Controlled Realization of Complex Objects by Reversing the Output of a Parser

David D. McDonald
Gensym Corporation, 125 CambridgePark Drive
Cambridge, MA 02139 dmcdonald@gensym.com

Abstract

This paper is a study in the tactics of content selection and realization at the micro-planning level. It presents a technique for controlling the content and phrasing of complex sentences through the use of data derived from a parser that has read through a corpus and taken note of which variations do and do not occur in the realization of the concepts in the genre the corpus is taken from. These findings are entered as annotations on a new representational device, a 'saturation lattice', that provides a systematic way to define partial information and is the jumping off point for the micro-planner. The generator and parser are both based on a declarative, bi-directional representation of realization relationship between concepts and text.

Keywords: generation, bi-directional, realization, model-driven

1. Motivations

If research on natural language generation ('nlg'), as it is understood by the audience of this workshop, is to garner support from commercial interests and move beyond sponsored research and academics, it will have to provide a set of commercially valuable, sophisticated tools that can achieve results well beyond those of a good database system's report generator ('dbg')—our practical competition. In the early days of work in nlg it was difficult enough to say "*The Knox is enroute to Sarasibo*", but any dbg can say that today. NLG techniques now ten years old can apply techniques for subsequent reference and aggregation and produce reasonably fluent paragraphs from sets of simple propositions, e.g., "*The Knox, which is C4, is enroute to Sarasibo. It will arrive …*"; see Hovy (1990). A tool set that can provide that sort of smoothing over the output of a dbg is a good start, but we need to do better.

When we look at human report generators, such as journalists or the authors of press releases, we see two skills that are utterly beyond the abilities of a dbg: compact, syntactically rich sentence structure, and the ability to realize partially saturated relations. The first ability is familiar to the nlg community and is typified by state of the art systems such as Robin's Streak (Robin & McKeown 1996). The second is best shown through an example. Consider the following sentence, [1] which was produced by a computer program applying the techniques describe in this paper.

[1] This is an example of the sort of sentence one tends to see at the beginning of the second paragraph of a press release reporting a company's quarterly earnings: the one that contains the primary information to be conveyed

(1) *"Net income in the third quarter ended Nov. 30, 1995 totaled $55.9 million, a 34 percent increase over net income of $41.8 million in the comparable quarter of the preceding fiscal year."*

This sentence is missing one of the terms required to make it a complete (saturated) relation, namely the company whose earnings these are. The person who originally planned this sentence recognized that the identity of the company would be 'obvious from context' and that it should be omitted here in order to be cohesive.

In a corpus of such earnings reports we see the primary, 14 term 'financial report' relation sliced and diced in myriad ways, with variations in what terms are omitted, how they are grouped into maximally projecting phrases, how they are distributed into sentences, and which terms are used as the head. Almost without exception each instance is the realization of only part of the relation, and the skill of the authors is in appreciating what combinations (partial saturations) they can legitimately express given the style that the intended audience is accustomed to. Capturing this skill in a natural language generation system will take us in the right direction visa vie our commercial competition. The question, of course, is how to do it.

The problem naturally divides into two aspects. Following Levelt (1989), these are macro-planning: the selection of what sets of terms are to be included, how they are to be distributed within the text as a whole, and most of the lexical selection; and micro-planning: determining the structure of the text and the apportioning of the terms into phrases, clauses and sentences that properly realize the macro-planner's specifications of salience while maintaining cohesion, avoiding unintended redundancy, and the like.

In this paper I propose how to solve, even finesse, the micro-planning[2] problems posed by a complex yet mundane domain such as earnings reports ('ern'), namely by reversing the output of a parser. In broad outline this is done as follows.

The parser[3] is augmented to record the manner in which the phrases that it parses have been realized. This record is couched in terms of the same set of linguistic resources as used by the generator, in this instance a TAG and a set of reversible mapping rules linking concepts and resources. After parsing a corpus of texts, the result is an annotation of all the particular ways in which the phrases (those that were understood) have been realized (and implicitly the ways in which they have not been realized). This lattice of realization types is deployed in the micro-planner by starting with the content selected by the macro-planner (some structure over partially saturated relations), finding the point in the lattice that corresponds to (each of) the partial relations, and then selecting from among the strands of alternative realization types within the lattice according to simple notions like theme vs. background. The selection is read out to create the text structure, and the rest of the generation process (surface realization) proceeds normally.

2 I frankly do not understand the basis of the macro-planning evidenced by the authors of ern articles. Perhaps 10% of the variance can be explained by the occasional goal of hiding bad news, but otherwise the pattern of the distribution and inclusion of information is anything but obvious; I suspect that much of the decision making is a rote form of copying what other authors write. A careful longitudinal study of the structure articles from known authors might shed some light on the problem. If pressed, I would make a macro-planner from reverse engineered schemas with a random element.

3 I will use the term 'parse' and 'parser' as a convenient short-hand for designating the full natural language understanding system that is actually being used. This does not stretch the usual senses of the term too much since Sparser does do its semantic interpretation at literally the same time as its parsing into syntactic phrases. The key difference from the usual parser is that the end result is a set of objects in a domain model rather than just a parse tree.

2. Architecture

The overall design of the generation architecture used here is as described in Meteer 1992, following a set of principles laid out in McDonald, Pustejovsky, and Meteer 1988. It is a message-driven system that begins with structures taken directly from the underlying system for which it is speaking, realizing them monotonically via a succession of progressively more linguistic represen-tational levels through the use of an extensive set of abstract linguistic resources ultimately grounded in a TAG grammar of English.

The source structures are represented in a Kl-one derived system called Krisp (McDonald 1994s), and the parser that produces the corpus-based lattice of realization types is Sparser (McDonald 1992, 1994a)—two complete, mature systems. We will introduce only as much information about them as necessary to support the rest of the discussion. Much of this paper will be devoted to an extension to Krisp, a 'saturation lattice', that is the basis of this technique of micro-planning by reversing the parser's output.

At present the new generator, to be christened "Magellan", implements only the micro-planning and later stages of generation. There is no speaker in a situation with genuine motivations, without which a generation system (or certainly a fully-articulated theory of generation) is incom-plete. In its stead, as a way to exercise the micro-planner, is a windup toy—a largely graphical interface that permits the experimenter to stipulate the input and decision criteria that a macro-planner would have produced and see what text results.

The generation process starts with the Krisp units representing a full relation. We select by hand the fragment of it to be expressed and some simple information-structure parameters. Then the micro-planning mechanism described here is deployed to populate a Text Structure representation, which has been excerpted directly from Meteer's Spokesman system (1992). Spokesman's mechanisms then read out the Text Structure to create the TAG derivation tree that is the input to Mumble (Meteer et al. 1987), which in turn produces a TAG-based surface structure and from that the eventual text.

2.1 Categories in the domain model

The micro-planner's task is to realize a single, highly-structured, compositional relation as a Text Structure. To illustrate the resources it uses to do this, consider these two (made up) sentences:

(2) " *GTE owns BBN*."
(3) " *BBN is a subsidiary of GTE*."

These express the same information. They should be represented by the same object in the domain model. Which of the two alternative realizations of this object a speaker will choose is a question of which of the two companies they decide to make the theme.

The expression below defines the type, or 'semantic category', that these texts instantiate. This is a Lisp expression that is evaluated at the time the domain model is loaded. It makes reference to several things that will already have been defined, notably the category's parent super-category in the taxonomic lattice, named 'owner-owns-owned',[4] and two tree families in

[4] There is an unfortunate tendency for names like these to dominate how a person thinks about concepts. Simply because names are necessarily (if they are to be useful) comprised of regular, suggestive natural language words, they can too often cloud the mind to the possibility that there are many different ways to realize the same conceptual content (see, e.g., Elhadad et al. 1996) This is something always to be guarded against.

The choice of names for all the objects in this domain model and grammar is arbitrary and strictly for the convenience of the human designer. Because they are implemented in terms of objects and pointers, the names are the grammar. Once the expression has been executed, the result is (1) a category object representing the type in the domain model; (2) a set of phrase structure rules in the semantic grammar used by the parser (based on the 'realization' field); and (3) a 'saturation lattice' (based on the 'binds' field) which is described below. (The realization field is only sketched here since it's values would make little sense without the background that will be supplied in a later section; here simply observe that there are two alternative classes of realizations available to objects of this category, each with its own family of syntactic trees and own primary, content-bearing lexeme.)

```
(define-category co-owns-co
  :binds ((parent . company)
          (subsidiary . company)
  :specializes (owner-owns-owned (owner . parent)
                                 (owned . subsidiary))
  :realization
    ( (:tree-family transitive/passive
        :mapping ( … "own" … ))
      (:tree-family nominal-binary-relation
        :mapping ( … "subsidiary" … ))))
```

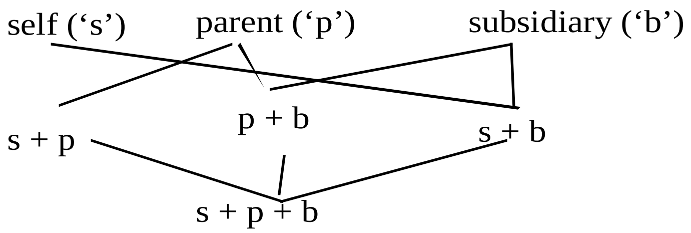
This category corresponds roughly to a KL-One concept with two slots named 'parent' and 'subsidiary', whose values are restricted to objects of type (category) company. The 'specializes' field indicates how this category is knit into the taxonomic lattice and the inheritance path of its slots, which are termed 'variables' in Krisp.

2.2 Saturation lattices

The reference model for Krisp is the Lambda Calculus, where there is a well articulated notion of expressions that have only bound a few of their variables to specific values and left the others open. Such 'partially saturated relations' have a first class representation in Krisp. This representation is supported by a lattice 'below' each of the categories of the normal taxonomic (is-a) lattice. This lattice defines types for all of the category's sets of possible partial instantiations and provides a representational anchor for the realization annotations that the parser lays down and the micro-planner uses.

As shown below, a saturation lattice consists of a linked set of nodes that represent all the possible combinations of bound and open variables of the category, including a pseudo-variable 'self' that allows us to include the category itself. (This variable is usually realized as the verb of a clause or the head noun of a np.) Notice the use of single-letter abbreviations for the variables when they appear in multi-variable nodes.

not even used at runtime, and serve only to provide a way to uniquely designate the objects in the written expressions that are needed to initially define them when a direct manipulation interface is not being used



In the present example, the lattice is relatively simple with just three levels.[5] At the top we have the information states where one of the three variables is bound and the other two open. Next these nodes ('lattice points') converge to form a level where each possible combination of two bound and one open variable is represented. These then join to form the lattice point that represents the state where the relation is fully saturated, i.e. all of its variables are bound. This bottom node in the lattice is annotated by the various contexts in this category has appeared as a contiguous phrase: as a whole sentence, as a subordinated clause, as a reduced phrase in a conjunct, etc. The abstract resources for these contexts correspond to attachment points in Mumble and usually adjunctions in a TAG.

The saturation lattice is used in the parsing direction to provide a set of indexes that anchor and organize partial results so that phrases with the same denotation are directed, through the paths of the lattice, to the same model-level object.[6] In the generation direction it is used to inform the micro-planner of the realization potential of each of the partial relation types. The basis of this information is a set of annotations on the lattice points that record what realizations the parser has seen for that combination of bound and open variables and in what contexts they have occurred.

These annotations are recorded or elaborated every time the parser reads a text that has instances (partial or full) of that category. For example, if we imagine that the parser has seen just examples 2 and 3, then, roughly speaking, it will have recorded that the combination of self and subsidiary ('s+b') can be realized as a VP ("*owns BBN*") and that s+p can be realized as a possessive NP ("*subsidiary of GTE*"), but it will have no reason to believe that there is a direct (self-contained) realization of p+b since it has never seen them together as the only content elements in one phrase.[7] Should it later read a text that includes the phrasing "*…BBN, a subsidiary of GT …*" (or for that matter "*…IsoQuest, a subsidiary of SRA…*"), it will extend the annotation on the s+b lattice point to include that relative clause pattern.

---

5 If a category defines N variables then its saturation lattice has N+1 factorial nodes over N+1 levels. For the initial financial example, which is the realization of a 14-tuple, this means its lattice could in principle contain several billion nodes distributed across 15 levels. It obviously does not, and the reason is simply that the lattice is only instantiated as the parser finds particular combinations of variables. Because the compartmentalization of the elements of the 14 tuple is high and their actual patterns of combination relatively few, the lattice has not quite a hundred nodes as this is written.

6 This is the way that Krisp implements the 'uniqueness principle' articulated by Maida and Shapiro 1982 whereby every individual has a single representation in the domain model regardless of how often or in what context it occurs.

7 Given our knowledge of English grammar, we can imagine the gapping construction where this would occur: "GTE owns BBN and IBM Lotus", but it has not occurred in this corpus, therefore it is not included in the realization patterns recorded in the saturation lattice.

2.3 Strands of annotations

The annotations on the lattice points are not independent. They are linked together in strands running down through the saturation lattice that reflect the parser's derivation tree as it accumulated successively larger portions of the text to the head-line of its maximal phrases, binding one variable after another in a particular order and thereby following a particular path down through the lattice. Each derivation tree that has been seen for a given combination of variable bindings is a separate strand. It is the micro-planner's job to choose one of these strands based on the properties of the individual nodes it is comprised of (one for each binding). Having selected a strand, it then creates (or extends) the Text Structure by taking the concrete relation that it has been given by the macro planner and using the strand as a recipe for introducing the objects in the relation into the Text Structure. They are added one by one as the micro-planner reads out the strand from top to bottom, at each step adding the object that is bound to the variable that was added at that lattice point.

This use of strands lets us capture some delicate co-occurrence constrains for free because the realizations of the terms in a relation (and their constituent terms) are not independent but must follow the pattern defined by the selected strand. In the ern domain consider the common alternation in the placement of the 'fractional time period' term ("*quarter*", "*nine months*", etc.) with respect to the 'financial item' term ("*earnings*", "*turnover*", etc.) in a phrase that anchors the reporting period to a particular point in the calendar. We typically see phrasings like #4 or #5 but never the combination in #6.

(4) "*…quarterly earnings for the period ending March 31…*"
(5) "*…earnings for the quarter ending March 31…*"
(6) * "*…quarterly earnings for the quarter ending March 31…*"

The question is how is #6 to be avoided. The source for the anchor adjunct includes the fact that the 'period' is one fiscal quarter; what is the constraint mechanism that suppresses the expression of the actual time period when it has been stipulated, thematically, to appear with the head noun?

The answer is simply that the pattern of realizations in #6 has never been seen by the parser and consequently there is no strand for it in the lattice. The parser has done all the work and the micro-planner reaps the benefits without the need for any sort of active constraint propagation mechanism. It just reads out the template that the parser has provided.

3. Bi-directional Resources

This technique is predicated on the parser and generator sharing the same model of the language so that the observations of the parser can be capitalized on by the generator. Such reversibility is a common, if seldom deployed, idea in computational linguistics (see papers in Strzalkowski 1994). Here the turn-around point is in the domain model that represents what the parser understood rather than at the typically chosen level of logical form (see, e.g., Shieber et al. 1990). This has choice has considerable advantage in leverage because the model can be very abstract, and in practical engineering since the domain model is invariably developed by reverse-engineering actual texts. We earlier saw an example of a category in the model. Now we turn to the resources that define the (bulk of) the linguistic knowledge.

The grammar is a TAG, given in its usual form on the generation side in Mumble, but a very different one on the parsing side.[8] For the parser, the TAG is reorganized (by hand) by

[8] In the sense of Appelt (1988) this is a 'compilation'-based treatment of bi-directional processing.

sectioning the trees horizontally into patterns of immediate constituents in the manner of Schabes and Waters (1992) as shown in the example below,[9] which is followed by the full detail of the part of the realization field of co-owns-co that goes with this tree family; syntactic categories on the left side of the mapping are replaced with the semantic categories on the right.

```
(define-exploded-tree-family  transitive/passive
  :binding-parameters ( agent patient )
  :labels ( s vp vg np/subject np/object )
  :cases ((:theme (s  (np/subject vp)
                   :head right-edge
                   :binds (agent left-edge)))
          (:theme (s  (np/subject vg)
                   :head right-edge
                   :binds (agent left-edge)))
          (:final (vp  (vg np/object)
                    :head left-edge
                    :binds (patient right-edge)))
          (:theme (s  (np/object vg/+ed)
                     :head right-edge
                     :binds (patient left-edge)))
          (:subordinated (vp (vg/+ed by/pp/np/subject)
                          :head left-edge
                          :binds (agent right-edge)))))

 (:tree-family transitive/passive
  :mapping ((agent . parent)
            (patient . subsidiary)
            (s . self)     ;; i.e. the category co-owns-co
            (vp . self)
            (vg . "own")
            (np/subject . company)
            (np/object . company)))
```

The annotations left by the parser on the nodes of the saturation lattice are essentially just pointers back to the rule in the 'exploded' tree family that it applied when it added the constituent that bound that term. Thus the annotation includes the label, such as theme or final, that characterizes the comparative status of the np term in the rule, making it available to the micro-planner to aid in its choice of strands.

4. Conclusions

By using the saturation lattices to guide its micro-planning process and control its choice of mappings from conceptual objects to linguistic resources, this system can readily produce the long, syntactically elaborate texts that a common in commercial news sources (after all, the parser has virtually laid down a template for the generator to follow), and by being trained on the appropriate corpus it can do so using the style that is natural to the genre. The macro-

9 Note that it does not include rewrite expressions for any of the 'oblique' forms that clauses are subject to (relatives, reductions under conjunction, clefts); these are standard to clauses of all sorts and the parser handles them through a common set of rules of a different kind.

planner can freely rearrange or factor the information that the parser read when it goes to generated new texts, but it will be unable to violate the norms of how that information is expressed simply because it will be using no source other than the saturation lattices to make the final realization decisions.

These sub-category saturation lattices are (to my knowledge) a new representational device, one that permits us to make use of the knowledge of how particular types of information are expressed that any parser implicitly deploys. As implemented, the process of annotating the nodes is a completely automatic side-effect of incrementally indexing the information contained in partial phrases during the course of a parse. Because the process is ubiquitous and covers all the particular facts[10] that are acquired from the reading (and these are all the facts the system knows), the planning that is done during generation is freed from needing to worry about fine-grained details, is assured that its text plans will be expressible, and can concentrate on the substantive issues of what particulars to include and where to place the emphasis.

Finally, this approach may lead us to a psycholinguistic model of how it is that people so readily adapt their style of writing or speaking to the patterns what they have recently heard or read.

References

Appelt, Doug. 1989. “Bidirectional Grammars and the Design of Natural Language Generation Systems”. in Wilks (ed.) *Theoretical Issues in Natural Language Processing,* Lawrence Erlbaum, Hillsdale, New Jersey, pp. 199-205.

Elhadad M., K. McKeown, J. Robin. 1996. Floating Constraints in Lexical Choice. *Computational Linguistics.*

Hovy, Eduard (1990) “Unresolved Issues in Paragraph Planning”, in Dale, Mellish & Zock (eds.) *Current Research in Natural Language Generation*, Academic Press, New York.

Levelt P. 1989. *Language Production.* MIT Press.

Maida A. & S. Shapiro. 1982. “Intensional Concepts in Propositional Semantic Networks”. *Cognitive Science* **6**, pp. 291-330.

McDonald D. 1992. “An Efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts”. proceedings of the 3d Conference on Applied Natural Language Processing (ACL), Trento, Italy, April 1992, pp. 193-200

McDonald D. 1994a. Reversible NLP by linking the grammar to the knowledge base. in Strazalkowski 1994, 257-291.

McDonald D. 1994b. ‘Krisp’ a representation for the semantic interpretation of texts. *Mind and Machines* 4, 59-73.

McDonald D., J. Pustejovsky, M. Meteer. 1988. Factors contributing to efficiency in natural language generation. In Kempen G. (Ed.) 1987 *Natural Language Generation*, Martinus Nijhoff, Dordrecht, 159-181.

Marie Meteer (1992) *Expressibility and the Problem of Efficient Text Planning*, Pinter Publishers, London.

Meteer, Marie W., David McDonald, Scott Anderson, David Forster, Linda Gay, Alison Huettner & Penelope Sibun, (1987) *Mumble-86: Design and Implementation*, TR #87-87 Dept. Computer & Information Science, UMass., 174 pgs.

[10] As opposed to the generic knowledge embodied in the semantic categories of the domain model.

Robin, J and K. McKeown. 1996. Empirically designing and evaluating a new revision-based model for summary generation. *Artificial Intelligence* 85, August.

Shieber, Stuart, Gertjan van Noord, Fernando Pereira & Robert Moore (1990) “Semantic-Head-Driven Generation, *Computational Linguistics* (16)1, March 1990, pp. 30-42.

Strzalkowski T. 1994. *Reversible Grammar in Natural Language Processing*. Kluwer Academic.