Towards classification of generation subtasks

Stephan Busemann
DFKI
Stuhlsatzenhausweg 3, D-66123 Saarbrücken email:
busemann@dfki.uni-sb.de

July 17, 1994

1 Generation subtasks and system components

The problem of NL Generation is often subdivided into subproblems, which correspond to separate, interacting system components. Hundreds of re-search papers refer to the subdivision of content planning vs. realization, or what-to-say vs. how-to-say-it, or strategical vs. tactical generation.[1] A closer look at implemented systems shows that subdivisions often differ very much from each other. Ongoing debates, e.g. about the interface between the components, suffer from a lack of a shared, precise terminology.[2] Comparison of NLG approaches is notoriously difficult since the way generation subtasks are assigned to system components depends on specific circumstances (such as the application task adopted, properties of the knowledge base used, the grammar theory adopted or the languages used).

For instance, some subtask called "lexical choice" is sometimes part of the content planning component, sometimes included in the realization compe-

[1] As you will have noticed, this paper, too, follows the habit to mention this subdivision at the very beginning.

[2] If it is precise it often is motivated by attempts to solve a specific generation problem at hand, and may thus not be very useful in general.

nent, sometimes it forms sort of a mediator between the two. Unfortunately the precise tasks to be achieved under the heading of "lexical choice" differ very much, both in breadth (content words, anaphora, compounds, inter-action with syntax, ...) and depth (are the input concepts designed with linguistic processing in mind, or just for the application task at hand?) For some overview see [7, Chapter 11].

Although some important work on the interface between content planning and realization has been done [9], there is, at present, no general and precise definition of the distribution of concrete generation subtasks to concrete system components available yet. And there is no general and precise definition of what constitutes a generation subtask either.

Clearly, such definitions must capture the variety of meanings included in intuitive and informal notions like "lexical choice". Hence a way of classifying the variants is needed. The definitions will serve to eliminate the vagueness of intuitive notions since authors will be able to define the functionality of their lexical choice components in terms of well-defined elementary subtasks. It is thus necessary to identify the basic building blocks of lexical choice components and try to provide a highly modular task description. For instance, author A might include anaphora, while author B views this as a separate subtask; author A might generate nominal compounds while author C assumes them to be lexicalized etc. The size of the subtasks should be such that the important differences between systems can be captured by virtue of different combinations of subtasks.

An interesting problem consists in the dependencies of subtasks on formalisms, data, control structures or on the definition of other subtasks. At some point, a further decomposition might not be possible for one lexical choice component while it still is for another one. Observations of this kind might raise the question of how modular the system was designed.

Using a shared terminology, which is based on modular views of the generation process, has large advantages. Existing work will be classifiable and comparable to the extent it can be decomposed into modules. It will be much easier to describe and understand the precise functionality of many systems. For classes of system components, standardized interfaces will become possible (e.g. for the input language), thus making NLG technology more reusable, and thus more interesting for industrial application.

Much future research and development can be based on elementary (or composite) subtask definitions. Application scenarios specify the functional-

ity of the NLG system needed; if a new system is to be built, at least parts of it could be formed by existing software with standard interfaces. At the horizon, the view of a "NLG toolbox" becomes visible.

For instance, despite the demanding and complex research and the promising results in our field, industrial application tasks are often simple enough to consider the use of well-known template-based techniques in NLG as perfectly adequate—techniques that have been condemned as too inflexible to yield natural, context-sensitive utterances. Interesting enough, there don't seem to be any standards, e.g. for the kind of input the system must be able to process. As a consequence, the simple, template-based approaches are very likely to be redone over and over again.

Having motivated the need for a precise, shared terminology and for the definition of elementary subtasks, I will undertake first steps to developing a classification scheme for the (composite) subtask of surface generation in order to illustrate what should go into a definition of a subtask.

2 Towards a classification for the surface generation subtask

Surface generation is the task of realizing a surface-level sentence description into a target language sentence using linguistic knowledge (e.g. a grammar). Input to surface generation components corresponds to a particular utterance, i.e. all decisions have been made and all meaning-encoding has been completed. The remaining task is the translation of the system-internal representation into natural language. Surface generation is thus a proper subtask of realization.[3]

The further subdivision of realization into a meaning-encoding and a meaning-ignoring part was introduced in order to gain modularity since the final subtask is a problem that should best be attacked with specific, linguistic methods and that can in fact be solved by building a reusable and adaptable system. Thus we have a tripartite model of generation:

- content planning determines the information to be conveyed to the

[3] I do not assume a particular architecture here; surface generation can go with feedback or integrated architectures as well as with incremental flow of control or the old-fashioned pipeline model.

hearer, which is sometimes referred to as the message level

- decision making with regard to linguistic shape (distribution of information onto sentences and sentence constituents, ordering information, deciding about anaphoric and elliptic elements, determining voice etc.)
- surface generation

The kind of surface generators as sketched above has various instances. The early systems grew up in the context of non-english languages (e.g. Dutch [8], German [3, 2, 6]). For highly inflecting languages the need of a separate component was simply more obvious. During the "syntax revival" in the early Eighties the linguistic deficiencies of that early work became apparent, especially the procedural approach to linguistic knowledge caused problems. A clear separation between linguistic grammar knowledge and the algorithm's control structure was achieved within the next generation of theory-dependent systems (e.g. [4] for GPSG, [10] for LFG). The algorithm does not depend on the grammar anymore, but still on aspects of the formalism. Hence it is not entirely transportable to other theoretical frameworks. New linguistic methods allowed a more general approach to linguistic phenomena, mainly due to formalisms based on type inheritance. The present generation of surface generators (e.g. [11, 12, 5]) includes algorithms that are theory-independent, thus allowing for better modularization: besides the common distinction between grammar and processing, we can now distinguish between the grammar formalism and the control strategy as well, thus allowing for relatively free combination of the modules. For instance, the semantic-head-driven approach has been used at least within the frameworks of HPSG, CUG, and DCG.

In the following survey, which is far from being complete, aspects of surface generation variability are identified that must go into a classification of the task. The idea is that the function of surface generators can be described by a combination of features, and that, eventually, feature patterns crystallize out that will form interesting classes of surface generators.

Algorithmic issues:

- Is the system a first, second, or third generation instance?
- Does the algorithm terminate on all input?

- Is the algorithm complete, i.e. does it generate from all the input specified?
- Is the algorithm coherent, i.e. does it generate only from the input specified?
- What does run-time depend on? Is the system deterministic? Does it use preferences?
- Is incremental processing possible? How does incremental output depend on the size of the input increments?

Formalism issues:

- For constraint-based frameworks, subsumption is an important predicate over pairs of feature structures. It can be used to guarantee completeness and coherence (in the above sense). Subsumption may be very costly, depending on the power of the formalism used. While this is not a big issue for simple PATR-like feature structures, formalisms including disjunction and negation (e.g. [1]) require a translation into disjunctive normal form, thus spoiling all the elegance and efficiency gained elsewhere.
- Is there an interface to the external world (for calling external functions, or accessing external data etc.)?

Grammar issues:

Is the grammar non-directional or is it geared towards generation?

If the generator works from semantic representation, how does it generate words that don't have a semantics at all (e.g. "to" in *he wants to see her?*

Does the grammar allow for partial cyclic representations, thus yielding non-termination of the generator?[4]

For every generation strategy certain features of a grammar are essential [12], i.e. processing is doomed to failure if they are not present. Can these features be automatically derived from a grammar?

[4] If the generator only looks at semantic information and semantics is copied to another place in the derivation tree, the danger of non-termination becomes apparent.

- Will an underspecified input yield multiple results?

Input issues:

- What kinds of information does the input language express?
- How is the correctness of a given input structure checked?
- Is the input organized as a hierarchy (e.g. a feature structure), or as a flat data structure (e.g. a list or set)? The latter would allow chart-based strategies while the former might not.
- Does the input uniquely determine lexical entries, or is lexical access ambiguous (e.g. by allowing both *bachelor* and *unmarried male human, which is not a catholic* priest)?

Output issues:

- Is the output written or spoken? What morphological or phono logical device is required?
- Are fully inflected forms lexicalized or computed?

Scalability:

- Is the grammar extendible without sacrificing efficiency?
- Can the algorithm be extended by including additional control strategies without lowering the value of its other properties?

5 Conclusion

This paper claims that there is a lack of terminological precision with respect to possible subdivisions of the generation task into smaller subtasks. As a consequence, comparison of generation systems is notoriously difficult. The wheel is reinvented too often, and systems (system component) are only rarely re-used for other tasks.

The remedy suggested is to provide a general and precise definition of the distribution of concrete generation subtasks to concrete system components. This requires a general and precise definition of what constitutes a generation subtask. Using surface generation as an example, first steps are suggested

towards a classification scheme that reveals the differences and stresses the similarities between the instances.

Clearly, thousands of different approaches to NLG cannot be broken down to elementary subtasks at once. The method suggested can be applied for any composite subtask. Surface generation is among those that are considered as relatively well denned and thus could serve to test the method before it is applied for structuring a highly controversial subtask such as lexical choice.

References

[1] R. Backofen, L. Euler, and G. Goerz. Towards the integration of functions, relations and types in an AI programming language. In *Proc. GWAI,* page 1990. Springer, 1990.

[2] Ernst Buchberger and Helmut Horacek. VIE-GEN: A generator for german texts. In D. D. McDonald and L. Bole, editors, *Natural Language Generation Systems,* pages 166-204. Springer, Berlin, New York, 1988. Symbolic Computation.

[3] Stephan Busemann. Surface transformations during the generation of written German sentences. In D. D. McDonald and L. Bole, editors, *Natural Language Generation Systems,* pages 98-165. Springer, Berlin, New York, 1988. Symbolic Computation.

[4] Stephan Busemann. Using pattern-action rules for the generation of GPSG structures from MT-oriented semantics. In *Proceedings 12th International Conference on Artificial Intelligence,* pages 1003-1009, Sydney, 1991.

[5] J. Calder, M. Reape, and H. Zeevat. An algorithm for generation in unification categorial grammar. In *Proceedings of the Jth Conference of the European Chapter of the Association for Computational Linguistics,* pages 233-240, Manchester, UK, April 10-12 1989.

[6] Martin Emele. FREGE-Ein objektorientierter FRont-End-GEnerator. In K. Morik, editor, *GWAI-87. llth German Workshop on Artificial Intelligence,* pages 64-73, Berlin, New York, 1987. Springer. IFB Bd. 152.

[7] Helmut Horacek and Michael Zock, editors. *New Concepts in Natural Language Generation: Planning, Realization, and Systems.* Frances Pinter, London, 1993.

[8] G. Kempen and E. Hoenkamp. An incremental procedural grammar for sentence formulation. *Cognitive Science,* 11:201-258, 1987.

[9] M. W. Meteer. The "generation gap", the problem of expressibility intext planning. Report no. 7347, Bolt, Beranek and Newman Inc., Cambridge, Mass., February 1990.

[10] Stefan Momma and Jochen Dorre. Generation from f-structures. In E. Klein and J. van Benthem, editors, *Categories, Polymorphism and Unification.* Universitat Edinburgh, Centre for Cognitive Science and Universitat Amsterdam, Institute for Language, Logic and Information, 1988.

[11] Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira. A semantic-head-driven generation algorithm for unification-based formalism. *Computational Linguistics,* 16(l):30-42, 1990.

[12] Tomek Strzalkowski. A general computational method for grammar inversion. In Tomek Strzalkowski, editor, *Reversible Grammars in Natural Language Processing,* pages 175-200. Kluwer, Boston, Dordrecht, London, 1994.