediate level of *rhetorical goals*. He groups the latter into two categories: *rhetorical goals of opinion* that are achieved through topic collection, grouping and the use of marked and slanted phrases and words, and *rhetorical goals of style* including formality, simplicity, haste, etc.

3. *Type of constraint(s)* Hard and soft.

4. *Units to which the constraint applies* Paragraphs, phrases, words.

5. *Nature of constraint(s)* Affect/opinion, style, syntax and lexical.

6. *Theoretical work or implemented* Implemented as the PAULINE system in LISP.

7. *Approach to addressing constraints in generation* Hovy advocates a planning approach to generation. He distinguishes between prescriptive and restrictive planning. 'Prescriptive strategies are formative: they control the construction and placement of parts in the paragraph and the sentence; that is they make some commitment to the final form of the text [...] Restrictive strategies are selective: they decide among alternatives that were left open (such as, for example, the possibility of including additional

topics under certain conditions, or the specific form of each sentence. A restrictive planner cannot simply plan *for*, it is constrained to plan *with*: the options it has to select from are presented to it by the realizer)' ([25] p. 167). Hovy focuses on how to deal with restrictive planning. He proposes a (execution) monitoring approach. It continually keeps track of the satisfaction status of the various restrictive goals. When making decisions, for each option it calculates which goals are advanced by each option and which are not. A heuristic is used to compute the relative priority of the goals. Hovy points out that there exist various alternative heuristics ([25] on page 168):

(a) Prefer common subgoals of various goals;

(b) Prefer goals that are easier to achieve/cheaper;

(c) Prefer goals that most clearly indicate their long-term promise;

(d) Prefer satisfaction of goals that are least satisfied (so far);

(e) Prefer goals that have been satisfied the longest time ago;

(f) A combination of the latter two strategies.

Hovy opts for strategy 4. The system keeps track of the number of times each goal was satisfied by the choice for a particular option. In case there are multiple conflicting options, it goes for the option that addresses the least satisfied goal. Hovy gives various examples of conflicting goals including the following:

> Rhetorical goal: simplicity = high ⇒ don't passivize.
> Rhetorical goal: partiality = high ⇒ suppress contentious parts, e.g., through passivization.

8. *Single or many constraints? How are constraints combined?* Many constraints/goals. See above.

9. *Any other comments on the paper* –

4.8 Inui, Tokunaga and Tanaka [26], 1992, 'Text Revision: A Model and Its Implementation'

1. *Name of the constraint(s) (if available)* –

2. *Brief informal description of the constraint(s)* A revision-based approach to enforcing constraints on structural ambiguity and sentence complexity is described. Sentence complexity includes lower and upper bounds for sentence length, depth of clause embeddings, depth of modification relation in NP and depth of center embedding.

3. *Type of constraint(s)* Soft: it seems possible to drop constraints if it is not possible to satisfy them. The paper, does, however, not specify in detail when a constraint is given up apart from saying that 'Our model repeats the revision cycle until we produce an acceptable text' (p.7 of [26]).

4. *Units to which the constraint applies* Syntactic surface structure of sentences.

5. *Nature of constraint(s)* Semantic (ambiguity) and stylistic (sentence complexity).

6. *Theoretical work or implemented* Implemented in the WEIVER system.

7. *Approach to addressing constraints in generation* See below.

8. *Single or many constraints? How are constraints combined?* Initial generation followed by a *cycle* of evaluation, revision planning, surface generation of changes, etc. The evaluator checks whether any constraints have been violated. If so, the revision planner tries to find changes which address these violations. A record is kept of previous revisions and dependencies to avoid infinite loops.

9. *Any other comments on the paper* –

4.9 Kibble and Power [28], 2004, 'Optimizing Referential Coherence in Text'

1. *Name of the constraint(s) (if available)* Constraints are reformulations of well-known rules from Centering Theory ([20]). The named constraints are: cohesion, salience, cheapness and continuity.

2. *Brief informal description of the constraint(s)* Centering theory is used to inform generation of coherent text and choice of referring expressions.

3. *Type of constraint(s)* Hard and Soft.

4. *Units to which the constraint applies* Text structures.

5. *Nature of constraint(s)* Discourse level (Centering Theory).

6. *Theoretical work or implemented* Implemented as an extension of the ICONOCLAST text planner ([49]; [50]).

7. *Approach to addressing constraints in generation* The text planner, described in [49], generates a number of text structures. For each text structure alternative realizations in terms of Centering Theory are computed. The realizations vary primarily in the realization of relations (e.g., 'The FDA approves Elixir' versus 'Elixir is approved by the FDA'). This in turn determines subject and object positions for referring expressions. There

is one hard centering constraint which all realizations need to satisfy: The backward-looking centre of an utterance should be the most salient forward-looking centre of the preceding sentence. In Centering Theory, subjects are more salient than objects and the backward-looking centre is the entity that is most likely to be pronominalized. Soft constraints regarding Centering (there are also other soft text planning constraints, see [49]) are implemented in terms of a cost function which assigns to each realization its cost, depending on the (weighted) number of violations of various constraints (e.g., any utterance with no backward looking centre, except for the first utterance, is penalized). Weighted costs are summed and realizations are ordered in terms of their summed weighted costs.

8. *Single or many constraints? How are constraints combined?* Many constraints. See above.

9. *Any other comments on the paper* –

4.10 Langkilde-Geary [30], 2004, 'An Exploratory Application of Constraint Optimization in Mozart to Probabilistic Natural Language Processing'

For this paper ([30]), answers to our questions were provided by the author (Irene Langkilde-Geary). Our comments have been added in square brackets.

1. *Name of the constraint(s) (if available)* None.

2. *Brief informal description of the constraint(s)* –

3. *Type of constraint(s)* Both (hard and soft constraints).

4. *Units to which the constraint applies* Words. Actually, to sets of features associated with individual words.

5. *Nature of constraint(s)* Syntactic.

6. *Theoretical work or implemented* Implemented. [Uses the Mozart/Oz programming language environment which has built-in Concurrent Constraint Programming (CCP)]

7. *Approach to addressing constraints in generation* CSP.

8. *Single or many constraints? How are constraints combined?* They are combined in the framework of concurrent constraint programming, and all constraints must be satisfied. The probabilistic constraints are used as part of a cost function, where the program optimizes for the best cost sentence given the input and the constraints. [The hard constraints are

on domains for representing word features, and on relationships between features of words, e.g., regulating the position of heads.]

9. *Any other comments on the paper* None.

4.11 Langkilde and Knight [29, 31], 1998, 'Generation that Exploits Corpus-based Statistical Knowledge'

1. *Name of the constraint(s) (if available)* –
2. *Brief informal description of the constraint(s)* See below.
3. *Type of constraint(s)* Soft/preferences.
4. *Units to which the constraint applies* Sentences.
5. *Nature of constraint(s)* n-gram language model.
6. *Theoretical work or implemented* Implemented in Nitrogen and its successor HALogen.
7. *Approach to addressing constraints in generation* Input (typically semantically underspecified) is mapped (using symolic rules) to a forest of possible expressions. The N most likely outputs are obtained from this using an n-gram word frequency model.
8. *Single or many constraints? How are constraints combined?* Single soft constraint.
9. *Any other comments on the paper*

4.12 Manurung, Ritchie and Thompson [35, 34], 2000, 'Towards a Computational Model of Poetry Generation'

1. *Name of the constraint(s) (if available)* –
2. *Brief informal description of the constraint(s)* The application is poetry generation. One constraint is specified as a target phonetic form of a particular metrical configuration. The other constraint is specified in terms of the input semantics.
3. *Type of constraint(s)* Soft.

4. *Units to which the constraint applies* The implemented constraint is phonetic (metre), but other constraints are also described including syntactic (rhyme, alliteration), and semantic ones.

5. *Nature of constraint(s)* Implemented constraint concerns metre.

6. *Theoretical work or implemented* Significant part implemented in Java.

7. *Approach to addressing constraints in generation* As a state search problem. A state is a text with its underlying representation and a move can occur on all levels. Search is executed by stochastic hillclimbing. An evolutionary algorithm is used. It starts with initialization: a number of texts are produced that minimally realize the input semantics. The input also contains a target phonetic form. Next, the following two phases are repeated several times: evaluation and evolution of an ordered set of candidate solutions. The approach is similar to [38]. For *evaluation* of phonetics and semantics a score-relative-to-target approach has been implemented. An alternative approach is also discussed that keeps a tally of the times a feature is encountered. Evaluation functions can be parameterized such that users can set coefficients and assign weights. For *evolution*, they suggest three (non-monotonic) operators: add, delete and change. These operate on LTAG ([27]) derivation trees with a flat semantics. Only candidates that had a high score according to the evaluation function are mutated (in fact, they spawn identical children to which the operators are applied). The resulting individuals replace lower scoring individuals.

8. *Single or many constraints? How are constraints combined?* At the time the paper was written only an evaluator for rhythm had been implemented. The semantics is always subsumed by the target semantics and no evaluator existed yet at the time the paper was written.

9. *Any other comments on the paper* –

4.13 Marciniak & Strube [36], 2005, 'Beyond the Pipeline: Discrete Optimization in NLP'

1. *Name of the constraint(s) (if available)* Not applicable. The constraints are mainly non-linguistic. They are used to formalize the problem of aggregating the outcomes of a set of classifiers in terms of integer linear programming (ILP; e.g., [41]). For example, there is a constraint which requires that each classifier assigns exactly one label. The constraints are on the variables of the ILP target function. This function itself can be seen as a soft constraint. Marciniak and Strube's target function minimizes cost of label assignment of individual labels to individual classification tasks and pairs of labels to related classification tasks. Costs of assignment are

computed from the probability distribution of individual labels and prior joint probability of pairs of labels in the annotated corpus.

2. *Brief informal description of the constraint(s)* The application concerns planning of route instructions. The input is a vector of discourse unit meanings, in terms of semantic frame and aspectual category. There is a set of classifiers which is applied to this input to realize a vector representation of the form of the output text. The classifiers include ones for ordering discourse units, choice of discourse connective, syntactic form of the sentence realizing a discourse unit, verb lexicalization of the main verb realizing a discourse unit, etc.

3. *Type of constraint(s)* The main soft constraint is realized as the ILP target function. Further hard constraints characterize classification tasks in general .

4. *Units to which the constraint applies* Discourse level, syntactic structure and lexical items.

5. *Nature of constraint(s)* Soft and hard.

6. *Theoretical work or implemented* Implemented and evaluated system (evaluation consists of comparison with pipeline systems). To solve ILP problems *lp_solve* is used.[1] For constructing the classifiers a the Naive Bayes algorithm was used as implemented in the Weka machine learning software package ([60]).

7. *Approach to addressing constraints in generation* Use of Integer Linear Programming (ILP).

8. *Single or many constraints? How are constraints combined?* Multiple constraints dealt with through ILP. The main linguistic constraint is formulated as the target function. This function operates on the costs for assignments of labels to individual objects and pairs of labels for related objects. Overall Cost is calculated through summation of the cost for tasks and pairs of related tasks. The cost of assigning label l is defined as $-log_2(p(l))$, where $p(l)$ is the probability that label l is selected (from some given task). For pairs of tasks $-log_2$ is taken of the prior joint probability of a pair of labels from the pair of tasks.

9. *Any other comments on the paper* –

4.14 Mellish, Knott, Oberlander and O'Donnell [38], 1998, 'Experiments Using Stochastic Search for Text Planning'

1. *Name of the constraint(s) (if available)* –

[1] http://www.geocities.com/lpsolve/

2. *Brief informal description of the constraint(s)* The goals is build the "best" legal Rhetorical Structure Theory (RST) tree ([33]) for a set of facts and relations between these facts (not all relations need to be included in the tree). Constraints are formalized as scores that are assigned to based on the properties of RST trees. For instance, -10 for trees whose top nucleus does not mention the subject of the text, and -4 for each fact that textually separates a satellite and its nucleus.

3. *Type of constraint(s)* Soft.

4. *Units to which the constraint applies* RST Trees.

5. *Nature of constraint(s)* Discourse level.

6. *Theoretical work or implemented* Partly implemented. Building on implementation and data structures of the ILEX system.[2]

7. *Approach to addressing constraints in generation* Mellish et al. propose stochastic search to address constraint satisfaction. It is a heuristic seach (i.e., an optimal solution is not guaranteed) that can be divided into the following steps:

   (a) A set of random candidate solutions CS_1 is constructed.

   (b) The following steps are repeated until some time limit is reached:

      i. At random, one or more items from the set CS_1 is selected (preferring items with the best "scores") to build a set CS_2.

      ii. CS_2 is used to produce further random variations that are collected in set CS_3.

      iii. Items in CS_3 are added to CS_1, possibly at the same time removing from CS_1 items with a low score.

   There are a number of parameters in this abstract algorithm:

   - Scores: There needs to be a scoring function to score items. In this paper, constraints are described which assign scores to RST trees depending on the properties of those trees. E.g., a tree which contains a fact that mentions no previously introduced fact is assigned score -9. The total score of a tree is obtained by summing the scores that individual constraints assign to that tree.
   - *Initialization.* A method for producing initial candidate solutions needs to be chosen. Typically, there are some hard constraints on these initial solutions.
   - *Production of further random variations.* Mellish et al. explore three approaches: swapping of random subtrees and two genetic algorithm approaches. The genetic algorithm approaches both use mutation

[2]See `www.sfu.ca/rst/04text_generation/ilex.html`.

and crossover on sequences of tree leaves to obtain random variations. Mutation is a unary operation: it maps an input to an output sequence. Crossover is binary: it takes two sequences and creates a new one from them. Two different internal representations of sequences are explored which give rise to different crossover and mutation operations.

8. *Single or many constraints? How are constraints combined?* Multiple constraints; see above.

9. *Any other comments on the paper* –

4.15 Moriceau and Saint-Dizier [39], 2004, 'A Constraint-Based Model for Preposition Choice in Natural Language'

1. *Name of the constraint(s) (if available)* –

2. *Brief informal description of the constraint(s)* Constraints for modelling of preposition lexicalization.

3. *Type of constraint(s)* Hard.

4. *Units to which the constraint applies* Sentences.

5. *Nature of constraint(s)* Syntactic and semantic.

6. *Theoretical work or implemented* Theoretical.

7. *Approach to addressing constraints in generation* –

8. *Single or many constraints? How are constraints combined?* Multiple hard constraints.

9. *Any other comments on the paper* –

4.16 Neumann and collaborators (1990–1998) on constraint-based grammars, reversible grammars and revision

Neumann and Finkler [44], 1990, 'A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures'

The first author of this paper ([44]), Günter Neumann, provided us with answers to our questions.

1. *Name of the constraint(s) (if available)* –

2. *Brief informal description of the constraint(s)*
    - lexical constraints for lexical choice
    - syntactic constraints based on head/modifier relationship; distinguish syntactic hierarchy and linearization
    - feedback with conceptual component to request missing constraints for local syntactic realization
3. *Type of constraint(s)* hard constraints.
4. *Units to which the constraint applies* mainly lexical and phrasal.
5. *Nature of constraint(s)* See 2.
6. *Theoretical work or implemented* fully implemented prototype fully embedded in a Dialog-System.
7. *Approach to addressing constraints in generation* unification-based grammar formalism based on constraint-based dependency grammar
8. *Single or many constraints? How are constraints combined?* used for realization in incremental and parallel manner, feedback with conceptual component
9. *Any other comments on the paper* –

Neumann and van Noord [45], 1992, 'Self-Monitoring with Reversible Grammars'

The first author of this paper ([45]), Günter Neumann, provided the following answers regarding the constraints dealt with in the paper:

1. *Name of the constraint(s) (if available)* self-monitoring.
2. *Brief informal description of the constraint(s)* When the semantic and syntactic structure of a sentence has been generated, it is parsed to check, whether there are alternative semantic readings; this is tested by comparing the semantic expression of both grammatical derivation trees.
3. *Type of constraint(s)* Mainly hard constraints, some preferences are built in as special cases
4. *Units to which the constraint applies* Sentence level.
5. *Nature of constraint(s)* Grammatical in the sense of modern constraint-based grammar theory, i.e, semantic, syntactic, morphological.
6. *Theoretical work or implemented* Prototype implemented using a large Dutch unification-based grammar.

7. *Approach to addressing constraints in generation* revision-based; integration of generation and parsing on basis of reversible unification-based grammar.

8. *Single or many constraints? How are constraints combined?* –

9. *Any other comments on the paper* –

Neumann [42], 1997, 'Applying Explanation-based Learning to Control and Speeding-up Natural Language Generation'

Information on [42] was provided by Günter Neumann.

1. *Name of the constraint(s) (if available)* Data-driven.

2. *Brief informal description of the constraint(s)* Based on NLG competence grammar & available corpus of annotated grammatical structures (disambiguated semantic/syntactic feature structures), a generation subgrammar is automatically extracted using Explanation-based Learning; the subgrammar automatically filters out all constraints not applicable for that domain.

3. *Type of constraint(s)* The remaining constraints are hard.

4. *Units to which the constraint applies* Phrasal, sentence.

5. *Nature of constraint(s)* –

6. *Theoretical work or implemented* –

7. *Approach to addressing constraints in generation* Explanation-based Learning of subgrammars for unification-based grammars.

8. *Single or many constraints? How are constraints combined?* –

9. *Any other comments on the paper* –

Neumann [43], 1998, 'Interleaving Natural Language Parsing and Generation Through Uniform Processing'

The following information on [43] was provided by Günter Neumann.

1. *Name of the constraint(s) (if available)* Revision on basis of integrated generation and parsing as part of realization component.

2. *Brief informal description of the constraint(s)* During realization with constraint-based grammar, parsing can be used to obtain "parsing-based constraints" which are then used to check, whether just generated forms should better be revised/reformulated or just canceled.

3. *Type of constraint(s)* hard constraints.

4. *Units to which the constraints apply* lexical, phrasal.
5. *Nature of constraint(s)* semantic and syntactic.
6. *Theoretical work or implemented* implemented for Dutch and German small constraint-based grammars.
7. *Approach to addressing constraints in generation* same chart-based algorithm for generation and parsing; item-sharing; revision-based; self-monitoring.
8. *Single or many constraints? How are constraints combined?* –
9. *Any other comments on the paper* –

4.17 Paiva and Evans [47], 2004, 'A Framework for Stylistically Controlled Generation'

1. *Name of the constraint(s) (if available)* –
2. *Brief informal description of the constraint(s)* Two factors obtained from a corpus-based factor analysis. The user can select target scores for the factors. The two factors roughly correspond with degree of reader involvement and style of referring expressions (pronominal to full nomimal reference).
3. *Type of constraint(s)* Soft.
4. *Units to which the constraints apply* Entire texts.
5. *Nature of constraint(s)* Stylistic.
6. *Theoretical work or implemented* Implemented.
7. *Approach to addressing constraints in generation* Individual generator decisions are correlated with surface stylistic features. The correlations are obtained by applying factor analysis to a corpus of texts to obtain stylistic factors. The generator is run in free mode and the output is scored in terms of the factors from the corpus study. Next, correlation equations are derived, using multivariate linear regression, relating generator decisions with factor scores.

   Using this information, we can then run the generator on new input with some target factor scores. At each decision point, all possible choices are enumerated and their score for both factors is computed. The decisions are ordered in terms of their distance to the target scores. The paper does not specify how the distance is precisely calculated.
8. *Single or many constraints? How are constraints combined?* Two constraints. See above.
9. *Any other comments on the paper* –

4.18 Piwek and Van Deemter [48], 2003, 'Dialogue as Discourse: Controlling Global Properties of Scripted Dialogue'

1. *Name of the constraint(s) (if available)* 'Number of turns' and 'degree of emphasis'.

2. *Brief informal description of the constraint(s)* Number of turns in which some content should be expressed (settings: min and max) and degree of marking information (through subdialogues) for emphasis (settings: min and max).

3. *Type of constraint(s)* Soft.

4. *Units to which the constraints apply* Scripted dialogue (script of a dialogue that is typically produced by a single author, i.e., a playwright).

5. *Nature of constraint(s)* Size and style.

6. *Theoretical work or implemented* Partly implemented.

7. *Approach to addressing constraints in generation* A dialogue planner produces an initial dialogue plan. Revision operations (adjacency pair aggregation and adjacency pair insertion) are performed exhaustively to obtain a set of revised dialogue plans. Each plan is scored in terms of its satisfaction of the turn and emphasis constraints. The plans are ranked in terms of the *product* of the scores. Multiplication is proposed as a fair way of comparing scores, following Nash's [40] work in Game Theory (known as the Nash arbitration plan).

8. *Single or many constraints? How are constraints combined?* Two soft constraints. See above.

9. *Any other comments on the paper* –

4.19 Power [49], 2000, 'Planning text by constraint satisfaction'

1. *Name of the constraint(s) (if available)* Constraint names include root domination, parental domination, argument order, multiple text-clauses, rhetorical grouping, etc.

2. *Brief informal description of the constraint(s)* The hard constraints characterize the set of text structures which represent rhetorical structure trees ([33]) and exclude fatal stylistic flaws in text structures. Soft constraints allow an ordering of text structures in terms of their number of non-fatal stylistic flaws.

3. *Type of constraint(s)* Hard and soft constraints.

4. *Units to which the constraints apply* partial text structures that are paired with rhetorical structure trees.

5. *Nature of constraint(s)* Constraints on text-structure formation, structural compatibility and style of text structure.

6. *Theoretical work or implemented* Implemented in the ICONOCLAST system using the ECLIPSE logic programming environment.[3]

7. *Approach to addressing constraints in generation* Constraint Satisfaction Problem (CSP) solver yields a set of text structures. Weighted soft constraints induce a preference order on this set (precise details of how the ordering is obtained are not given in this paper).

8. *Single or many constraints? How are constraints combined?* Multiple hard constraints are addressed using a CSP solver. The following steps are identified: 1. Creation of solution variables. This amounts to adding text level and order variables to nodes in the input rhetorical structure tree. 2. Assignment of domains to the variables. This can, for example, determine that the tree should be realized as a paragraph. 3. Application of constraints. 4. Enumeration of solutions. 5 Computation of complete text structures. Power points out that it is difficult formulate a configuration task as a CSP, because it is not known in advance how many variables are needed. It is, however, possible to use CSP to produce part of the required output text structure and have a subsequent deterministic phase that supplies the missing structure.

9. *Any other comments on the paper* –

4.20 Reiter [52], 2000, 'Pipelines and Size Constraints'

1. *Name of the constraint(s) (if available)* Size constraint.

2. *Brief informal description of the constraint(s)* Generation of personalized leaflets for smoking-cessation. The leaflets have to fit on a fixed number of pages, and these pages should be used optimally to present as much information as possible.

3. *Type of constraint(s)* Hard (the document must fit on four A5 pages), and soft (communicate as much information as possible; i.e., try to use as much of the four A5 pages as possible).

4. *Units to which the constraints apply* Entire text.

[3] See `http://www-icparc.doc.ic.ac.uk/eclipse/`.

5. *Nature of constraint(s)* Size constraint.

6. *Theoretical work or implemented* Implemented in the STOP system.

7. *Approach to addressing constraints in generation* Three alternative approaches were explored:

   (a) Single solution pipeline: The document planner produces an initial document plan. A heuristic size estimation function then assesses whether the document fits the page limit, if not a trimmer identifies the least important message and remove it. This is repeated until the document plan is predicted by the size estimator to fit the page limit. Then the document plan is passed to the microplanning module, which in turn passes its output to the realization component.

   (b) Multiple solution pipeline: The document planner runs several times, each time it is set to overestimate the size of the document to a different degree. This results in a set of documents differing slightly in how close they are to exactly using the amount of space allowed by the page limit. All document plans are processed by the microplanning and realization modules. A choice module selects from the final documents the one which is within the page limit and has the highest word count.

   (c) Extension with a revision module: A complete document is generated by the system. The exact size of the document is then obtained (number of pages and word count). A revision module then goes through a cycle adding and deleting messages until it finds the document with the highest word count that still satisfies the hard size constraint.

8. *Single or many constraints? How are constraints combined?* A hard constraint in combination with a soft one.

9. *Any other comments on the paper* –

4.21 Thiam Chye [10], 2005, 'Multi-document Summarization by Generation'

This is a master's thesis. The thesis advisor, Min Yen Kan, provided us with answers to our list of questions.

1. *Name of the constraint(s) (if available)* None.

2. *Brief informal description of the constraint(s)* A set of constraints are identified, which decide the word, phrase and sytactic choices (e.g. Active/Passive form).

3. *Type of constraint(s)* Preferences (Soft constraints).

4. *Units to which the constraint applies* Words and sentences.

5. *Nature of constraint(s)* Both syntactic and semantic (where a set of similar words/phrases are constrained).

6. *Theoretical work or implemented* Partially implemented (prototype system).

7. *Approach to addressing constraints in generation* The constraints are addressed via classification using supervised learning.

8. *Single or many constraints? How are constraints combined?* Yes, there are several sets of inter-dependent linguistic choices which need to be classified. The approach, which the author termed as Opportunistic Classification, is similar to a constraint satisfaction problem, in which linguistic decisions are mapped to a constraint graph, and the decisions are iteratively classified until a best solution to the set of constraints is achieved.

9. *Any other comments on the paper* The paper focused on answering question 8, dealing with the algorithm in solving inter-dependent linguistic choices.

4.22 Williams [59], 2003, 'Language choice models for microplanning and readability'

1. *Name of the constraint(s) (if available)* –

2. *Brief informal description of the constraint(s)* The constraints derive from a corpus. For each discourse relation we have six attributes (length of first text span, length of second text span, ordering of text spans, Position(s) of discourse cue phrase(s), between-text-span-punctuation and discourse cue phrases) and data on pairwise legal combinations of these. Given a specific discourse relation as input and the reader model, optimal values for the six attributes are sought. The corpus-based pairwise constraints can be tightened when reader model says that the reader is a poor reader. In particular, Williams tightens the constraints to obtain solutions that have cue phrases, especially common ones, and use punctuation between spans. Williams references a number of empirical studies which suggest that such features are beneficial for poor readers.

3. *Type of constraint(s)* Hard and soft.

4. *Units to which the constraints apply* Cue phrases, text spans and punctuation.

5. *Nature of constraint(s)* Discourse level, layout (ordering of text spans, punctuation).

6. *Theoretical work or implemented* Implemented in Java. Paper describes GIRL (Generator for Individual Reading Levels).

7. *Approach to addressing constraints in generation* The input is a model of the user's reading ability and a document plan (consisting of discourse relation trees). Two phases: 1. Constraint Satisfaction Problem (CSP) satisfier; 2. Filter: 'picks most frequently occurring one for good readers and the one with overall shortest sentences for poor readers' ([59], p.17).

8. *Single or many constraints? How are constraints combined?* Yes. Multiple hard constraints are managed by CSP solver. Soft constraints are applied as filters on solutions from the CSP solver.

9. *Any other comments on the paper*

Acknowledgements

We would like to thank our colleagues in the NLG community for helping us to identify relevant papers on constraint-based NLG. In particular, thanks go to Anja Belz, Chris Mellish, Min-Yen Kan, Imtiaz Khan, Irene Langkilde Geary, Tomasz Marciniak, Günter Neumann, Richard Power, Ehud Reiter, Graeme Ritchie and Sandra Williams.

Bibliography

[1] Wikipedia linear programming. `http://en.wikipedia.org/wiki/Linear_programming`. Consulted December 2005.

[2] Wikipedia constraint programming. `http://en.wikipedia.org/wiki/Constraint_programming`, 2005. Consulted December 2005.

[3] D.E. Appelt. *Planning English sentences*. Cambridge University Press, Cambridge, 1985.

[4] R. Barták. On-Line Guide to Constraint Programming. `http://kti.ms.mff.cuni.cz/ bartak/constraints/index.html`, 1998. Consulted 12 December 2005.

[5] S. Beale, S. Nirenburg, E. Viegas, and L. Wanner. De-Constraining Text Generation. In *Proceedings of the Ninth Workshop on Natural Language Generation*, Niagara-on-the-lake, Ontario, 1998.

[6] T. Becker and M. Lockelt. Liliput: a parameterizable finite-domain constraint solving framework and its evaluation with natural language generation problems. In *TRICS: Techniques foR Implementing Constraint programming Systems, a CP 2000 Workshop*, Singapore, 2000.

[7] S. Bistarelli. *Soft Constraint Solving and programming: a general framework*. PhD thesis, Dipartimento di Informatica, University of Pisa, 2001.

[8] R. Blutner. Some Aspects of Optimality in Natural Language Interpretation. *Journal of Semantics*, 17(3):189–216, 2000.

[9] C. Callaway and J. Lester. Dynamically improving explanations: a revision-based approach to explanation generation. In *Proceedings of IJCAI97 conference*, Nagoya, Japan, 1997.

[10] Lee Thiam Chye. Multi-document Summarization by Generation. Master's thesis, School of Computing, National University of Singapore, 2005.

[11] R. Dale. Generating Recipes: An Overview of EPICURE. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*. Academic Press, London, 1990.

[12] R. Dale and N. Haddock. Content Determination in the Generation of Referring Expressions. *Compututational Intelligence*, 7(4):252–265, 1991.

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

[14] M. Elhadad. *Using Argumentation to Control Lexical Choice: A Unification-based Implementation*. PhD thesis, Computer Science Department, Columbia University, New York, 1993.

[15] M. Elhadad, K. McKeown, and J. Robin. Floating constraints in lexical choice. *Computational Linguistics*, 23(2):195–239, 1997.

[16] C. Gardent. Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, 2002.

[17] C. Gardent and S. Thater. Generating with a grammar based on tree descriptions: a constraint-based approach. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, Toulouse, 2001.

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

[19] P. Gray, W. Hart, L. Painton, C. Phillips, M. Trahan, and J. Wagner. A Survey of Global Optimization Methods. `http://www.cs.sandia.gov/opt/survey/main.html`, 1997. Consulted December 2005.

[20] B. Grosz, A. Joshi, and S. Weinstein. Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, 1995.

[21] Bruce Hayes. OTSoft: Optimality Theory Software. `http://www.linguistics.ucla.edu/people/hayes/otsoft/`, 2004. Accessed December 2005.

[22] J. Hayes and L. Flower. Writing research and the writer. *American Psychologist*, 41:1106–1113, 1986.

[23] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming*. The MIT Press, Cambridge, Massachusetts, 1989.

[24] E. Hovy. *Generating Natural Language Under Pragmatic Constraints*. Lawrence Erlbaum, Hillsdale, New Jersey, 1988.

[25] E. Hovy. Pragmatics and Natural Language Generation. *Artificial Intelligence*, 43:153–197, 1990.

[26] K. Inui, T. Tokunaga, and H. Tanaka. Text revision: A model and its implementation. In R. Dale et al., editor, *Aspects of Automated Natural Language Generation: Proceedings of the Sixth International Natural Language Generation Workshop*, pages 215–230. Springer Verlag, 1992.

[27] A. Joshi and Y. Schabes. Tree adjoining grammars and lexicalized grammars. In *Tree Automata and Languages*. Elsevier Science, 1992.

[28] R. Kibble and R. Power. Optimizing Referential Coherence in Text Generation. *Computational Linguistics*, 30(4):401–416, 2004.

[29] I. Langkilde and S. Knight. The practical value of n-grams in generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-lake, Ontario, 1998.

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

[31] I. Langkilde-Geary and K. Knight. Halogen statistical sentence generator. In *Proceedings of the ACL-02 Demonstrations Session*, Philadelphia, 2002. Association for Computational Linguistics.

[32] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[33] W. Mann and S. Thompson. Rhetoretical structure theory: towards a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[34] Hisar Maruli Manurung, Graeme Ritchie, and Henry Thompson. A flexible integrated architecture for generating poetic texts. In *Proc of the Fourth Symposium on Natural Language Processing (SNLP 2000)*, pages 7–22, Chiang Mai, Thailand, 2000.

[35] Hisar Maruli Manurung, Graeme Ritchie, and Henry Thompson. Towards a computational model of poetry generation. In Geraint A. Wiggins, editor, *Proc of the AISB 00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 79–86. Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2000.

[36] T. Marciniak and M. Strube. Discrete optimization as an alternative to sequential processing in nlg. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG 2005)*, Aberdeen, UK, August 2005.

[37] C. Mellish. *Computer interpretation of natural language descriptions*. Ellis Horwood, Chichester, UK, 1985.

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

[39] V. Moriceau and P. Saint-Dizier. A constraint-based model for preposition choice in natural language generation. In *International Workshop on Constraint Solving and Language Processing – CSLP 2004*. Roskilde University, September 2004.

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

[41] G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization*. Wiley, New York, NY, 1999.

[42] G. Neumann. Applying explanation-based learning to control and speeding-up natural language generation. In *Proceedings of ACL/EACL-97*, Madrid, 1997.

[43] G. Neumann. Interleaving natural language parsing and generation through uniform processing. *Artificial Intelligence*, 99:121–163, 1998.

[44] G. Neumann and W. Finkler. A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures. In *COLING-90*, Helsinki, 1990.

[45] G. Neumann and G.J. van Noord. Self-Monitoring with Reversible Grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, 1992.

[46] Linear Programming Packages. `http://www.ici.ro/camo/hlp.htm`, 2004. Consulted December 2005.

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

[48] P. Piwek and K. van Deemter. Dialogue as Discourse: Controlling Global Properties of Scripted Dialogue. In Reva Freedman and Charles Callaway, editors, *Natural Language Generation in Spoken and Written Dialogue: Papers from the 2003 AAAI Spring Symposium*, pages 118–124. AAAI Press, 2003.

[49] R. Power. Planning texts by constraint satisfaction. In *Proceedings of COLING 2000*, pages 642–648, Saarbrücken, Germany, 2000.

[50] R. Power, D. Scott, and N. Bouayad-Agha. Document structure. *Computational Linguistics*, 29(2):211–260, 2003.

[51] A. Prince and P. Smolensky. Notes on connectionism and Harmony Theory in linguistics. Technical Report CU–CS–533-91, Dept. of Computer Science, University of Colorado, Boulder, 1991.

[52] E. Reiter. Pipelines and Size Constraints. *Computational Linguistics*, 26:251–259, 2000.

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

[54] K. De Smedt, H. Horacek, and M. Zock. Architectures for natural generation: Problems and perspectives. In G. Adorni and M. Zock, editors, *Trends in natural language generation: An artificial intelligence perspective*, volume 1036 of *LNAI*, pages 17–46, Berlin, 1996. Springer.

[55] O. Stock and M. Zancanaro, editors. *Multimodal Intelligent Information Presentation*, volume 27 of *Text, Speech and Language Technology*. Springer, Dordrecht, 2005.

[56] M. Stone. On Identifying Sets. In *Proceedings of INLG-2000*, Mitzpe Ramon, 2000.

[57] K. van Deemter. Towards a Probabilistic Version of Bidirectional OT Syntax and Semantics. *Journal of Semantics*, 21(3), 2004.

[58] K. van Deemter and J. Odijk. Context modeling and the generation of spoken discourse. *Speech Communication*, 21(1/2):101–121, 1997.

[59] S. Williams. Language choice models for microplanning and readability. In *Proceedings of the Student Workshop of the Human Language Technology and North American Chapter of the Association for Computational Linguistics Conference*, pages 13–18, Edmonton, 2003.

[60] I. Witten and E. Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, California, 2000.