**In the proceedings of the RAGS Workshop, November 12-13, 1999, Edinburgh**

# A Rational Reconstruction of Genaro

David D. McDonald
14 Brantwood Road, Arlington MA USA
davidmcdonald@alum.mit.edu

The Genaro text generation system was developed by Jeff Conklin for his Ph.D. thesis at the University of Massachusetts (Conklin 1983, Conklin & McDonald 1982). It was based on the hypothetical but plausible output of a vision system that examined pictures of houses. In keeping with the trend of the vision research then underway at UMass (Parma et al. 1980), the scene-understanding system was taken to be able to construct a set of facts or observations about the picture that it was viewing, and to assign each of these facts a rating according the relative visual salience of that element of the picture as compared with the others. Genaro's task was to render this output as a paragraph of fluent English text that captured this salience.

Genaro was the first substantial system to be layered on top of the original version of Mumble (McDonald 1980, 1984). In so doing, it exposed technical and theoretical flaws in that design, flaws that influenced the later work of my group, ultimately leading to the development of Meteer's theory of Text Structure and expressibility (1992) and to our reference model of generation (McDonald, Meteer & Pustejovsky 1987).

The goal of our model (which we will abbreviate as MMP) was to establish a theoretical basis by which one could evaluate alternative generation architectures in terms of their relative efficiency. We were trying to bring to generation the same kind of measures as have been available to parsing from its earliest days.[1] Because realistic language generation is always done at the behest of some speaker (human or artificial), MMP takes the essence of the task to be how to find a mapping between the situation that the speaker takes itself to be in and some utterance[2]. Frequent, simple situations (greeting a friend while passing in the hallway) should have simple, direct mappings; the unfamiliar and complex (a job interview, a new talk) may require extensive planning and replanning.

This position paper begins by describing how Genaro would be rewritten today given the experience and theoretical developments of the last fifteen years.[3] This

[1] E.g. Earley's algorithm has a worst case complexity of O2N3 in the size of the grammar and the length of the input text respectively and is O2N for deterministic grammars.

[2] A utterance here may be of any length so long as it is a single ply in the larger discourse. Situations are composite instances of instance types following Barwise & Perry (1983, Devlin 1991).

[3] We expect to do an implementation over the course of the next six to nine months. Particulars described here have either been implemented in other instantiations of our paradigm, such as Meteer's, or come from the implementation of other generation systems that we have completed in recent years, or are expressly indicated as how we anticipate the implementation will go.

reconstruction will provide a concrete example of how we construe the generation process today (a small evolution from our model of 1987), and from its relationship to the reference model proposed by the RAGS group.

## 1.0 Genaro's Original Architecture

Genaro produced single paragraphs of, for its time, quite fluent prose starting from a stream of simple propositions representing the attributes of the objects it described. The same source could be made to yield literally dozens of alternative paragraphs by varying a set of parameters that governed such things as how much detail was provided for each object and the length of sentences, (unpublished work by James Pustejovsky). Below is a canonical example of its output. The parameter settings that led to it seem to provide one of the best fits to the set of human-generated descriptive paragraphs collected in Conklin's study at the beginning of his work (see Conklin et al. 1983).

> "*This is a picture of a white, New England house with a fence in front of it. The door of the house is red and so is the gate of the fence. There is a driveway next to the house. Next to the driveway are some trees. In the foreground is a mailbox.*"

### 1.1 Representational Levels

These texts were produced by an architecture comprised of three levels of representation that were constructed sequentially and incrementally by three successive processes. We will begin by describing the levels. The first was a representation of the objects and relationships depicted in the picture being described. This level was logically an internal part of the vision system in whose service the generator was acting (the 'speaker). However, since the judgements of the vision system were only simulated in Genaro, the level was actually a standalone collection of objects, essentially a set of individuals and simple propositions over them. The objects were implemented using a local treatment of KL-ONE. These included a set of individuals (nexuses) such as the house, the fence, the driveway and such; and there was a set of types for spatial relations and simple attributes (properties). When the type distinctions do not matter we will refer to these as "units". Each object and proposition included a numerical salience rating. Here is a sample:

```
#<House house-1 1.0>
#<Gate gate-1 .7>
#<in-front-of house-1 fence-1 .85>
#<two-story-building house-1 .38>
```

At the time, we talked about this as the 'message level' of the generator. Now we would call it simply a semantic model. 'Semantic' because the minimal representational units that comprise it are all expressible (Meteer 1992). Each could, in principle, be realized in isolation as an utterance, albeit a very small one. In RAGS terms, it would appear to be concrete semantic stuff, a judgement we will discuss at some length later.

The second level of representation was the result of rhetorical decisions about how to assemble the units into sentence-length directives to Mumble. These directives were referred to as 'r-specs' (rhetorical specifications). They incorporated units from the model and also special objects that Conklin dubbed 'rhetoremes', i.e. units that have no

correspondence in the model and are inserted strictly for their rhetorical function; the name is to indicate that they are the smallest units of rhetorical information that the system has.

Here is the expression for the r-spec that would be realized as “This is a picture of a white house”

```
(RSPEC NO!
  BODY
   (ELMT1 RHETOREME introduce (house-1)
      (house-1 (SUPERC house)
        NEWITEM))
   (ELMT2 PROPERTY   color-of (house-1 white-1)
      (house-1 IBID)
      (white-1 (SUPERC white))))
```

In RAGS terms this has to be abstract syntactic stuff since it is the input to syntactic realization. It is certainly a text plan.

The third level of representation was Mumble’s surface structure. This was and continues today to be a functionally annotated phrase structure, not too dissimilar in concept from the surface level of representation used in a systemic functional grammar such as Davey’s (1974) or Kempen & Hoenkamp’s (1987).. Each node in the tree is labeled with features and active triggers. We say that the node is contained within a ‘slot’, where the features in the slot can constrain what realizations are permitted, such as forcing clauses to be tenseless. Slot features are also the source of function words, and the source of punctuation as well as richer orthographic effects such as markup tags. Details of this level are given in Meteer et al. 1987.

### 1.2 Processes

Conklin called the first (earliest) of the three processes operating over these two representational levels ‘iterative proposing’ (“IP”). Its job was first to notice the relative salience of the different objects in the picture, and then for each object in turn to notice the relative salience of each of its attribute or its relationships to other objects given their relative visual salience. This ordering defined an object stream that could, in principle, enumerate every unit in the underlying system’s model of the picture except for the intervention of the parameter settings that define cut-off points once an object or one of its attributes drops below the indicated salience level.

The second process, the consumer of this stream, was a paragraph planner that applied a simple set of rhetorical rules to assemble successive r-specs. Some of these ensured that certain properties were always included, others were sensitive to salience; examples include property-salient-color, relational-salience, condense-property. They functioned by guiding the successive units from the stream into the appropriate parts of the growing r-spec.

The rhetorical rules were grouped into ‘packets’ following Marcus (1980). Which packet was active at a given moment determined what rules were available. Four packets were defined: Introduce, Elaborate, Shift-topic, and Conclude, which were controlled by a simple state-machine. The first and last amount to a schema-style of discourse

organization: the planner always started in Introduce and ended with Conclude (which emitted a conventional description of the weather depicted in the picture). Introduce created a fixed skeleton specification for the first sentence ("This is a picture of ___") and then automatically shifted to the Elaborate packet.

The variable governing the state machine was the 'current-item'. This identified one of the objects in the scene, initially the house because it had the highest salience. IP would pass a succession of units that described the current-item to the Paragraph Planner in the order of their relative visual salience until the salience of the next unit fell below a certain parameter setting. The units would be considered by the rules in the Elaborate packet, which would add them one after the other to the ongoing r-spec.

Since r-specs are realized as sentences, there was a need for a calculus within Elaborate that would determine when the proper size for an r-spec had been reached and that r-spec passed to Mumble and another one started. This was done by assigning a textual weight to each type of unit and maintaining a running total as units are added. An object had weight 0; properties were .5; relations were 1.0; rhetorical elements were 2.0; and condensations (clause aggregations) were .5. So for example "The house has a red door" would 'weigh' 1.5, while "In front of the house is a white picket fence with a red gate and in front of that is a sidewalk with a person walking on it" would have a weight of 8.0: four properties plus four relations plus a rhetorical element for the conjunction. Experiments showed that the optimal weight for a r-spec/sentence was 3.5.

When the next unit from the stream would push the weight of the current r-spec about the parameter setting for sentence size that r-spec would be finished and sent off to Mumble and another r-spec started. This would also happen when the next unit was a meta-unit (rhetoreme) indicating that the limit on the description of the current object had been reached and the paragraph planner was now to change to using the Shift-topic set of rhetorical (text-constructing) rules to introduce the new current-object.

In RAGS terms, Genaro's paragraph planner is an odd duck. It is a text planner,  But is operates over just one level of representation. There is no explicit rhetorical structure, only immediate actions taken for rhetorical effect. There was no document structure, just actions taken to induce the linguistic realization component to use constructions that would signal that a new segment the text, discussing a new object had been reached. It's output was passed directly to the realizer, making it an abstract syntactic representation by definition.

The third process was Mumble (the '1980' version), which itself was comprised of three interleaved coroutines. Its first received r-specs from the Paragraph Planner one at a time as each was finished, interpreted it as a traversal pattern, and walked through it top-down mapping each unit in the r-spec to a well-formed fragment of surface structure that embedded the unit's sub-elements at its leaves. The structure-constructing actions of this first coroutine were interleaved with the operations of the second, which took the surface structure (functionally annotated phrase structure tree) built by the first as soon as any was ready and traversed it in prenext order (top down and left to right). The first process resumed whenever the traversal reached unrealized units, and the surface structure was extended dynamically.

Embedding the last stage of a unit's realization within the read-out of the surface structure was important to correctly appreciating all of the relevant context since it is only at that point that one can know precisely which units have been realized and what form their realization took, e.g. in order to make correct judgements about pronouns.

A third co-routine within Mumble takes the stream of lexemes generated by the traversal and utters the words of the text. Content words are the terminal leaves of the surface structure. Function words, including punctuation, arise from the functional context that governs the phrase structure's non-terminals and are added to the word stream either as a node is entered or after the traversal of its sub-tree is finished. This word-level process maintains its own level of representation, a fifo buffer of successive words, against which it does such things as the morphological adjustments that implement aux movement, adds plural and tense morphemes, determins the correct case of pronouns, or collapses redundant punctuation (e.g. given a comma followed immediately by a period emit just the period.

None of the structure internal to Mumble ever appears on RAGS' radar as far as I can tell. One possible exception is the operations that occur when embedded semantic units are encountered during the surface structure traversal. In principle, these could do an extensive amount of text planning (and in early work with Mumble they did), however in Genaro the realizations are virtually all anticipated and mapped out via structure in the r-spec.

## 2. Genaro Reconstructed

What is wrong with the way Genaro was done seventeen years ago, other than changes in style? What would we change in its design today that would be a genuine architectural improvement and not just a change for its own sake. Two problems became apparent shortly after Conklin finished his thesis and we began to apply Genaro's architecture to different domains.

The first and most serious problem is that Genaro tried to leap too large a gap in mapping directly from descriptive propositions to a linguistically complete surface structure (from the input to the Paragraph Planner, taken unit by unit, to Mumble). For one, the mapping tables that provided the resource knowledge for realizing the propositions had to incorporate far more syntactic detail than actually needed to do the job: To determine whether a given item type can be expressed as, e.g., an attributive adjective or via a copula, we only need to know about major syntactic categories; knowing the shape and features of the corresponding elementary trees in the TAG is completely unnecessary (and violates the MMP dictum of minimal commitment); yet in the original Genaro those two things (or rather their 1982 equivalents) were inextricably bound together.

The solution to this problem in the evolution of thought in my group was to develop a new level of representation that encoded just enough linguistic information to force critical decisions about the choice of linguistic type to be made in such a way that guarantees that they can be made indelibly, i.e. Meteer's Text Structure. The Paragraph Planner now feeds Text Structure rather than surface structure. The r-specs it used to produce can be dispensed with in favor of constructing Text Structure.

Moving the linguistic constraints on assembly to the Text Structure allows simplifications to be made to the treatment of the surface structure since it now has a smaller burden to bear. The choice sets of the original version of Mumble had, of necessity, covered the full range of possible surface realizations of a semantic unit. They had to since they were the only source (representational level) of linguistic resources available in the system. The problem is that when you try to develop any sort of capacity for sophisticated reasoning over possible alternatives, the syntactic details that have to be dealt with overwhelm the attempt.

The separation of choices from execution allowed us to sharpen the grammar that Mumble uses to be virtually a standard Tree Adjoining Grammar. Its choice sets are now identically TAG tree families – all the options are now projections of the same syntactic category. All that remains are contextually or informationally governed variations such as infintives vs. tensed forms, clefts vs. 'there' constructions, and so on.

The result of this simplification, Mumble-86, is uninteresting in the RAGS model since it is a terminal process/level – concrete syntactic structure – and does not emit / export in RAGS.[4] The interesting question is the nature of Text Structure.

### 2.1 Text Structure

Structurally, Text Structure is a dependency tree of nodes connected by directed links. Its nodes are the same units of content that came out of (in this case) IP. At this level, however, the units are viewed as instances of a carefully chosen of semantic types. These types can be seen as participating in a grammar of linguistically abstract 'expressive categories', categories that embody the distinctions that are required to guarantee that any text structure formed from them will be expressible because they impose constraints on what compositions of individually expressible units are valid.

When the generator starts from the conceptual representations of programs that were not designed with natural language in mind, the effort to establish the expressive types of its units can be complex and arbitrary, requiring its own mapping procedures. Here we will have the luxury of using a source representation of our own design[5], expressly tailored for the semantic problems that are particular to the analysis and construction of (English) texts. We will be able to treat the expressive categories of Text Structure as an Upper Structure in the domain model.

As we will use them, the links in the dependency tree will annotate (define) the constraints on what surface linguistic relationships can be used to express the corresponding combinations of content when it is projected to surface structure. Meteer used annotations like matrix, head, argument, and adjunct. We will use those relations as well as somewhat more specific and surface-oriented relations such as modifier, qualifier, or adverbial. This reflects our expectation that we should be able to assemble a grammar for Text Structure using example-driven methods by projecting back from the combinations found in the texts in domain specific corpora as analyzed by our language understanding system, Sparser (see, e.g., McDonald in press).

---

[4] It could. A fully elaborated phrase structure with annotations indicating focal or contrastive information and information structure should be a valuable resource for driving synthetic speech.

[5] KRISP. See McDonald 1994, in press, and discussion later in this paper.

The addition of another representational level implies the need for another mapping process, or, as MMP would put it, a process to read out the Text Structure and assemble the next level down. This process will be engaged once the Paragraph Planner decides that adding another unit to the sentence it has been assembling would make it exceed its parameter setting for content and structure and begins to move to assemble the next sentence. Following Meteer's lead, this target of this readout process is what Mumble-86 calls an 'lspec' ('linguistic specification').

An lspec is another dependency tree. It's structure will always be isomorphic to the structure of the Text Structure modulo some linear precedence decisions. Its nodes however are the elementary trees of Mumble's Tree Adjoining Grammar. Formally it is a TAG dependency tree – the standard starting point for TAG derivations. There are some interesting differences from a standard TAG, but they are not relevant to our concerns here. If someone wanted to construct a source to drive Mumble-86 directly, this TAG dependency tree is what they must assemble.

### 2.2 Buffering the unit stream

Returning to the difficulties in Genaro's original architecture, a second problem came from the fact the Paragraph Planner had to process each unit completely as soon it arrived before even the type of the next unit in the stream could be known. This included the judgement about whether the sentence under construction had accumulated enough weight, syntactically, once that unit in hand was added, leading to a new sentence being started regardless of what might follow in the stream. This often led to the what we called 'orphaned' units, where only a single unit remained in the description of the focal object, say the house's color, before the arrival of a 'change focal object' unit (rhetoreme) would arrive and the Paragraph Planner would be forced to start a new sentence. The result could be a sentence like "It is white", which would be completely out of the norm for the intended sentence length and complexity.

The fix for this problem is the introduction of an additional representational layer, albeit a trivial one, between IP's unit stream and the Paragraph Planner. This is just a 'staging level', where several units at a time are buffered to give the Paragraph Planner the option to look at more than one unit at a time when making its decisions. A buffer of three units should be sufficient given our work with aggregation. The arrival of a change focus rhetoreme into the staging level acts as a block against further look-ahead since it will necessarily force a sentence break.

## 3. What happens where?

Up to this point we have touched on how Genaro deals with the functional tasks that RAGS has focused on, but we should now talk about them directly and thoroughly. First is the question of Genaro's fit to the canonical three stage pipeline: content determination, sentence planning[6], and surface realization.

[6] We would prefer to say 'text planning'. This choice of phrasing emphases the role of this stage as the point of the transition between the semantic stuff used in the underlying program (the 'speaker' that the generation process is working for) and the first level of linguistic stuff. Also note that in Genaro the input to this level is not already partitioned into sentences. This is one of this processing level's primary tasks.

Genaro divides into three processes that could be given those names. Furthermore the division is a natural one given the facts and not simply an arbitrary choice. Each process is the exclusive user of a particular level of representation: the vision system's description of the picture, Text Structure, surface structure. Each makes exclusive use of a particular body of resources or 'reference knowledge': the salience annotations on the units of the description, the realization classes for each of the units' type (in effect an abstract semantic grammar), the elementary trees of the surface grammar. They do largely share the same class of control structure in that they are all data-directed, but in the MMP model this is taken to be the most efficient means of control and there is no reason not to use it given this suite of tasks and the presumption of monotonic, on-line, indelible processing.

**Lexicalization**. Genaro's lexicalization is trivial. The types (domain-level predicates) of the underlying system map directly to content words or fixed phrases. There is no choosing between alternative words. The transition from semantic type to lexeme is done as an integral part of choices made when the Paragraph Planner selects among the alternative available for units of that type. This is because the alternatives take the form of phrase-types in a lexicalized semantic grammar. The lexemes are implicit in the parameter mappings that will instantiate the phrase-types when the Text Structure is readout to construct lspecs. They do not participate in the choice criteria since within a realization class it is based on the semantic categories of the units and the selectional restrictions already in place within the Text Structure, not on the identity of the lexemes. The lexemes are later projected to morphologically appropriate, concrete words inside Mumble as the surface structure is read out.

**Aggregation**. In its general interpretation as the process of combining small units into larger ones, Genaro's Paragraph Planner practically does nothing other than aggregation. It is also the site of the narrower interpretation of aggregation as clause (or other category) combining. Note however that the medium that represents the 'clause' combination is not comprised of syntactic objects; rather the combination is represented via the annotation on the units within the text structure. The job of observing that some elements are shared between units and that reducing them in thereby possible is done by the Paragraph Planner looking at sequential pairs (opportunistically extended to triples, etc.) at the head of the staging level's buffer that holds the units coming out of IP.

The choice between the possible surface realizations of the reduction, e.g. "(both) the door and the house and the gate of the fence are (both) red", "the door of the house is red and so is the gate of the fence", will be done as part of the reading out of the Text Structure to form lspecs. The criteria for the choices in the original Genaro amounted to flipping a coin and hoping that reading enough of the results would provide some insight into what the real criteria might be. We remain at that state of knowledge today, though it is possible that work underway in inverting a corpus of texts (corporate quarterly earnings reports) via Sparser to reverse engineer the read-out and Text Structure formation decisions may at least provide a register-specific textual criteria, namely match the patterns that have actually been seen.

**Rhetorical structuring**. The rhetorical relationships between the elements of one of Genaro's texts are not particularly deep. As discussed earlier, the only substantive

rhetorical moves are elaborate and shift-topic. These govern the disposition of the individual units of content as they are read out of the initial stream, mapped to presentation types, and entered into the Text Structure, and, quite notably, depend for their accurate results in being treated as a sequential, salience-driven stream of text-constructing decisions. Absolutely nothing would be gained and much might be lost by trying to capture this rhetorical structure, such as it is, in its own level of representation where all of the information had to be present at one time.

**Referring Expressions**. Genaro does not plan the content of its referring expressions so much as stumble into them via the exigencies of what the salience of the properties and relationships that the objects it describes happen to be in and the parameter settings in force at the time. Whether a given property ends up as a modifier or qualifier in a referring expression or a statement in its own right depends on factors that are not planned.

As it happens, in Genaro's domain of house scenes there is no need to distinguish two objects of the same type since they simply will not appear in the model of the picture. Consequently explicit reasoning about how to incorporate pertinent distinguishing attributes (see, e.g., Bateman 1999) is not needed.

Such 'reasoning' as does occur involves the use of determiners and pronouns. It is simple but effective in this domain. Initial references to particular objects (which will always be couched in terms of the objects' types; nothing in this domain has a proper name) are given indefinite determiners.[7] Subsequent references will get definite determiners unless the object in question is the current-item, in which case it is always pronominalized; no object that not the current-item at the time is ever pronominalized.[8]

**Ordering**. In Genaro, the order in which a particular fact was introduced was strictly a function of its visual salience. This means in effect that the ordering is determined at an extremely high level, namely as part of the judgements of the scene recognition system that analyzed the picture and constructed the semantic model that Genaro (as simulated) works from. Locally within the Paragraph Planner the ordering decisions that take place are not a matter of deciding linear precedence so much as functional role. Realizing a property as a qualifier rather than a modifier, for example, reflects a higher salience for that property since it has the effect of conveying a narrower, more specific classification, e.g. Genaro would always say "white New England house" rather than "New England-style white house" since the hourse style, if present in the model, will usually receive a higher salience than the color unless, of course, the color happens to be out of the ordinary such as the purple Victorian down the street from us in our neighborhood.[9]

**Segmentation**. Paragraph level segmentation is not an issue in Genaro since by design it was only capable of producing single paragraphs. Its job was to aggregate the individual

---

[7] The choice of "a" vs. "an" is phonetic and done quite late in the process at Mumble's word-stream level.

[8] This has led to some interesting situations when we looked at the output of scenes with two objects of comparably high salience (the house and the fence in the example given earlier), since some proposed item sequences would introduce the gate as part of a descriptive relationship to the house and later references to the gate while the house was still the current item could feel strained when they were not pronominalized.

[9] Though note that even here a color is not capable of sustaining a referring expression on its own since has no nominal form, and that "Victorian" is available as a style with a nominal realization where "New England" (qua house type) is not.

units passed to it in order of their salience into sentence of suitable length given the character of the information they carried (a weight of about 3.5, see above), and also to clearly indicate the segmentation of the paragraph into a succession of descriptions of the picture's salient objects through the use of syntactic constructions that carried the appropriate rhetorical effect. There-constructions and fronted locative phrases, for example, always appeared at the beginning of the description of the next current item and never within an item's description no matter how many sentences it might have.

**Centering / salience / theme**. The task set for Genaro did not require it to maintain any sort of elaborate machinery for reflecting the information structure of its source model in the textual structure of its paragraphs. The single notion that the object in focus and only that item, i.e. the current item within the Paragraph Planner, should be pronominalized was sufficient.

## 4. What are Genaro's levels and representations as data structures

The substantive question for this workshop is whether Genaro's representational levels and the units that comprise them have a plausible fit to those presently defined in the RAGS project. In particular, what is the nature of the I/O these levels and could it reasonably be emitted as expressions that conformed to what RAGS is proposing.

### 4.1 Concrete Conceptual Structure

Taking the RAGS levels up in the order they appear in the annual report, we begin with the so-called 'concrete conceptual structure'. This is the mechanism proposed for providing uniform access to the information within the speaker's 'knowledge base'. Note that we would prefer to talk about the speaker's model of their situation as it is relevant to their wanting to say something, since their knowledge base will be vast and virtually all irrelevant to their present goals. Be that as it may, we can take 'knowledge base' as code for the set of assumptions being made about the objects that comprise the model, in particular that they can be construed as (preferably) a classification hierarchy of objects with roles in, e.g., the KL-One tradition.

The task for this representation is the export of the relevant units from the speaker's model in a form that can be expressed as stuff that you can say in XML – a melieu of characters rather than pointers and structured objects within a live computational process. As you may properly infer from this choice of phrasing, all of Genaro, and its (simulated) speaker lives in a single live computation and works by passing pointers to typed structured objects.[10] Could we emit them as cdata item ids and role attributes that could then be used as value accessors? Of course. It's only a matter of writing the code (and finding the time).

Do we think that a Penman-style inquiry semantics is the correct way to approach the determination of content? (Given as the justification for RAGS choice in section 4.5 of the report.) Absolutely not. Any realistic speaker that has a reason to communicate

[10] If we happened to need to implement the various components Genaro over a number of different virtual machines, say in Java, we would still pass pointers, albeit via RMI or CORBA-based serializations of the objects as they passed from one machine to another. These serializations are not the kind of thing one would care to express in XML however.

knows what it wants to say and knows its conceptual (if not its semantic) structure better than any of the downstream components of the generator. It should tell those components what content to express just as Genaro does through its salience order stream of units. Our general arguments are presented in some length in McDonald, Meteer, and Pusteyovsky 1987.

If we did publish the contents of the model and its schematic structure (i.e. its set of types, set of roles, and the layout of its classification hierarchy) we would have to fudge the representation of salience somewhat if all we are given to work with is roles and ids. Certainly we could make salience another role available on every object (n.b. that would require multiple inheritance), but as Conklin saw it in the UMass VISIONS system, the output of the scene recognition process gave him an already assembled sequence; all he then had to do to get the unit stream was apply cutoff points to it. Of course an outside process could reconstruct this sequence for itself by searching the entire (sic.) knowledge base of entities and sorting it on the salience role. But the correct thing to do would be to export the sequence per se, cutoffs in place, i.e. to do the content determination within the process space of the model and take advantage of all its pre-existing structure.

### 4.2 Rhetorical and Document Structure

As we said earlier, Genaro has no place for a representational level that captures strictly rhetorical information, be it abstract or concrete.[11] Could we emit one (a concrete one) if forced? No. Genaro has processes that use a knowledge of rhetoric to achieve their goals of reflecting the salience of the elements of the picture in the structure of the text, notably knowledge of how to set off the description of one object from another within the body of a paragraph. But this is knowledge that is applied at the time that it is needed.

One of the dictums of MMP is that the only circumstances in which there is a requirement that the result of a decision to be given an explicit representation is when the consequences of the decision cannot be all be realized at the time the decision is made. Genaro's rhetorical decisions, all within the Paragraph Planner, are immediately reflected in a particular extension of the Text Structure and need no other representation. Since we do not maintain a rhetorical level inside Genaro we cannot emit one.

The same reasoning applies to any document level of representation. Textual boundaries are determined on the fly. There happen to be no intra-sentential boundaries, though we can easily imagine the need for them in registers that employ markedly longer sentences with a genre-specific structure, especially if they were to be uttered by a speech system rather than printed out as text; earnings reports are a prime example. The decisions to segment the content into sentences are reflected immediately by the Paragraph Planner sending the ongoing segment of Text Structure off to be read out and uttered by Mumble. The order of the text over all directly reflects the relative visual salience of the objects in the picture and thus is effectively already established with the speaker's model before actual generation ever begins.

---

[11] And, frankly, in the case of rhetorical structure, the abstract / concrete distinction in rhetorical structure appears forced and difficult to justify outside of a framework that assumes that all levels of stuff will begin in an abstract form and later appear in a concrete form requiring, perforce, that it apply here as well.

We quite agree that extended texts incorporate a rich body of document-level structural information. To what extend this level must be planned in advance of its realization (and thus require an explicit representation) is a question on which we are agnostic for lack of experience.

### 4.3 Abstract Semantic Representation

Abstract semantic representation is characterized in the report as "the first stage at which conceptual information is packaged together in ways that foreshadow an eventual linguistic reali[z]ation" (pg. 31.) From this perspective, all of the information that populates the 'conceptual' model that Genaro draws from is semantic. If we imagine the integration of Genaro with the actual UMass VISIONS system (which would have been possible about ten years after Conklin's work), the stuff prior to the semantic model would have been things like light gradients, Gaussian densities, and schematic two and a half dimensional object models – not the sort of stuff one represents with a specialization-based type hierarchy and accesses with KBIds. VISIONS' conclusions as to what it had seen in a picture were semantic objects from the moment they came into existence. They could not be anything else since the whole notion of understanding a picture as a depicting objects in the world is to recognize within it things that we are able to characterize linguistically.[12]

The question of whether Genaro's semantic model is abstract or concrete hinges on whether it is language independent, something that we never attempted. Alternatively there is the matter of whether there is any "translation" between the schema that comes out of the conceptual model (types and roles) and the what is actually used by later components in the generator; the answer to which is no. On that basis we conclude that we dealing with 'concrete' semantic objects.

Though events or even really states are not a part of Genaro's domain, we have always believed in Davidson's notion that events and their kin can be quantified over. To be specific, we most closely follow the treatments of Emmond Bach and refer to all such stuff as 'eventualities'.

Looking at the example of abstract semantic information given in the XML of section 7.4.2, we do have to wonder a bit about how far away from economically mainstream European languages one can go with that an still get an isomorphic translation to concrete equivalents for it. Would it preserve its factorization when rendered into the proximal source for an agglutinative language? As it stands, the presence of predicates like 'top of' or 'with' make it feel much more like a gloss of the original English text in the style of Bill Martin's mid-1970s work in using natural language directly (if deterministically) as the representation language.

The idea being conveyed in 7.4, that we are emitting the cdata equivalent of pointers into the knowledge base is reasonable enough if the speaker (the owner of the knowledge base) is not in a position to assemble the concrete semantic objects itself – and Genaro can do that.

---

[12] Yes, pictures affect people in myriad other ways that one cannot put into words, but these responses are not (yet) accessible to computer vision systems either.

### 4.4 Concrete Semantic Representation

Turning to the discussion of concrete semantic representations, it would appear that much is made of whether the representational stuff that crosses over to the next level (whatever that happens to be) is a set of minimal propositions versus an object with internal structure that embeds references to other objects. The first taken as typical of an abstract semantic representation, and the second as typical of a concrete one, i.e. strongly oriented towards linguistic realization, while the abstract case is only loosely so via its apparent lexicalization. In the case of Genaro, and particularly of our other work currently underway, taking this distinction as written would be a red herring.

On the one hand, the units in the stream out of the content determination module (IP) are deliberately minimal and could be taken, in isolation of their computational context, as being propositions that can standalone. However, what is really going on, from a theoretical and soon to be actual standpoint, is that the stream is the result of reading out a highly structured and thoroughly interconnected model; in effect serializing it according to some rhetorical criteria, in this case visual salience.

Looking at the stream from the other direction, the receiving module that is responsible for (beginning to) actually give it some explicitly linguistic content and structure, our Paragraph Planner that assembles the Text Structure, this module wants the semantic units it receives to be minimal. That way it can get the maximal flexibility to array the units as compositions of annotated expressive types. If it received a single structure that was a ‘recursive’ representation of, say, a three clause, ten element sentence (typical in the financial domain), it would have to first decompose it back into its minimal elements and sub-relations before it would have a chance of doing an adequate job of fitting the it into the current discourse and informational context.

This to say that we could not, realistically, emit a concrete semantic representation that resembled ESPL. It would be too alien to the sort of linguistic reasoning that we do. But at the same time we should emphasize that our ‘minimal’ units are typed structured objects with typed slots that make reference to other objects. Written out as an expression they would look more like the ‘recursive’ composite on page 37 of the report, and not at all like the propositions just above it.

Having said that, a crucial difference should be pointed out regarding semantic roles. Simplifying somewhat, it would be correct to say that in Genaro their term set is strictly domain-specific (except insofar as its upper structure ends in expressive categories that are virtually all linguistically motivated): ‘biter’ and ‘bit’ rather than ‘agent’ and ‘affected’.

We can do this and still have a very much linguistically-oriented concrete semantic structure because every category (relation type) in one of our domain models is associated directly with the linguistic schema(s) that can realize it and (eventually) a history of the ways it has actually appeared in (various) corpora. The details would take us much too far afield, but it has the effect of letting us have our cake and eat it too.

With this caveat, we could readily emit the expressions that fit the DTD for RAGS concrete syntax. At the same time we would obviously have to include a great deal of reference information (probably our domain-model schema with their surface-grammar

mappings) so that a project that required classic, Filmore semantic relations could attempt to derive them.

### 4.5 Abstract Syntactic Representation

This takes us to abstract syntactic representations – the last stop of the RAGS journey since RAGS wisely does not care to make any pronouncements in the realm of strict syntax with it plethora of theories and disputes. Here I must confess to have taken a garden path in reading the original chapter (9), since I take an 'abstract' representation to mean one that would fully cover the meaning being conveyed, but state the meaning in term of (in this case linguistic) categories (concepts) that underspecify the ultimate set of choices. Instead, the only sense I could make out of the examples given there of abstract features structures was that they were comprised of concrete syntactic stuff that just happened to be incomplete. I can not locate these examples in the revised report, so perhaps a different conception is being taken now.

The compositional grammar that drives the assembly of Text Structure[13] works in terms of categories (expressive types) that are for us a paradigm example of an abstract syntactic structure. The essential features of syntactic structure are retained (head, argument, adjunct) but the details (case assignment, ordering, grammatical categories per se) are omitted. It is essentially underspecified with respect to the eventual syntactic structure in that a great many additional choices remain to be made. In particular the list of 'items omitted' from the bottom of page 46 all remain to be decided (or selected among or simply instantiated as the case may be).

Could we emit a Text Structure following the RAGS DTD?[14] The answer is a qualified no unless we hijacked the interpretation of the terms in the DTD to our own purposes.

For one, the leaves of Text Structure as we intend to implement it are not words, per se, but pointers to a particular syntactic schema (roughly a tree family) from among the set of such schema that are part of the representation of the a unit's category in the domain model. A mapping structure that accompanies the schema will populate and particularize the schema to the unit when it is read out of Text Structure and into a dependency tree of lexicalized elementary trees, and it is within these lexicalized trees that the lexemes live. All of this is reference knowledge referenced by pointers; it would take considerable effort to render it exportable.

If we figured out a way to get around that, we could certainly emit the structure of the Text Structure dependency tree by taking the head, adjunct, etc. annotations on the links, but since we support more specific kinds of elements than the RAGS DTD, we would need to publish an extension to it if the export was not to lose any information.

---

[13] So far this grammar has been implicit in the code that does the assembly. We expect to make it explicit over the course of the next year or so.

[14] Note that it would only make sense to do this sentence by sentence as each successive planning unit is completed, since down stream processes should have the same sort of incremental access to context that we do. To emit the Text Structure of a whole paragraph as one expression would tend to imply that it was a representation that could be processed starting at any point one choose, which would be contrary to our intent.

More fundamental is the question 'would anyone want to get an export of an instance of Text Structure' – is there a syntactic realization component with a close enough fit to its assumptions about what information is fixed at in its representation and what left un(der)-determined that it could make use of it? Certainly it would be useless to a system like Penman since the two make completely different assumptions here. An obvious candidate would be another surface realizer that uses a TAG grammar like we do (Mumble-86). Such a realizer would already have a body of elementary trees at its disposal, and we can see how a set of ids could be worked out to establish the correspondences between its trees and our own. Exposing the mapping structures that populate tree schema to arrive at the lexicalized trees actually used in the realizer would take more work but should be doable.

Note that we could certainly and relatively trivially export the lspecs that are read out of the Text Structure to a TAG-based realizer since they are just a variation on the dependency trees it would use itself. This could lead to some rather interesting experiments depending on what kinds of knowledge is brought to bear within the realizer. If it simply linearizes the tree then all we could look at would be matters like efficiency. If it does more so much the better. In our case we know that the things that we will want to handle after the Text Structure has been established are the impact of the information structure (e.g. the use of topic-shift indicating constructions; heuristics for doing this were developed with Genaro), and adherence to the fine-grained textual structure of the target domain (mostly its choice of how arrange the internal structure of phrases and choices of connectives: "first quarter earnings" vs. "earnings for the first quarter"; this work is just starting).

**References**


Barwise, Jon & John Perry (1983) *Situations and Attitudes*, MIT Press, Cambridge, Massachusetts.

Conklin, E. Jeffery (1983) **Data-Driven Indelible Planning of Discourse Generation Using Salience**, Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts at Amherst, May, 1983; available as COINS Technical Report 83-13.

_____, Ehrlich, Kate & McDonald, David (1983) "An Empirical Investigation of Visual Salience and its Role in Text Generation" in *Cognition and Brain Theory*, 6(2) Spring 1983.

_____ & McDonald, David (1982) "Salience: the key to the selection problem in natural language generation", Proceedings of the ACL, University of Toronto, June 16-18, 1982, pp. 129-135.

Devlin Keith (1991) **Logic and information**, Cambridge University Press

Kempen, Gerard & Eduard Hoenkamp (1987) "An incremental procedural grammar for sentence formulation, *Cogntive Science*, 11, pp. 201-258.

Marcus, Mitch (1980) A Theory of Syntactic Recognition for Natural Language, MIT Press.

McDonald, David (1980) **Natural Language Production as a Process of Decision-making under Constraints**, Ph.D. Dissertation, MIT Artificial Intelligence Laboratory, August 1980.

(1984) “Description Directed Control:, *Computers and Mathematics* 9(1); Reprinted in Grosz et. al (eds.) **Readings in Natural Language Processing**, Morgan Kaufman Publishers, Los Altos California, 1986, pp. 519-538.

_____ (1994) "Reversible NLP by Linking the Grammar to the Knowledge Base", in Strzalkowski (ed), **Reversible Grammar in Natural Language Processing**, Kluwer Academic, pp. 257-291, 1994.

_____ (in press) “Issues in the comprehension of real texts: the deign of Krisp”, to appear in volume edited by Wojca and Shapiro, MIT Press.

_____, Meteer, Marie & Pustejovsky, James (1987) “Factors Contributing to Efficiency in Natural Language Generation”, in G. Kempen (ed.), **Natural Language Generation**, Martinus Nijoff Publishers, Dordrecht, pp. 159-182.

Meteer, Marie (1982) **Expressibility and the Problem of Efficient Text Planning**, Pinter, London.

_____, David McDonald, Scott Anderson, David Forster, Linda Gay, Allison Huettner (1987) “Mumble-89: Design and Implementation, UMass Technical Report 87-87, 173 pgs.

Parma, Cesare C., Hanson, Allan R., and Riseman, Edward M. (1980) “Experiments in Schema-Driven Interpretation of a Natural Scene”, in J.C. Simon & R.M. Haralick (Eds.) *Digital Image Processing*, Reidel Publishing Co. Dordrecht, Holland, pp. 303-334.