Generating referring expressions containing quantifiers

Sebastian Varges & Kees van Deemter
Information Technology Research Institute
University of Brighton, UK
{Sebastian.Varges|Kees.van.Deemter}@itri.brighton.ac.uk

Recent work on the Generation of Referring Expressions has increased the generating capability of algorithms in this area. This paper asks whether the models underlying these proposals can still be used if even more complex referring expressions are generated. To discuss this issue, we will investigate a variety of referring expressions that pose difficulties to current generation algorithms. In particular, we will discuss the difficulties associated with quantified referring expressions (such as '*those women who have fewer than two children*', or '*the people who work for exactly 2 employers*') and explain how they can be generated by extending the inference-based approach described in (Varges, 2004).

1 GRE: descriptions using conjunctions of atoms, and beyond

The Generation of Referring Expressions (GRE) is a key task in most Natural Language Generation (NLG) systems, which is usually viewed as part of microplanning (Reiter and Dale 2000, Chapter 5). Simply put, GRE amounts to finding suitable descriptions in natural language that allow a hearer/reader to identify the intended referent. GRE systems take their departure from a Knowledge Base (KB) of information that is shared between speaker and hearer, and which can therefore be used by the speaker (that is, the computer) to identify the intended referent for the hearer. The starting point for most recent work in this area is the Incremental Algorithm of Dale and Reiter (1995). It uses an `Attribute:Value` notation to represent properties in the KB; descriptions are represented as lists of `Attribute:Value` pairs, for example the following list of two such pairs [*Type* : *Instrument*, *Colour* : *Red*] which later modules may realize linguistically as '*the red instrument*' or '*the instrument that is red*'. Logically, a list of `Attribute:Value` pairs is interpreted as a *conjunction* of properties. In the last decade, algorithms of this kind have been adapted and extended in various ways, allowing the use of more complex properties, for example

1. *Relational* properties (e.g., '*the dog in the shed*', Dale & Haddock 1991),

2. *Boolean* properties (containing negation, disjunction and conjunction of properties) (van Deemter 2002, Gardent 2002, Horacek 2004).

Current GRE algorithms are very limited in that there are many types of referring NPs that they are unable to generate.[1] Here we will explore the possibility of further extensions.

2 GRE: Formalizing the problem

As GRE becomes more complex, two things become increasingly important. One is to use formally neat and well-understood formalisms, the other is to use fast algorithms. Both issues were addressed by a recently proposed formalism using labeled directed graphs (Krahmer et al. 2003). The issues that we will discuss in section 3 are relevant to any type of GRE, but they show themselves with particular force when applied to the graph-based formalism, which is why we will use it as an example. Labeled directed graphs are formally well understood, many results about its computational properties are known, and fast optimizations are available for many algorithms. An additional advantage *appears* to be that the graph-based approach represents descriptions using the exact same formalism as the KB. We shall later see, however, that this is also the main limitation of the approach. First, however, let us informally sketch the graph-based approach.

Both the domain model (the 'scene graph') and the description are labeled directed graphs. Properties and types are looping edges labeled with `Attribute:Value` pairs, whereas relations are modeled as edges between nodes. Consider the scene graph depicted in figure 1. There are four musicians and four red instruments. Musician m_1 drops instrument i_1, Musician m_2 holds instrument i_2, musician m_4 holds instrument i_4 and musician m_3 holds the three instruments i_2, i_3 and i_4. Let us assume that the designated target element s is i_1, i.e. we want to generate an expression referring to domain object i_1.

Representing both the description and the scene using graphs allows one to view GRE as a *graph construction* problem. The task is to construct a description pair that 'refers uniquely' to a given scene pair. The notion 'refers uniquely' is defined via the notion of a **subgraph isomorphism** (see Krahmer et al. 2003 for more details). Determining whether a given

[1] Non-referring NPs are harder still, and usually not addressed in GRE.

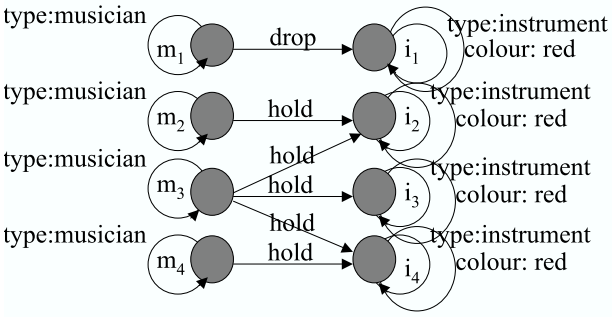

Figure 1: A scene graph involving eight objects

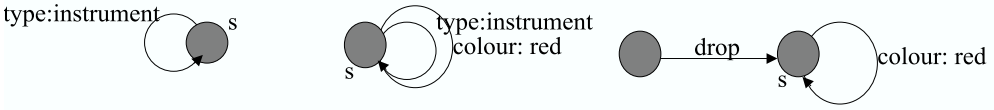

Figure 2: Three possible description graphs

description graph (with a designated element in it) refers uniquely to a designated target element in the description graph becomes a matter of *comparing* the two graphs.

Figure 2 shows a number of description graphs, each of which has s as its designated element. The first two of the three description pairs refer to s but not uniquely (they may also refer to the 'confusables' i_2, i_3, i_4), while the right-most description graph refers to s uniquely. It might be worded as *'the red thing that was dropped'* (many other references are possible of course).

Reference to *sets* is achieved by constructing a description graph that uniquely refers to a set of target referents. In van Deemter and Krahmer (IN PRESS), it is shown that a number of more complex phenomena can also be handled, namely Boolean properties, degrees of salience (Krahmer and Theune 2002), and context-dependent properties such as 'large(st)' in 'the large(st) cat' (van Deemter 2000).

3 Generating Quantified Referring Expressions

In this paper, we will focus on a limitation of the graph approach that seems difficult to repair. The graph-based approach requires every description graph to look like a scene graph. This means that descriptions can never contain any logical structure that is not already present in the scene

graph. This rules out, for example, descriptions like '*those musicians who play fewer than three instruments*' simply because things like '*fewer than three instruments*' cannot be expressed. Imagine that we somehow found a way of interpreting a description pair as saying '*musicians who play fewer than three instruments*'. The idea of a subgraph isomorphism is essentially *existential* in nature: the definitions of section 2 dictate that the description refers if *there exists* a subgraph of the scene graph such that a truth-preserving bijection between subgraph and scene graph can be found. But this would predict incorrectly that the description refers if, according to the scene graph, some musicians played *ten* instruments, for in this case we select a subgraph that leaves out a number of the instruments. In other words: the idea of a subgraph isomorphism is limited to the case of monotonically increasing quantifiers (e.g. Barwise and Cooper 1981): it works well for '*those musicians who play more than three instruments*' but not for '*those musicians who play fewer than three instruments*' or '*those musicians who play exactly three instruments*'.

The task that we are setting ourselves here is to generate complex referring expressions of this kind, in a framework that has – ideally – the same advantages as the graph-based approach.

With quantified referring expressions we are entering the domain of relative clauses. In principle, relative clauses can contain anything. This is brought out perhaps most vividly by Montague's PTQ system, for example, which contains the following rule (which overgenerates and exaggerates the problem slightly as a result):

> 'If $\zeta \epsilon P_{CN}$ and $\phi \in P_T$ then $F_{(3,n)}(\zeta, \phi) \in P_{CN}$, where $F_{(3,n)}(\zeta, \phi) =$ "ζ such that ϕ'"; and ϕ' comes from ϕ by replacing each occurrence of he_n or him_n by he/she/it or him/her/it respectively, according as (...)'

This means that ϕ can be literally any sentence of the language.[2] Given the almost unlimited scope of the problem, we cannot solve it in its full breadth, but we will attack what we see as the core problem: to generate distinguishing descriptions of the form

> [The [CN (that are) R-ing D CN]$_C N]_N P$",

[2] If ϕ does not contain he_n or him_n then $F_{(3,n)}(\zeta, \phi) =$ "ζ such that ϕ, e.g., 'woman such that John walks'.

where CN is a (potentially complex) Common Noun, R is a verb having two argument slots, D is a determiner expressing a quantifier that is either monotonically increasing, or monotonically decreasing, or none of the two (as when Q = 'exactly 3').

4 The inferential approach to GRE

Partly in response to the representational problems of the graph approach, Varges (2004) proposed the 'inferential approach' as an alternative. It uses boolean combinations of attributes, types and relations to describe individuals and sets of objects. The approach can be characterized as 'inferential' in the sense that it derives facts from a domain representation and uses these as ingredients for the construction of referring expressions. Thus, in contrast to the graph approach, there is a separation between the domain representation and the representation of the logical content of the referring expressions. This avoids the above-mentioned problem of having to express both within the same representational framework.

The inferential approach to GRE tries to find all combinations of properties that are true of a given target referent – and, more specifically, those combinations that distinguish the target from all distractors. The approach pairs the logical forms derived from the domain model with the corresponding 'extensions', the set of objects that the logical form denotes. In our example domain, the set of all domain objects for which attribute 'colour' has value 'red' contains objects i_1, i_2, i_3 and i_4. This is represented as the pair $<attr\_val(colour : red), \{i_1, i_2, i_3, i_4\}>$. Object types are similarly described by pairs of logical form and extension: $<type(musician), \{m_1, m_2, m_3, m_4\}>$. This also allows us to represent the super- and subtypes of the types associated with the domain objects, for example the set of all 'artifacts' or the set of all 'persons' if a subsumption hierarchy is defined in the domain model.

The extension serves as the basic interface for combining pairs of logical form and extension (in what follows, we will use the term 'descriptions' for these pairs). For example, conjoining the logical forms of two descriptions requires us to intersect their extensions. Likewise, negation of a logical form means forming the complement set of the corresponding extension (under a closed world assumption), and disjunctive descriptions are built by forming the: closed world assumption union of the extensions. Relations are regarded as a means of relating the sets of objects described by two extensions, for example those that hold something and those that are being held. Thus,

given two descriptions, we consult the domain model to check whether a given relation holds between at least one pair of objects in the two extensions. If this is the case, we combine the descriptions by means of the relation in question, for example $type(musician) + relation(hold) + type(instrument)$. We can also use the same mechanism to find the set of objects for which the relation does not hold.

Relations involve (at least) two descriptions, each with their corresponding extension, but our combination operations require the existence of a single extension in every description. We solve this potential problem by introducing the notion of the 'focus' of a relational description. In essence, this means deciding on which one of the descriptions involved we want to focus, for example on *'the musicians holding the instruments'* or on *'the instruments being held by the musicians'*?

In figure 3, we detail the description-combining rules and give examples of descriptions and the corresponding realizations. The relation rules described deal with binary relations only.[3] Since there are two options for the focus of the output of the relation rule, and two options for positive and negated relations, four relational descriptions can be produced.

A major factor in the conception of the inferential approach were issues of realization and search. Content determination, i.e. the generation of descriptions, is closely linked to (bottom-up) realization: only descriptions that have been successfully realized can be recombined further. This prevents the generation of ever more complex descriptions that cannot be realized. Furthermore, surface realizations of partial referring expressions are combined compositionally. This allows us to define search algorithms that make use of monotonic cost functions to score competing referring expressions. For example, we may search for the referring expression that contains the smallest number of words and we can exploit the fact that realizations are concatenated compositionally (Varges 2004).

5 Generating quantified referring expressions in the inferential approach

The inference rules described in the previous section can identify those musicians that hold an instrument and those that do not. They can also identify those instruments that are held by a musician and those that are not (see

[3] We believe our approach can be straightforwardly extended to n-ary relations for $(n > 2)$.

1. Conjunction: given two descriptions d_1 and d_2, generate a new description d_3 whose extension ext_{d_3} is the intersection $ext_{d_1} \cap ext_{d_2}$, for example $<(type(instrument) \wedge attr\_val(colour : red)), \{i_1, i_2, i_3, i_4\}>$.
   Realization: *'the red instruments'*.

2. Disjunction: given two descriptions d_1 and d_2, generate a new description d_3 whose extension ext_{d_3} is the union $ext_{d_1} \cup ext_{d_2}$, for example $<(type(musician) \vee type(instrument)), \{i_1, i_2, i_3, i_4, m_1, m_2, m_3, m_4\}>$.
   Realization: *'the musicians or the instruments'*.

3. Negation: given a description d_1, generate a new description d_2 whose extension ext_{d_2} is the complement $\overline{ext_{d_1}}$, for example $<\neg type(instrument), \{m_1, m_2, m_3, m_4\}>$. Realization: *'Not an instrument'*.

4. Relations: given two descriptions d_1 and d_2 and a relation name rel:

   - let extension $ext_{dom,pos}$ contain all $o_i \in ext_{d_1}$ that are in domain of rel with at least one member of ext_{d_2} in its range.
   - let extension $ext_{dom,neg}$ contain all $o_i \in ext_{d_1}$ that are *not* in domain of rel with any member of ext_{d_2} in its range.
   - let extension $ext_{ran,pos}$ contain all $o_j \in ext_{d_2}$ that are in range of rel with at least one member of ext_{d_1} its domain.
   - let extension $ext_{ran,neg}$ contain all $o_j \in ext_{d_2}$ that are *not* in range of rel with any member of ext_{d_1} in its domain.

   4.1 generate new description d_3 with extension $ext_{dom,pos}$ and focus on domain of rel, for example: $<hold(FOCUS(type(musician)), type(instrument)), \{m_2, m_3, m_4\}>$.
   Realization: *'the musicians holding the instruments'*.

   4.2 generate new description d_5 with extension $ext_{dom,neg}$ and focus on domain of rel, for example $<\neg hold(FOCUS(type(musician)), type(instrument)), \{m_1\}>$.
   Realization: *'the musician not holding the instrument'*.

   4.3 generate new description d_4 with extension $ext_{ran,pos}$ and focus on range of rel, for example $<hold(type(musician), FOCUS(type(instrument))), \{i_2, i_3, i_4\}>$.
   Realization: *'the instruments being held by the musicians'*.

   4.4 generate new description d_6 with extension $ext_{ran,neg}$ and focus on range of rel, for example $<\neg hold(type(musician), FOCUS(type(instrument))), \{i_1\}>$.
   Realization: *'the instrument not being held by the musicians'*.

Figure 3: Content determination rules of inference-based GRE without quantification

figure 3). However, they cannot generate the following referring expressions although the appropriate information is available in the domain:

(1) a. $\{m_2, m_4\}$: 'the musicians holding exactly one instrument',

b. $\{m_3\}$: 'the musician holding exactly three instruments',

c. $\{i_3\}$: 'the instrument being held by exactly one musician',

d. $\{i_2, i_4\}$: 'the instruments being held by exactly two musicians'.

Being able to generate these NPs is clearly desirable since without using quantifiers one is not able to 'refer to' the sets of objects (the extensions) on the left of (1). In other words, quantifiers increase the expressibility of the referring expression generator, and our goal is to extend the inferential approach to the generation of quantified referring expressions such as examples (1a-d).

The following ingredients seem to be necessary in our compositional framework: as before, we will need unquantified NPs such as '*the musicians*'. We will also need a relation rule. In addition, we also seem to require the generation of quantified NPs such as '*exactly one instrument*'. This immediately poses a problem to the inferential approach as outlined above: the NP '*exactly one instrument*' can "refer" to any of the four instruments in the example domain of figure 1. This can be represented as a set-of-sets $\{\{i_1\}, \{i_2\}, \{i_3\}, \{i_4\}\}$. However, this is incompatible with the single-set interface required by the already established rules. There seem to be at least the following options for dealing with this problem: First, one could extend the entire framework for dealing with set-of-sets representations. Second, one could generate individual descriptions for each of the possible extensions, i.e. an NP '*exactly one instrument*' for extension $\{i_1\}$ and another one for $\{i_2\}$ etc. Third, one can try to find a way of generating quantified descriptions without being forced to represent the 'uncertainty' expressed by set-of-sets containing more than one set. In the following, we investigate the last of these options.

5.1 Generating quantified descriptions using single-set representations

At a first glance, it may seem impossible to generate quantified relational descriptions without representing extensions as set-of-sets. However, when looking more closely at the examples of quantified referring expressions we

set out to generate (examples 1), we see that it is always the *non-focus* description that is quantified. In other words, the relational description incorporating the quantified NP still has a single-set focus extension: we know exactly which musicians hold two instruments, for example. Thus, the relational description can be combined with any other description as before.

This still leaves the problem that in a compositional approach it seems to be necessary to generate NPs such as '*exactly one musician*', which does appear to require a set-of-sets representation. However, even when using a bottom-up algorithm for realization, generation is triggered by logical forms ('descriptions'). It is possible to generate a logical form for the quantified relation but then realize it in a more template-based way 'in one go'. At the realization level, this can be done by combining a focus NP with a non-focus Nbar constituent, for example NP '*the musicians*' + '*holding exactly two*' + Nbar '*instruments*', or NP '*the instruments*' + '*being held by exactly two*' + Nbar '*musicians*'. This alone is not sufficient in general as it prevents the quantification of more complex NPs. Therefore, in order to increase expressibility at the realization level (while keeping the compositionality of surface forms), we also use combinations such as '*exactly one <u>of</u>* + NP *(the musicians or the instruments)*', for example.

The next issue is the required content determination rule, and here the advantages of the inferential approach become clear: if we can infer a relation involving quantification from the domain model, then we can generate the appropriate description and realize it. More formally:

Rule 5: Given already-realized descriptions d_1 and d_2, a cardinal N and a relation name rel:

- let extension $ext_{dom,card:N}$ contain all $o_i \in ext_{d_1}$ that are in domain of rel and map to exactly N $o_j \in ext_{d_2}$.
- let extension $ext_{range,card:N}$ contain all $o_j \in ext_{d_2}$ that are in range of rel and are mapped to by exactly N $o_i \in ext_{d_1}$.

5.1 generate new description d_3 with extension $ext_{dom,card:N}$, focus on domain of rel and cardinal quantification of range of rel, for example:
$<hold(FOCUS(type(musician)), CARD_3(type(instrument))), \{m_3\}>$.
Realization: *'the musician holding exactly three instruments'*.

5.2 generate new description d_4 with extension $ext_{range,card:N}$, focus on range of rel and cardinal quantification of domain of rel, for example:

$<hold(CARD_2(type(musician)), FOCUS(type(instrument))), \{i_2, i_4\}>$.
Realization: *'the instruments being held by exactly two musicians'*.

Rule 5 imposes a distributive reading of quantification: in the last example, each instrument individually is being held by two musicians (although the two sets of musicians do not need to be disjoint). A collective reading can be obtained by accumulating the domain objects in the extensions to be quantified over across mappings to/from the focus extension. In addition to the desired output, rule 5 also triggers the generation of 'redundant' output such as:

(2) $\{m_2, m_4\}$ = '(the musicians holding exactly one instrument) holding exactly one instrument'.

The generation of (2) can be prevented by applying a constraint that requires the focus extension to be reduced when attempting further combinations. This 'redundancy constraint', originally developed for constraining rules 1-4 (Varges 2004), can also be used for dealing with quantified referring expressions: in (2), the focus extension realized by the NP in brackets already refers to $\{m_2, m_4\}$. Thus, the redundant composition can be prevented because the focus extension is not reduced.

6 Discussion: possible extensions and limitations

The currently implemented generator without quantification uses 34 description building rules (distinguishing between negation of different description types, amongst others) and 21 rules for NP generation. The rules of the generator are expressed as productions in an expert system shell (JESS, Friedman-Hill 2003), the knowledge base of which serves as the chart of our chart generator. Quantification as described in the previous section adds one description building rule (implementing rule 5) and two realization rules (for gerunds in active and passive) to the system but leaves the existing rule base unchanged. This has practical (software engineering) advantages over changing the entire system to dealing with extensions represented as set-of-sets (the first option mentioned in section 5). The rules of our expert system-based implementation tend to be more complex than traditional grammar rules, for example making function calls to compute the extensions.

Our approach can be extended to generating *'fewer than N'* and *'more than N'*. For determining N, we only use the cardinals established in the

computations for generating *'exactly'*-NPs. In addition to the computational advantage of limiting the number of Ns that need to be considered, this also avoids problems concerning false implicatures since *'fewer than three X'* seems to imply that there are also *'exactly three X'* or *'more than three X'*. Other 'precise' quantifiers such as *'every'* or *'at least'* can be treated similarly. 'Imprecise' quantifiers such as *'some'*, *'most'* or *'many'* are harder to deal with because they have to be made precise in order to handle them. However, this is a general research issue, not a particular problem of the inferential approach.

Quantified expressions can be composed recursively as example (2) shows. By using the extension as the main interface between descriptions, arbitrarily nested NPs can be generated. Considering Montague's PTQ system (section 3), our ϕs cannot be any string of the language but have to restrict the set of denoted domain objects. This is due to the redundancy constraint which contains overgeneration. Our treatment of quantifiers inherits this advantage from the already established inferential framework.

On the other hand, the inferential approach still generates large numbers of referring expressions. In a recent paper, Norman Creaney (Creaney 1999) proposed a principle of Informativity for ranking and selecting among different (non-referring) quantified NPs in NLG, each of which is verified by a given model. Informativity could be used, in the setting of the present paper (where the quantifiers in question are part of referring expressions), to rank referring expressions that pick out *the same set of domain objects*. In other words, Informativity would only apply within equivalence classes. (In a more sophisticated approach, there could be a trade-off between Informativity and other goals such as minimizing surface length.) However, the ranking of referring expressions is beyond the scope of this paper.

There are also limitations to our approach, and it is informative to point these out: our basic strategy is to avoid having to represent the uncertainty involved in generating NPs such as '*exactly two trumpets*'. However, it seems that such constituents are necessary to capture some of the syntactic phenomena we would like to handle, for example '*exactly two trumpets or fewer than five drums*', or '*holding two trumpets or blowing exactly one flute*'. It seems to us that these could only be handled by more radical changes to the framework if we want to avoid the introduction of additional special-purpose content and realization rules. On the other hand, for practical purposes, using special-purpose rules may be an option because it avoids the complexities associated with set-of-sets representations.

The example domain used in this paper contains only a few domain objects, begging the question how our approach scales to more densely pop-

ulated domains. Our strategy in such cases is to make the agenda-based chart algorithm more goal-directed by changing the agenda ordering. Instead of generating descriptions of all domain objects in parallel we give preference to those descriptions that are 'salient' in some way, for example due to salient attributes such as colour or spatial proximity to the target object(s). This may mean loosing optimality. However, humans are not always optimal either which may be the result of similar time/processing constraints. Goal-directed search for GRE is largely left to future work.

7 Conclusion

In this paper, we have discussed the implications of letting a GRE program generate referring expressions containing quantifiers, a problem that has not been addressed before, to the best of our knowledge. We have shown that the graph-based approach faces substantial problems when it comes to representing quantifiers and with respect to its operational core, the subgraph isomorphism algorithm. We have shown how another approach to GRE, the 'inferential approach' which builds on the proposal of Varges (2004), avoids the representational difficulties of the graph approach by using different types of representations for domain model and descriptions. Furthermore, it is extensible in the sense that new phenomena (quantifiers, in this case) can be handled by the introduction of additional inference rules.

8 Acknowledgments

Our thanks go to Albert Gatt for helpful comments on this paper. The work reported here was carried out as part of the TUNA project, which is funded by the UK's Engineering and Physical Sciences Research Council (EPSRC) under grant number GR/S13330/01.

9 References

Barwise, J. and Cooper, R. (1981). *Generalized Quantifiers And Natural Language.*, Linguistics and Philosophy **4** (1981):159–219.

Chartrand, G. and O. Oellermann (1993). *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York.

Creaney, N. (1999). *Generating Quantified LFs from Raw Data.* In R. Kibble and K. van Deemter (Eds.), Procs. of workshop on 'The Generation of Nominal Expressions', 11th European Summer School in Logic, Language, and Information (ESSLLI'99), Utrecht.

Dale, R. (1992). *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*, The MIT Press, Cambridge, Mass.

Dale, R. and N. Haddock (1991). Generating Referring Expressions involving Relations, *Proceedings of the European Meeting of the Association for Computational Linguistics* (EACL 1991), Berlin, 161–166.

Dale, R. and E. Reiter (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions, *Cognitive Science* **18**: 233–263.

van Deemter, K. (2000). Generating Vague Descriptions, *Proceedings of the First International Conference on Natural Language Generation* (INLG 2000), Mitzpe Ramon.

van Deemter, K. (2002) Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm" *Computational Linguistics* **28** (1): 37-52.

van Deemter and Krahmer (IN PRESS). *Graphs and Booleans: on the generation of referring expressions.* To appear in 'Computing Meaning', Volume 3, 'Studies in Linguistics and Philosophy' series, Kluwer, Dordrecht.

Friedman-Hill, E. (2003). JESS - the Java Expert System Shell, Version 6.x, Sandia National Laboratories.

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

Garey, M. and D. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H.Freeman.

Gibbons, A. (1985). *Algorithmic Graph Theory*, Cambridge University Press, Cambridge.

Horacek, H. (2004). On Referring to Sets of Objects Naturally. *Proceedings of the Third International Conference on Natural Language Generation (INLG-04).*

Krahmer, E. S. van Erk and A. Verleg (2003). Graph-based Generation of Referring Expressions, *Computational Linguistics*, **29**(1): 53–72.

Krahmer, E. and M. Theune (2002). Efficient context-sensitive generation of referring expressions, in: *Information Sharing*, K. van Deemter and R. Kibble (eds.), CSLI Publications, CSLI, Stanford, 223–264.

Reiter, E. and R. Dale (2000). *Building Natural language Generation Systems.* Cambridge University Press, Cambridge, UK.

Stone, M. (2000). On Identifying Sets, *Proceedings of the First International Conference on Natural Language Generation* (INLG 2000), Mitzpe Ramon.

Varges, S. (2004). Overgenerating referring expressions involving relations, *Proceedings of the Third International Conference on Natural Language Generation (INLG-04).*