# Reconciling Parameterization, Configurability and Optimality in Natural Language Generation via Multiparadigm Programming

Jorge Marques Pelizzoni and Maria das Graças Volpe Nunes

Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador São Carlense, 400. CEP 13560-970. São Carlos – SP – Brasil
{jorgemp, gracan}@icmc.usp.br

**Abstract.** This paper focuses on how multiparadigm – namely, constraint, object-oriented and higher-order – programming can be drawn upon not only to specify multiparameterized linguistic realization engines but also and above all to rationalize their configuration into full-fledged generation modules for specific language-application pairs. We describe Manati, one such engine whose instantiations render linguistic form to conceptual/semantic directed hypergraphs, and point out how its constraint-based concurrent architecture entails collaboration and interleaving so as to allow the definition and optimization of global quality measures.

## 1 Introduction

Natural Language Generation (NLG) refers to rendering linguistic form to input in a non-linguistic representation. As pointed out by e.g. Reiter & Dale [13], Cahill & Reape [2], Paiva [11], this can be a very complex task involving processing both linguistic (e.g. lexicalization, aggregation and referring expression generation) and otherwise (content selection and layout planning). In this paper, we are exclusively concerned with the linguistic aspect of generation, herein referred to as *linguistic realization*.

A range of linguistic realization work has been reported on so far in the literature varying in scope and depth. Nonetheless, it is a rare work that focuses on configurability issues, especially in a multiparameterization scenario. By *configurability* we mean ease of configuration, or rather, instantiation of required parameters in a disciplined, manageable, friendly manner. *Parameter*, in turn, refers to any blank whatsoever that should be filled in so as to make a generic solution into a full-fledged linguistic realization component. Possible parameters are grammars, lexicons, strategies, heuristics, etc.

In fact, there is far more usual to be material either detailing an isolated solution (an instantiation of a single parameter, e.g. Eddy [7]) or sketching a complex solution (i.e. a multiparameterized one, e.g. Stone & Doran [16]) – in either case, usually with not much regard to the discipline of instantiation. This communication takes a complementary path and attempts to focus on (i) the case of abstracting away a reusable NLG solution by (ii) (multi)parameterizing it in (iii) a hopefully highly configurable

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 352 – 356, 2005.

fashion on the basis of (iv) multiparadigm – namely, constraint, object-oriented and higher-order – programming. An additional concern is to give evidence that the concurrent constraint-based architecture thus obtained is highly collaborative and prone to yield globally optimal results by enabling interparameter synergy.

Most of this paper is dedicated to demonstrating how the above principles guided the design of Manati, a linguistic realization engine that has originally and so far been developed as a hopefully reusable component in a Portuguese-Brazilian Sign Language (LIBRAS) interlingua-based semiautomatic translation project.

The paper proceeds as follows. Section 2 motivates our linguistic realization effort by describing the requirements of the translation application that gives rise to Manati. Section 3 sketches Manati proper and shows how it addresses the three-way tug-of-war between parameterization, configurability and optimality and meets the requirements in Section 2. Finally, Section 4 draws conclusions and hints at future enhancements.

## 2 Context: Interlingua-Based Semiautomatic Cross-Modal Translation

This paper reports on partial results of a comprehensive, currently ongoing project in machine translation named PUL∅ (Portuguese-UNL-LIST deOralizer), whose aim is to reduce the cost and turnaround of translating written Portuguese into "spoken"[1] LIBRAS, the Brazilian Sign Language. PUL∅ is not intended to produce actual LIBRAS speech, but a script thereof – LIST (LIBRAS Script for Translation) – to feed an eventual speech synthesizer.

It might be assumed that **cross-modal translation** – i.e. bridging the gap between oral and sign languages – should have no special status *in principle*, were one to avail of sufficient formal descriptions of the languages involved. Even so, it is often the case that:

- sign languages are much less understood, thus lacking in description;
- one cannot rely so much on isomorphism as in intra-modal translation. For a start, one-to-one mappings between sentences is much less likely in cross-modal translation. Second, different text planning strategies arise with mode-specific referential devices [1][15]. Third, the iconic/mime substratum of sign languages usually pulverizes semantic fields, spoiling lots of usually valid direct translations;
- sign languages are usually not written, in spite of the occasional availability of a writing system. This entails serious further difficulties, as a translation project can hardly be validated without a speech synthesis/recognition module.

Finally, PUL∅ follows the semantic transfer approach to machine translation (Hutchings & Sommers [8]), using the UNL (Universal Networking Language [10][17]) as an **interlingua** or s**emantic-content representation formalism**. The UNL attempts to capture sentence meaning by means of directed hypergraphs, where (i) basic nodes refer to instances of basic "universal concepts", or Universal Words

[1] The words *spoken, speech,* etc. are employed here especially as opposed to *written*, *writing*, etc. More specifically, those words should not be regarded as necessarily implying orality.

(UWs), (ii) labeled directed edges state binary relations between node referents, (iii) hypernodes provide for recursion and nesting, or simply "complex concepts" and (iv) node attributes allow for concept "modalization".

## 3 Case Study: Manati

Manati is the linguistic realization engine all UNL-LIST conversion in PUL∅ is based on. In other words, PUL∅ includes a configuration of Manati, i.e. a module obtained by fixing Manati's parameters. The engine is fully implemented in Oz (http://www.mozart-oz.org [14][18]) and heavily draws upon the expressiveness and elegant, seamless multiparadigm integration of this language to meet its requirements. Like Koller & Striegnitz's generation work [9], it builds upon work by Duchier [3][4] & Debusmann [6], but is fundamentally distinct from the former, which strictly focuses on taming flat semantics, a non-issue here. For space reasons, a very shallow description follows; for further information, please refer to [12].

### 3.1 Parameters

Manati currently allows the rationalized configuration of ten *orthogonal* parameters in that independently and modularly defined, namely:

a) **input formalism**, which, even though restricted to hypergraph types, is free to accept any open set of UWs (node labels) and closed set of relations (edge labels) and attributes;
b) **morphosyntax:** each part of speech (POS) in the target language must be defined as a record with arity *{avm, constr}*, where feature *avm* is an attribute-value matrix (AVM) type, and *constr*, a constraint on instances of *avm*;
c) **syntactic mapping:** a specific mapper class hierarchy must be provided in order exclusively to specify the mapping of UNL (hyper)graphs onto syntactic dependency trees [3] in the target language. Roughly speaking, mappers simply convert (i) semantic nodes into lexemes (classes of lexical items) and (ii) semantic relations into syntactic roles; or, in NLG jargon, they are responsible for *lexical choice* and *aggregation*. It is worth noticing that mappers are not interested either in morphosyntatic constraints, such as agreement, or in final linear ordering of morphemes;
d) **mapping preconditions:** in order to optimize resource usage during search, part of if not all precondition checking in mappers can optionally be delegated to a specific class hierarchy. Such so-called *precond* classes are associated with mappers by lexicon data;
e) **governor-governee constraints:** a specific class hierarchy must be provided in order exclusively to tell morphosyntactic constraints on each pair of syntactically related target nodes (i.e. words or morphemes). Such so-called *gamma* classes are associated with mappers by lexicon data and have methods of the signature *Role(Parent.feats Child.feats)* invoked for each syntactic relation *Role* their corresponding mappers establish between any target nodes *Parent* (governor) and *Child* (governee);

f) **linear precedence:** a specific class hierarchy must be provided in order exclusively to determine the final ordering of target nodes and carry out whatever further tasks that might occasionally be required on mapper completion, when all direct child nodes are accessible – though not as yet fully determined – for e.g. telling further constraints. Such so-called *finishUp* classes tackle linear precedence by telling constraints relating target nodes to each of their children and children to each other;
g) any number of **oracles** – e.g. user prompts, knowledge bases, etc. – to resort to at virtually any generation stage;
h) **lexicon:** Manati's lexicon is more of a **transfer rule base**, each of whose entries is a tuple *(UW, TransList, POS, Precond, Mapper, Gamma, FinishUp)*, where *UW* is a source node label; *TransList*, a character string list of possible target language translations; *POS*, the part of speech of the elements of *TransList*; and *Precond*, *Mapper, Gamma* and *FinishUp*, classes of the homonymous types;
i) **output formalism,** i.e. how the resulting syntactic trees are to be printed out. This is highly configurable ranging smoothly from raw lists of target language words to fully structured trees by means of user-defined bracketing. Words and bracketed groups may be associated with arbitrary *Output AVMs* (OAVMs) created by *FinishUp* classes. OAVMs may be useful to add syntactic and prosodic annotations (as required by PUL∅) or even to output morphologic features, leaving full inflection of words to dedicated modules and thus downsizing the lexicon.

## 4 Conclusions and Future Work

We have presented Manati, a linguistic realization engine that attempts to equate the tension between parameterization, configurability and optimality. Manati is currently being configured to generate the Brazilian Sign Language and shall be evaluated against other generation engines in the near future. Scheduled further work on Manati includes full coverage of generation tasks – e.g. content selection and referring expression generation – and support to configurability of additional or alternative optimality measures goals and optimum search strategies.

## References

1. Brito, L. F. Por uma Gramática de Línguas de Sinais. Tempo Brasileiro Ed., Departamento de Lingüística e Filologia, Universidade Federal do Rio de Janeiro, 1995.
2. Cahill, L. and Reape, M. *Component tasks in applied NLG systems*. Technical Report ITRI-99-05, Information Technology Research Institute (ITRI), University of Brighton, 1998. http://www.itri.brighton.ac.uk/projects/rags.
3. Duchier, D. Configuration of labeled trees under lexicalized constraints and principles, *Journal of Language and Computation*, 2002.
4. Duchier, D. Axiomatizing dependency parsing using set constraints. In *Proceedings of the 6th Meeting on the Mathematics of Language*, USA, 1999.
5. Duchier, D., Gardent C., and Niehren J. *Concurrent Constraint Programming in Oz for Natural Language Generation*. http://www.ps.uni-sb.de/~niehren/Web/Vorlesungen/Oz-NL-SS01/vorlesung (accessed on Aug. 2004).

6. Duchier, D. and Debusmann, R. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the Association for Computational Linguistics (ACL)*, France, 2001.
7. Eddy, B. Toward balancing conciseness, readability and salience: an integrated architecture. In *Proceedings of the International Natural Language Generation Conference (INLG'02)*, 2002.
8. Hutchins, W. J. and Somers, H. L. An introduction to Machine Translation, Academic Press, San Diego (CA), 1992.
9. Koller, A. and Striegnitz, K. Generation as Dependency Parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2002, 17-24.
10. Martins, R. T. *A Língua Nova do Imperador*. Ph.D. Thesis, Instituto de Estudos da Linguagem, Universidade Estadual de Campinas (UNICAMP), 2004.
11. Paiva, D. *A survey of applied natural language generation systems*. Technical Report ITRI-98-03, Information Technology Research Institute (ITRI), University of Brighton, 1998. http://www.itri.brighton.ac.uk/techreports.
12. Pelizzoni, J. M. and Nunes, M. G. V. Flexibility, Configurability and Optimality in UNL Deconversion via Multiparadigm Programming. In: I. Boguslavsky, J. Cardeñosa, A. Gelbukh. *UNL, other Interlinguas and their Applications*. Research on Computer Science, 2004, to appear. http://www.CICLing.org/2005/UNL.htm.
13. Reiter, E. and Dale, R. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.
14. Schulte, C. *Programming Constraint Services: High-Level Programming of Standard and New Constraint Services*, Lecture Notes in Computer Science Series, Springer-Verlag, 2002.
15. Speers, d'A. L. *Representation of American Sign Language for Machine Translation.* PhD. dissertation, Graduate School of Arts and Sciences, Georgetown University, 2001.
16. Stone, M. and Doran, C. Sentence planning as description using tree-adjoining grammar. In *Proceedings of the Association for Computational Linguistics*, 1997, 198-205
17. Uchida, H., Zhu, M., and Della Santa, T. *UNL: A Gift for a Millennium.* Institute of Advanced Studies, University of the United Nations, 1999. http://www.undl.org/publications.
18. Van Roy, P. and Haridi, S. *Concepts, Techniques, and Models of Computer Programming*, MIT Press, 2004.