GLINDA:
Natural Language Text Generation in the Oz Interactive Fiction Project

Mark Kantrowitz

July 1990

CMU-CS-90-158

School of Computer Science
Cargnegie Mellon University
Pittsburgh, PA 15213-3890
mkant+@cs.cmu.edu

Abstract

Interactive fiction presents new requirements for natural language generation. In particular, changing the style of output text should be as easy as turning a knob or flipping a switch. GLINDA, the natural language generation module of the Oz interactive fiction system, is an implemented natural language text generation system which uses the propagation of features during generation to license generation rules. These rules control the selection, organization, transformation and combination of elements from the input semantic and pragmatic representation within a uniform, multi-level linguistic framework. Several examples of English tense and aspect structure illustrate the benefits of this approach. GLINDA is fast and elegant, and generates text in an incremental, indelible, left-to-right fashion.

This research was sponsored in part by a Hertz Foundation research fellowship grant. The author is supported by a National Science Foundation (NSF) graduate fellowship.

Keywords: Natural Language Generation, Natural Language Processing, Computational Linguistics, Interactive Fiction.

GLINDA: Natural Language Generation in the Oz Project

Mark Kantrowitz
mkant+@cs.cmu.edu

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

1. Introduction

GLINDA is the natural language generation module of the Oz interactive fiction system developed at the CMU School of Computer Science [Bates 1989]. Oz is intended to simulate dramatically interesting microworlds, as a form of interactive fiction. The major goals of language generation in Oz are to produce a variety of views of the reality that Oz simulates and to produce vivid dialogue for computer modelled agents. To accomplish this the system must be able to "tune" the natural language generation to engender subtle emotional reactions in and influences on the human player [Smith & Bates 1989; Laurel 1986]. At the level of realization this means providing an interface of "knobs" for controlling the style of natural language output. GLINDA provides such an interface by means of a set of user-specified generation rules which define the knobs and their effects.

Imagine sitting in front of a huge control panel covered with dozens of knobs and switches. Press the voice button, and the text becomes passive. Close the slang switch, and vernacular issues forth. Turn the suspense knob up a notch, and a bank of switches on the right activates; some of these switches then automatically flip themselves on and off as the generation progresses, depending on the context. Under the direction of a plot manager, such a control panel would allow Oz to adjust its style of output to match the changing dramatic content of the story.

Figure 1-1 shows how GLINDA fits in with the rest of the Oz system architecture. The human user types commands at BUP, a bottom up parser, which translates them into acts for the Player character. The Player's acts affect the physical world model, creating sense data (mainly sight and sound data) which propagate through the structure of the world. These sense data, along with the sense data created by other characters' acts, return to the Player and are translated into an inter-lingua text (ILT) format by the Gen-ILT module. The Gen-ILT module also informs the BUP parser about new objects sensed, so that it may recognize them when the user refers to them. The ILTs, along with a set of knobs specified by the plot manager, are passed to GLINDA, which generates the text output, complete with punctuation.

To operationalize the knob paradigm, GLINDA represents knobs using features. A feature is an attribute or property-value pair, where the property name identifies the knob, and the property value the setting. The presence of a particular feature during generation can license rules at each level of generation, controlling focus, syntactic structure, lexical selection, and morphology. The rules, in turn, can modify the old features, block or allow their propagation, and introduce new

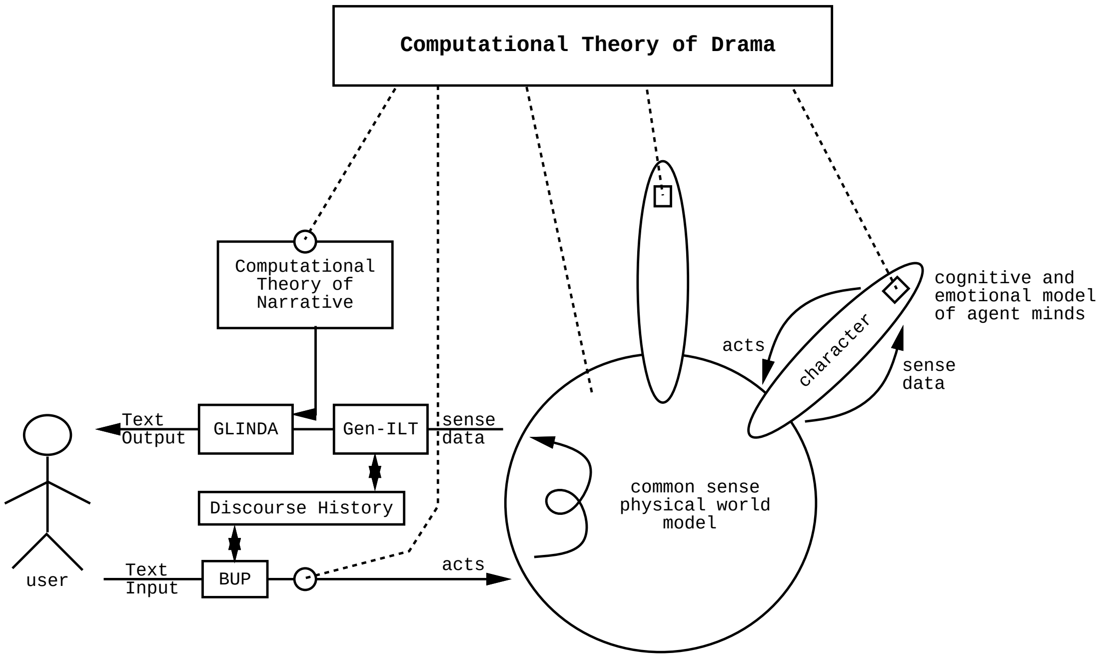

Figure 1-1: The Oz System Architecture

features.

A generator maps from a semantic and pragmatic representation of a message to the surface utterances that express its meaning and style. Generation tends to be semantically local, in that realization of a predicate-argument relation or of an object is largely independent of their attachment within a phrase or other constituent. GLINDA attempts to separate the purely semantic and pragmatic from the purely syntactic by using features to communicate between the two. The features act as the sole intermediary connecting the conceptual content with syntactic expression.

Because features isolate any interaction between syntax and semantics to a single interface, they concentrate control of all realization processes into one mechanism. This has yielded benefits in uniformity and simplicity. The system seems to be capable of treating each level of linguistic structure (e.g., sentence organization, phrase structure, lexical selection, morphology, and phonology) in an identical manner, instead of requiring a series of modules. This has reduced the size of the generator considerably, and increased its efficiency.

GLINDA's design was inspired by Ed Hovy's PAULINE text planning and generation system [Hovy 1988] and the CMU Center for Machine Translation's KBMT system [Tomita 1988]. The operation of the basic text-planning primitives of PAULINE seems similar to the mapping and pseudo-unification primitives of the KBMT system. We felt that we might be able to generalize these primitives and benefit from the uniformity. GLINDA is the result of our efforts along these lines.

GLINDA is conceptually rather simple. There are three primitive operations for generating output, which we refer to as *selection*, *organization*, and *combination*. Selection filters the input and chooses what to include and what to exclude. Organization orders what has been selected and fixes the general structure of the output. Combination takes the results of organization after they have been recursively generated, and merges them into the final output string. Rules for each of the three operations determine how an item will be realized.

Feature constraints on the generation rules for selection, organization, and combination provide a means for choosing which rules to apply. The rules, in turn, control how features propagate during generation, and which features will be imposed on the generation of a particular linguistic element. There are also specific rules governing the transformation or *mapping* of features at each recursive call of the generator.

GLINDA is similar in many ways to MUMBLE [McDonald 1980, 1983, 1985]. Both systems are description-directed, in that the input and not the grammar drives the generation process. Both produce output in an indelible manner; once a rule has been invoked, it cannot be retracted. On the other hand, MUMBLE attaches plan units as the generation progresses, whereas the semantic content of GLINDA's input is fully specified from the start.[1] More importantly, GLINDA is a cleanly and tightly integrated architecture; grammatical information is clearly encoded in rules, as opposed to being hidden in MUMBLE's arbitrary LISP procedures, making it easier to examine and modify the grammar.

In the following sections we describe the semantic and pragmatic representation, the rule formats, and the operation of the algorithm. Several examples of English tense and aspect structure demonstrate how GLINDA works. We conclude with a discussion of specific contributions of this system and areas of future work.

2. The Semantic and Pragmatic Representation

The semantic and pragmatic representation we use emphasizes the functional properties of the input in a manner similar to LFG [Kaplan & Bresnan 1982] and PATR-II [Shieber 1983].

We define the semantic and pragmatic representation and present a simple example to illustrate its use.

2.1. The Definition of "Groups"

The basic data structure in GLINDA is called a *group*. A group is a typed set of *items* acting as a single entity.[2] The type of a group indicates its level in the linguistic hierarchy (morphology,

[1]This difference is not of immediate concern, as GLINDA could easily be modified to operate like MUMBLE if there were a reason to do so.

[2]Like f-structures [Bresnan 1982; Nyberg 1988; Tomita 1988], feature bundles [Gazdar 1985], functional structures [Kay 1983], and dags [Shieber 1983], the arguments of a relation are included in the group. Thus a group is the set consisting of a predicate and *all* its arguments. Each item of a group is tagged with its functional role.

lexical, object, parameter, relation, multi-relation relation, etc.). The items of a group may be features or other groups. Higher level linguistic structures are created by combining lower level groups into a group.

While the user may specify a feature as a (property value) pair, internally features are represented as single-item groups whose type is the property-name and whose item is the value. The values are typically symbols or strings identifying primitive objects in the semantic domain. Features are also known as primitive or "root" groups.[3]

Groups differ from case-frame representations of language in that the functional role of a group is included as an item of the group, and is realizable by the generator, instead of just being a slot name. The role of a group is the item whose type is `role`. If no such item exists in the group, as in primitive groups, the type of the group is used instead. For this reason we occasionally refer to features as role-value pairs. Since roles are often realized as prepositions, this conflates the NP and PP categories into one group. (Similarly, features not only control which generation rules are fired, but can themselves be realized.)

Thus the group's type indicates its structural level, while the role indicates its functional relationship with its siblings. Together the items of a group encode the meaning of the group.

2.2. Abbreviating Groups as Features

It is useful in discussing groups to abbreviate them as the feature which most closely summarizes their meaning. When we use the pair (property value) to refer to a group, the property is the value of the group's role, and the value is the group's projector.

For example, the group

```
(PARAMETER (ROLE AGENT)
           (ARG (WORD (ROOT JOHN-1))
                (PERSON 3) (NUMBER SINGULAR) (GENDER MALE)))
```

may be abbreviated as the feature `(AGENT JOHN-1)`.

---

[3]GLINDA's features are parameters with a name and value. There is no a priori limit on the number of possible values for a given parameter. If a parameter is absent, the value defaults to nil. This is in contrast to three other common methods of representing features:

- Binary features with two values, + and -: This scheme has three values, `+plural`, `-plural`, and "not-present".
- Flags or markers: This scheme has two values; either the `plural` flag is present or it isn't (the latter being singular).
- Operators: This scheme uses `plural` as an operator which when applied to a singular noun creates a plural one.

We feel that our use is more flexible than the alternatives (it generalizes the first two) without becoming too complex. The advantage over the third scheme is that it allows us to treat features as messages passed around during generation.

The *projector*[4] of a group is the value of the feature eventually contained[5] within the group which is central to the meaning of the group.

More formally, each group has a distinguished item called a *head* or *local projector*. The head is selected based on its role. The head-role is fixed for each type of group.

For example, the following definitions assign the head-roles for several of the types of groups from GLINDA.

```
(define-group-head sentence matrix)
(define-group-head relation predicate)
(define-group-head parameter arg)
(define-group-head arg word)
(define-group-head word root)
```

Descending along head links from the group we eventually reach a primitive group (feature). The value of this feature is the *projector* of the group and the group is the *projection* of the feature.

The non-head items of a group are terminal in the sense that their projectors are prevented from being inherited by higher groups. Such groups are called the *maximal projection* of their respective projectors. Formally, XP is the maximal projection of X if XP is a projection of X and XP is not the head of the group containing XP.[6]

Because rules in GLINDA are attached to projectors, the maximal projector of a group can control the realization of its siblings[7] and their children by specifying how features are passed to its siblings. For example, in phrase-structure grammar the verb governs its complements; in GLINDA, the projector of the predicate group governs the other items in its group's parent, which includes the agent.

---

[4]It is hard to use terminology without evincing a panoply of relevant and irrelevant associations. In the definitions that follow, the intent is to define functions as they are actually used in the generator. While the definitions are similar to traditional notions from syntactic theory, they often have subtle twists. Accordingly, for any loaded term, use the nontechnical sense. Where possible, we will define the meaning of such terms in a footnote.

[5]A group X *immediately dominates* (contains) a group Y if and only if $Y \in X$ (Y is one of X's items). If X immediately dominates Y, we say that Y *is in* X, X is Y's *parent*, and Y is one of X's *children*. For example, the items of a group are immediately dominated by the group.

X *dominates* (eventually contains) Y if and only if X immediately dominates Y or X immediately dominates a group that dominates Y. Thus X does not dominate itself.

[6]The maximal projections correspond to the phrase-level categories in traditional CFG vocabularies, and primitive groups to zero-level categories. A consequence of this definition is that only maximal projections may occur in the specifier and complement positions of a parse-tree.

[7]X and Y are *siblings* if and only if X and Y are immediately dominated by the same group and $X \neq Y$.

This notion of control over realization is called government.[8] We sometimes call the head of a group the *relevant* governor since it is the only item which can case-mark its siblings and their governable descendents. Formally, the relevant governor of Y is the head X of the nearest maximal projection XP that dominates Y, if X governs Y.

2.3. A Simple Example

The notion of a group was selected for semantic reasons, because sets distill the essence of a predicate-argument relationship. This makes it easy to translate the Oz semantic representation into this format. The top-level group is in effect a hierarchy of nested sets, with each element of a set an identifiable object or relation.

Figure 2-1 shows a sample group representing a relation with agent `JOHN-1`, object `MARY-1`, and predicate `LOVE`.

```
(SENTENCE (ROLE SENTENCE)
 (RELATION (ROLE MATRIX)
  (PARAMETER (ROLE PREDICATE) (ARG (WORD (ROOT LOVE))))
  (PARAMETER (ROLE AGENT)
   (ARG (WORD (ROOT JOHN-1))
    (PERSON 3) (NUMBER SINGULAR)
    (GENDER MALE) (FORM LEXICAL)))
  (PARAMETER (ROLE OBJECT)
   (ARG (WORD (ROOT MARY-1))
    (FORM LEXICAL) (GENDER FEMALE)
    (PERSON 3) (NUMBER SINGULAR))))
 (MOOD DECLARATIVE))
```

Figure 2-1: A Simple Group

The sentence-group contains a single relation-group, which represents the matrix clause. The relation-group contains three parameters, which we abbreviate as features: `(PREDICATE LOVE)`, `(AGENT JOHN-1)` and `(OBJECT MARY-1)`. The first of each pair is the role of the

[8] X *g-commands* Y with respect to an ordering if and only if X is in a group that dominates Y, X does not dominate Y, and X is generated before Y in the ordering. In other words, X is either a sibling of Y or a sibling of a group that dominates Y, and X is generated before Y. For example, `(AGENT JOHN-1)` g-commands its siblings `(PREDICATE LOVE)` and `(OBJECT MARY-1)` in Figure 2-1.

X *governs* Y if and only if

1. X is a primitive group,
2. X g-commands Y, and
3. Y is not dominated by any maximal projection which does not also dominate X.

Strictly speaking, the organization rules are attached to projectors of groups and not the groups themselves. This generalizes the first condition of the definition of the government relation above.

group and the second is the value.

Initially the items of a group are unordered. During generation (here, much simplified), features are used to impose an ordering on the items of a group. For example, when the feature `(VOICE ACTIVE)` is present we might use the order `AGENT PREDICATE OBJECT`. Once the items are ordered, we recursively generate each item in a left-to-right fashion. As each item is generated, it may return some features to be used during the generation of the next item. For example, after `JOHN-1` is realized as "John", it returns the `PERSON` and `NUMBER` features, which license the use of the suffix "s" in the realization of the predicate as "loves". Thus the feature passing mechanism may be used to enforce subject-verb agreement.

Figure 2-2 demonstrates the generation of a group with a variety of features. Here the variable `test` is set to the group from Figure 2-1.

Note that in the passive voice the `(ROLE AGENT)` feature is realized as the preposition "by".

```
<cl> (progn
       (generate test '((voice active) (time past)))
       (generate test '((voice active) (time present)))
       (generate test '((voice active) (time future)))
       (generate test '((voice passive) (time present)
                        (truth negative)(contraction aux)))
       (generate test '((voice active) (time present)
                        (truth negative)))
       (generate test '((voice passive) (time past)
                        (perfect t)(progressive begin)
                        (modal ability)
                        (truth negative) (contraction aux)))
       (generate test '((voice active) (time present)
                        (progressive region)
                        (truth negative)))
       (generate test '((voice active) (time present)
                        (progressive region)
                        (truth negative)(slang t))))
John loved Mary.
John loves Mary.
John will love Mary.
Mary isn't loved by John.
John does not love Mary.
Mary couldn't have begun being loved by John.
John is not loving Mary.
John ain't loving Mary.
```

Figure 2-2: Generation Examples

As we shall see later, rules may be defined in GLINDA in an incremental manner, so that as soon as we've defined the rules for slang, say, we can combine them without modification with the rules for progressives by specifying both features.

3. Organization and Combination Rules

The realization of a group is governed by organization and combination rules. The organization rules include the selection operation in GLINDA. The rules are attached to categories in a projector ontology.

This section describes the projector ontology definition, feature mapping rules, organization rules, and combination rules.

3.1. Ontology Definition

The projectors form the leaves of an ontology, in which rules may be attached either to projectors or to categories of projectors. A given projector will inherit rules from its ancestors if they aren't shadowed by rules closer to the projector in the hierarchy.

For example, one could define count-nouns and mass-nouns in GLINDA by means of the following definitions:

```
(define-ontological-type count-noun noun)
(define-ontological-type mass-noun noun)
```

One would then define organization rules for both count nouns and mass nouns by attaching them to the `noun` category. Where count nouns differ from mass nouns (e.g., in the use of the `NUMBER` feature), specialized rules could be attached to either `count-noun` or `mass-noun`, as appropriate.

3.2. Feature Mapping Rules

When a group is generated, the accompanying features (the features returned by the last group generated, global features, and any "marking" features specified by the organization of the parent group) and the group's items are passed through a filter. Currently the filter (feature-mapping rules) just specifies some additional features. In future versions of GLINDA we expect to extend this to handle history testing for pronoun generation.

The rule which handles do-support in GLINDA is illustrated below:

```
(define-maprule relation
  :constraints ((truth negative)
                (modal (not *))(perfect (not *))
                (progressive (and (not t) (not region)))
                (voice active))
  :return ((modal emphatic)))
```

This rule specifies that when generating a negative relation-group, where the modal, perfect, and progressive features are either nil or not present, and the voice is active, a modal feature of value

`EMPHATIC` is added. The fifth sentence from Figure 2-2 gives an example resulting from the use of this rule.

There is currently no need for feature mapping rules to delete features, as the organization rules may implicitly delete them by not passing them to the group's items. The mapping rules may also be used to "translate" one feature into one or more other features, as we shall see in Section 4 with the time features.

3.3. Organization Rules

Organization rules are attached to categories in the projector ontology, and specify, for a particular type of group, how the items in the group will be ordered, and which features, if any, will be imposed (marked) on individual items as they are recursively generated. The latter may be used to implement case-marking.

The order of the arguments in an organization-rule is specified in terms of their roles. Subcategorization restrictions appear as role restrictions in the rules attached to specific categories of projectors. This suggests that only ontologically meaningful classes of projectors may share the same subcategorization frame. Additionally, this mechanism of rule attachment generalizes subcategorization, since such role restrictions can occur not just for predicates of relations, but also for heads of objects, as in "[the teacher [*of* English]]".

Since the roles of the items of a group are used to identify particular items of the group in organization rules, the following two properties must be satisfied:

- *Role Uniqueness:* We say that a role has a *value* in a group if there is an item of that role in the group. The uniqueness condition is that in a given group, a particular role (property, feature) may have at most one value. This is necessary for us to be able to refer to items in the organization rules via their roles without ambiguity.
- *Organization Rules as Permutations:* An organization rule imposes an order on the items of the group. Since organization rules also perform the selection operation in GLINDA, the order is defined on a subset of the items of the group. In any event, the order is a permutation of a subset of the items of the group, so a particular role appears in the organization rule's ordering at most once.

In order for an organization rule to be active, its constraints must be satisfied. The constraints are matched against features and items in the group and any features passed down from its parent. These constraints are property-value pairs, with the value optional (in which case it is a wildcard value). The value may be specified using combinations of AND, OR, and NOT. Since roles and projectors are treated like properties and values, the constraints may also restrict the items of the group. For example, a role constraint with wildcard value makes the presence of an item with that role required; otherwise the inclusion of the item specified in the ordering is optional. If a projector value is also specified for the role, then the constraint may be used to restrict the projector for that role to an ontological subtype of the value. Together with the role-projector constraints, the specific ordering provides a generalization of subcategorization restrictions.

If more than one rule is active, GLINDA picks the first one it finds. Ideally all rules would be distinguished by disjoint constraint specifications. We are considering using either random rule choice, backtracking, or static ordering if several rules are active due to underspecification.

The organization rules have been used to represent a variety of levels of linguistic phenomena, including syntax, lexical selection, and morphology. Since the selection, organization and combination operations are generalizations of the inclusion, ordering and casting operations of Ed Hovy’s PAULINE text planning system [Hovy 1988], GLINDA should be able to do text planning. Eventually we will extend GLINDA to handle phonology as well. (While each level can be considered as a two-level representation, the group and its ordering, it seems that phonology requires that GLINDA carry the underlying representation along with the surface realization. This is mainly needed for handling exceptional spelling conventions, such as "write" and "receive".)

The attachment of rules to projectors and categories provides for a natural exception handling mechanism. It allows any item to have an escape for rules specific to that item’s projector. For example, GO has an irregular past tense of "went"; the conjugation rules for GO will shadow the generic verb rules:

```
(define-organization-rule GO word
  :constraints ((inflection past))
  :order ("went"))
```

Such rules are actually stored in a *trie* [Knuth 1975], with constraints serving as the nodes.[9] To define a collection of related rules together, one would use the syntax:

```
(define-concept go verb
  root (:constraints () :order ("go"))
  word (:system ((inflection past)
                 (:rule (:order ("went"))))
                ((inflection past-part)
                 (:rule (:order ("gone"))))))
```

A *concept* is simply a gathering together of all the rules attached to a particular projector or projector category. The projector’s parent in the ontology, which is specified after the projector (in this case it is VERB), serves as the root concept from which the new concept varies.

The next rule, part of the system of rules for pronouns, shows how the organization rules may be used for lexical selection. This rule says that the word-group for nouns, for example, is realized as the string "he" if it is in the pronominal form, nominative case, and 3rd person singular masculine.

[9]This internal representation is similar to the choice experts in Penman [Mann 1982] and systemic grammars, except that the choice experts in GLINDA are constraints, not arbitrary LISP code. Moreover, some choices of the systemic grammar are actually encoded in the projector ontology. A projector implicitly has a feature which distinguishes it from its parents; the projector’s siblings differ from each other in the value of this feature. If we were to replace the projector ontology with a table that defined projector categories as collections of features, the rules would become more similar to systems, but also much more complicated.

```
(define-concept NOUN thing
  WORD
  (:system
   ((form pronominal)
    ((case nominative)
     ((person 1)
      ((number singular)
       (:rule (:order ("I") :return (number person))))
      ((number plural)
       (:rule (:order ("we") :return (number person)))))
     ((person 2)
      (:rule (:order ("you") :return (number person))))
     ((person 3)
      ((number singular)
       (:rule (:order ("it") :return (number person)))
       ((gender male)
        (:rule (:order ("he") :return (number person))))
       ((gender female)
        (:rule (:order ("she") :return (number person))))
       ((gender neuter)
        (:rule (:order ("it") :return (number person)))))
      ((number plural)
       (:rule (:order ("they") :return (number person))))))))))
```

These rules also specify that after the noun is realized, it should return the number and person features for use in generating the next item in the order (e.g., the predicate). Also note that the rules are ordered from the last tested to the first, allowing the specification of defaults. In the 3rd person singular rules the string "it" is the default, in case no `GENDER` feature is specified.

The following rule illustrates relation ordering (the actual rule is much longer and appears in Section 4). It is attached to the verb category of projectors, and specifies that groups of type relation will be realized as agent followed by predicate, object and location, with the agent in nominative case and the object and location in objective case. The agent and predicate must be present in the group, and the voice property must be active:

```
(define-organization-rule VERB relation
  :constraints (agent predicate (voice active))
  :order ((agent (case nominative))
          predicate
          (object (case objective))
          (location (case objective))))
```

3.4. Combination Rules

Combination rules, which are attached to the type of the group, specify how strings generated for the items in the group are to be concatenated. These rules operate on two strings at a time, and based on the characters in each string near the junction point, may modify, insert, and delete characters. For example, at the morphological level combination rules may be used to implement standard English morphology; at the word level, inter-word spacing; at the sentence level, sentence capitalization.

Here is a combination rule for gemination in English, which doubles the last consonant in the root when it obeys certain conditions and is followed by a suffix beginning with a vowel (e.g., begin+ing = beginning):

```
(define-char-variable vowel "aeiou")
(define-char-variable gemination-consonant "bdfglmnprst")
(define-combination-rule word
  :left      ((NOT vowel) vowel gemination-consonant)
  :right     (vowel)
  :new-left  (left (last-char left))
  :new-right (right))
```

Combination rules may also be used for morphological changes at the word level:

```
(define-combination-rule arg
  :left "a"
  :right ((OR vowel #\h))
  :new-left ("an")
  :new-right (right))
```

The following simple rule handles inter-word spacing:

```
(define-char-variable no-space-after "‘#")
(define-char-variable no-space-before ",’.!?")
(define-combination-rule arg
  :left  ((NOT no-space-after))
  :right ((NOT no-space-before))
  :new-left  (left " ")
  :new-right (right))
```

4. Extended Examples

In this section we discuss one way GLINDA has handled English tense and aspect structure, including the examples from Figure 2-2 and the temporally coordinated clauses from Figure 4-1 below.

We use Hans Reichenbach’s three-point semantics for English tenses [Reichenbach 1947], with slight modifications. Reichenbachian SRE timelines describe the underlying structure of tenses with respect to three moments or points in time:

- The point of Speech (**S**)
- The point of Reference (**R**)
- The point of Event (**E**)

The interpretation of S and E is self-evident. R is the temporal perspective from which the described event is viewed. The relative ordering of S, R, and E (e.g., before, simultaneous, and after) may be used to define the semantics of a variety of tenses. For example, Reichenbach defines simple past, present, and future as:

```
Past    = S>R=E
Present = S=R=E
Future  = S<R=E
```

Instead, we split SRE triples into two pairs, SR and RE, and let each correspond to a GLINDA feature. For example, our `PERFECT` feature corresponds to RE, and the `TIME` feature corresponds to SR.

```
<cl> test3
(SENTENCE
 (RELATION (ROLE MATRIX)
  (PARAMETER (ROLE PREDICATE) (ARG (WORD (ROOT WIN))))
  (PARAMETER (ROLE AGENT)
   (ARG
    (WORD (ROOT JOHN-1))
    (PERSON 3) (NUMBER SINGULAR)
    (FORM LEXICAL) (GENDER MALE))))
 (RELATION (ROLE ADJUNCT)
  (PARAMETER (ROLE PREDICATE) (ARG (WORD (ROOT ARRIVE))))
  (PARAMETER (ROLE AGENT)
   (ARG (WORD (ROOT MARY-1))
    (FORM LEXICAL) (GENDER FEMALE)
    (PERSON 3) (NUMBER SINGULAR))))
 (MOOD DECLARATIVE) (ROLE SENTENCE))
<cl> (progn
       (generate test3 '((voice active) (time past)))
       (generate test3 '((voice active) (time present)))
       (generate test3 '((voice active) (time future))))
John won when Mary arrived.
John wins when Mary arrives.
John will win when Mary arrives.
```

Figure 4-1: Temporally Coordinated Clauses

However, rather than define `PERFECT` as a feature with three values, E<R, E=R, and E>R, it seems only necessary to define `PERFECT` as a binary feature with values E<R and NOT(E<R). Similarly, we translate the `TIME` feature into an `R-TIME` feature with values S<R and NOT(S<R), represented in GLINDA as `PAST` and `NOT-PAST`. Thus `(TIME PAST)` becomes `(R-TIME PAST)`, `(TIME PRESENT)` becomes `(R-TIME NOT-PAST)`, and `(TIME FUTURE)` becomes `(R-TIME NOT-PAST)`. To distinguish present and future we define an extra feature-mapping rule for future which adds the feature `(MODAL INTENT)` if no modal feature is specified.

The following feature-mapping rules implement this correspondence:

```
(define-maprule relation
  :constraints ((time past))    :return ((r-time past)))
(define-maprule relation
  :constraints ((time present)) :return ((r-time not-past)))
```

```
(define-maprule relation
  :constraints ((time future))  :return ((r-time not-past)))
(define-maprule relation
  :constraints ((time future)(modal (not *)))
  :return ((modal intent)))
```

Limiting the time distinctions to PAST vs. NOT-PAST is equivalent to Brent's modified definition of the simultaneity operator [Brent 1990]. In particular, we are able to generate legal combinations of temporally coordinated clauses simply by having relation-groups return only the R-TIME feature. This suggests that tenses are coordinated by an agreement mechanism between the clauses, parallel to the way in which a verb agrees with its subject in number and person.

The following rules are some of those actually used for generating English tense and aspect in GLINDA. In the current syntax for organization rules, the order is an ordered list of role-specifiers. A role-specifier is either a role or a list beginning with a role, and specifies which item of the group is attached in that position. The rest of the role-specifier list is a specification of which features of the group are passed downward when generating the item specified by the role-specifier.

```
(define-organization-rule VERB relation
   :constraints ((voice active))
   :order ((role r-time)
           (agent (case nominative) mood form)
           r-time
           truth
           (modal contraction)
           (perfect contraction)
           (progressive contraction slang)
           (voice contraction slang)
           predicate
           (object (case objective) mood form)
           (iobject (case objective) mood form))
   :return (r-time))

(define-organization-rule VERB relation
   :constraints ((voice passive))
   :order ((role r-time)
           (object (case nominative) mood)
           r-time
           truth
           (modal contraction)
           (perfect contraction)
           (progressive contraction slang)
           (voice contraction slang)
           predicate
           (iobject (case objective) mood form)
           (agent (case objective) mood form))
   :return (r-time))

(define-organization-rule t perfect
  ;; Perfect = HAVE + PastPart
```

```
    :constraints ()
    :order ((#G(word (root have))
               negative contraction mood
               inflection number person))
    :return ((inflection past-part)))

  (define-organization-rule VERB word
    :constraints ()
    :order (root (suffix person number)))

  (define-feature-concept inflection suffix
    not-past (:constraints ((person 3)(number singular))
              :order ("s"))
    past     (:constraints () :order ("ed"))
    past-part (:constraints () :order ("ed"))
    pres-part (:constraints () :order ("ing")))
```

`R-TIME` is realized as `nil`, but returns an `INFLECTION` feature of value `PAST` or `NOT-PAST`. Its position in the ordering ensures that only the first auxiliary verb is tensed. Suppose the first feature after `R-TIME` was the `PERFECT`, with a value of `t`. It is realized as the word `HAVE` with appropriate inflection, number and person agreement, and returns a new inflection feature with value `PAST-PART`. From the organization rule for realizing verbs at the word level, we see that the `ROOT` is followed by the `SUFFIX`. An `INFLECTION` feature is a subtype of `SUFFIX`, so it is realized as "ed" for value `PAST-PART`, combining with the root. Of course, irregular verbs can shadow these rules, as does the rule for GO in Section 3.3 above.

For a further discussion of tense and aspect systems, see [Brent 1990], [Comrie 1985], [Dahl 1985], [Reichenbach 1947], and the June 1988 Special Issue on Tense and Aspect of *Computational Linguistics*.

5. The Text Generation Evaluator

As in the text-generation evaluator model [McDonald 1983], the generation process in GLINDA is driven by the message to be expressed and not by the hierarchical structure of the grammar. GLINDA evaluates the input to produce the output text in an incremental, indelible, left-to-right fashion. In some sense GLINDA is an interpreter for a programming language for natural language generation.

The generation algorithm is elegant and surprisingly simple. In generating text for a group and a set of features, the algorithm works as follows:

1. Add any features specified by the feature mapping rules, based on the group and features.
2. Choose an organization rule for the group and features.
3. Use the organization rule to select items of the group to be realized and put them in order.
4. Proceed in a left to right order among the items of the organized group, recursively

generating them. When an item is generated, it is passed any features marked for it in the organization rule, and the features returned by the previous item generated (this handles subject-verb agreement).

5. Combine the results of this recursive depth-first generation into a string to be returned.

6. Also return a list of the features specified by the organization rule.

See Figure 5-1 for an pictorial representation of the flow of groups through the evaluator.

The combination step actually occurs only when a string is generated, and outputs the strings when they are no longer subject to combination. This ensures a continual production of output throughout the operation of the generation algorithm. Currently we output at the word level, since doing it at lower levels produces slightly choppy flow of output. We may modify the system so that the level at which strings are output is under generator control.

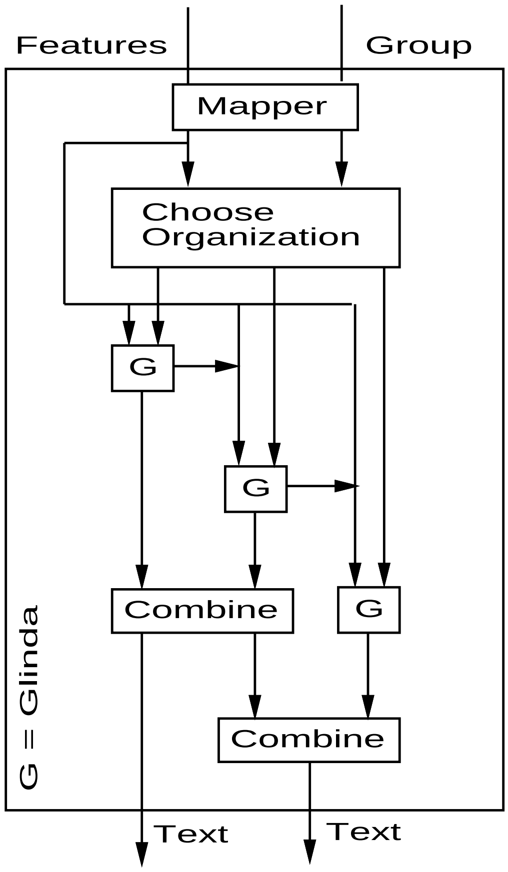

Figure 5-1: The GLINDA Text Generation Evaluator

Global control over generation is provided by specifying a list of features at top level, which are then passed among the components dominated by the group as they are recursively generated. Local control is provided by the feature-mapper which modifies the flow of features before rule selection.

The feature-mapper will eventually be extended to include tests against a history of recently generated groups (for pronoun generation), and a blackboard containing communication goals and general world knowledge, including the relationship of the speaker to the things described by the groups, and a history of items generated. By specifying additional features (and in a future version, removing features), the feature mapper can in effect decide which rules get fired, precisely controlling the output text. This affords reactive control over generation.

6. Discussion

GLINDA is a real-time on-line generation system. Running in CMU Common Lisp on an IBM RT-APC, a 3-MIPS machine with a good LISP implementation, GLINDA generates an average of 2 simple clauses per second, or a conservative estimate of greater than 500 words per minute. Figure 6-1 gives some sample timing data.

```
<cl> (time (generate test '((voice passive) (time past)
                            (perfect t) (progressive begin)
                            (modal ability)
                            (truth negative) (contraction aux))))

Mary couldn't have begun being loved by John.
Evaluation took:
  0.629999 seconds of real time,
  0.640625 seconds of user run time,
  0.0 seconds of system run time,
  2 page faults, and
  7548 bytes consed.
```

Figure 6-1: Timing Data

The generator is under 10 pages of lisp code. An additional 15 pages provide the ontology, 450 organization rules and 20 combination rules for a simple semantic domain. This is a total of less than 1800 lines of lisp.

The use of feature propagation to impose grammatical constraints is not new, having already been used in attribute grammars, LFG, and PATR-II, among others [Correa 1986; Kaplan 1982; Shieber 1983]. The major difference is that features are realizable elements which are percolated throughout the system, and are not tied to the categories (there aren't any categories in GLINDA, per se), but are contained within the groups.

The system is capable, in theory, of generating any context free language whose parse trees obey precedence and dominance relations. Feature propagation extends the generative power to that of context-sensitive systems, in a manner similar to GPSG [Gazdar 1981, 1985].

The attachment of organization rules to projectors is similar to the definition of words and concepts in lexical-conceptual structure [Dorr 1987, 1988; Jackendoff 1983; Kornfilt 1989].

The *Subjacency Condition*, which states that movement from node X to node Y in a tree is valid if and only if X and Y are subjacent,[10] is a direct consequence in GLINDA of organization being an operation on a single group that refers only to items of the group. No special procedure checks that the subjacency condition is satisfied; instead, it is by necessity satisfied since output violating this condition cannot be produced by GLINDA. Others have previously claimed similar epiphenomenal properties for parsers [Marcus 1980; Berwick and Weinberg 1984; Correa 1986], but we are aware of no such prior claims for generation.

7. Further Research

There are many areas of future research concerning the GLINDA architecture. We list some of them here.

While we have concentrated on phrase, lexical, and morphological levels of the language hierarchy, nothing precludes us from applying the architecture to higher and lower level language constructs. This is an obvious area in which to push this research.

Along the same lines, we should extend the grammar and feature-mapping rules to raise the level of the interface, producing more complex knobs. This would allow us to study generating text from high level features for style and pragmatics.

Adding a kind of backtracking to the system will be important for text planning. Since the generation rules are like simple productions in the way they control the flow of features, a rule could signal an error by returning an error feature. When an error feature is encountered, "backtracking" could occur in the form of restarts and interruptions, or the buffering and erasing of output, in much the same way as people do when speaking.

The organization rules present a fascinating opportunity to experiment with Universal Grammar and other parametrized language models. The exploration of automatic generation of rules from higher level descriptions of a language should be fun. Better yet, it may be fruitful to have the generator directly implement the parametrized model.

Many of the rules seem to be symmetric in structure. Perhaps we may be able to invert the generation algorithm to obtain a parser.

Layering of rules may be a good means of simulating learning. For example, we could have whole layers of rules "drop out" when the STRESS feature is set to high.

---

[10] The Government-Binding Theory definition of subjacency is: Y is said to be *subjacent* to X if and only if X is superior to Y and there is at most one bounding node C $\neq$ Y such that C contains Y and C does not contain X [Chomsky 1981; Ambrose-Grillet 1978]. Insofar as GLINDA groups correspond in an informal sense to the bounding nodes (e.g., S and NP nodes) of Government-Binding Theory, groups X and Y are *subjacent* if and only if X and Y are siblings. Then our definition here of the Subjacency Condition is analogous to Chomsky's definition: A transformation may not move an item within the cyclic category [bounding node] A to a position within the cyclic category C that includes category B including A and included in C [Chomsky 1975].

8. Acknowledgements

I would like to thank Joe Bates, my advisor and head of the Oz Project. Eric H. Nyberg III and Sergei Nirenberg of the CMU Center for Machine Translation and Bob Berwick and Michael Brent of the MIT AI Lab have provided advice on natural language processing and computational linguistics issues. Eduard Hovy of the USC Information Sciences Institute has helped with matters of pragmatics and style.

The name GLINDA was chosen because the acronym GWNLO may stand for either "The Good Witch of the North in the Land of Oz", who is Glinda, or "The Generation of Wonderful Natural Language Output".

9. References

[Ambrose-Grillet 1978] Ambrose-Grillet, Jeanne, *Glossary of Transformational Grammar*, Newbury House Publishers, 1978.

[Bates 1989] Bates, Joseph, *Oz Project Overview and Schedule*, internal memo, 1989.

[Berwick 1984] Berwick, Robert C., and Amy Weinberg, *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*, MIT Press, 1984.

[Brent 1990] Brent, Michael R., *A Simplified Theory of Tense Representation and Constraints on their Composition*, in Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh, 1990.

[Chomsky 1975] Chomsky, Noam, *Reflections on Language*, Pantheon Books, New York, 1975.

[Chomsky 1981] Chomsky, Noam, *Lectures on Government and Binding*, Foris Publications, Dordrecht, 1981.

[Chomsky 1982] Chomsky, Noam, *Some Concepts and Consequences of the Theory of Government and Binding*, MIT Press, 1982.

[Comrie 1985] Comrie, Bernard, *Tense*, Cambridge University Press, 1985.

[Correa 1986] Correa, Nelson, *An Attribute-Grammar Implementation of Government-Binding Theory*, Syracuse University, 1986.

[Dahl 1985] Dahl, Osten, *Tense and Aspect Systems*, Basil Blackwell Ltd., New York, 1985.

[Dorr 1987] Dorr, Bonnie J., MIT AI Lab Memo 1000, 1987 (master’s thesis on UNITRAN).

[Dorr 1988] Dorr, Bonnie J., MIT AI Lab Memo 1015, 1988 (PhD proposal).

[Dorr 1990] Dorr, Bonnie J., *Lexical-Conceptual Structure and Machine Translation*, PhD Thesis, MIT, Cambridge, MA, May 1990.

[Gazdar 1981] Gazdar, Gerald, "Unbounded Dependencies and Coordinate Structure", *Linguistic Inquiry*, Volume 12, Number 2, MIT, 1981.

[Gazdar 1985] Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag, *Generalized Phrase Structure Grammar*, Harvard University Press, 1985.

[Hinrichs 1988] Hinrichs, Erhard W., *Tense, Quantifiers, and Contexts*, Computational Linguistics 14:2, Special Issue on Tense and Aspect, June 1988.

[Hovy 1988] Hovy, Eduard H., *Generating Natural Language Under Pragmatic Constraints*, Lawrence Erlbaum Associates, New Jersey, 1988.

[Jackendoff 1972] Jackendoff, Ray, *Semantic Interpretation in Generative Grammar*, MIT Press, 1972.

[Jackendoff 1983] Jackendoff, Ray, *Semantics and Cognition*, MIT Press, 1983.

[Kaplan 1982] Kaplan, Ronald M., and Joan Bresnan, "Lexical-Functional Grammar: A Formal System for Grammatical Representation", Chapter 4 of *The Mental Representation of Grammatical Relations*, Joan Bresnan, ed., MIT Press, 1982.

[Karttunen 1983] Karttunen, Lauri, and Kent Wittenberg, *A Two-level Morphological Analysis of English*, Texas Linguistic Forum 22, 1983.

[Kay 1983] Kay, M., *Unification Grammar*, Xerox Palo Alto Research Center Tech Report, Palo Alto, CA, 1983.

[Knuth 1975] Knuth, Donald E., *The Art of Computer Programming*, Volume 3, "Sorting and Searching", Section 6.3, pp 481-499. Addison-Wesley Publishing Company, Reading, MA, 1975.

[Kornfilt 1989] Kornfilt, J., and Correa, N., *Conceptual Structure and its relation to the structure of lexical entries,* Syracuse University, 1989.

[Laurel 1986] Laurel, Brenda K., *Toward the design of a computer-based interactive fantasy system,* Phd Thesis, Ohio State University, 1986.

[Mann 1982] Mann, William C., *The Anatomy of a Systemic Choice,* USC Information Sciences Institute, Tech Report ISI/RR-82-104.

[Marcus 1980] Marcus, Mitchell, *A Theory of Syntactic Recognition for Natural Language*, MIT Press, 1980.

[McDonald 1980] McDonald, David D., *Natural language production as a process of decision making under constraint*, PhD Thesis, Cambridge, MA, MIT Press, 1980.

[McDonald 1983] McDonald, David D., "Natural Language Generation as a Computational Problem: An Introduction", in Brady, Michael, and Robert C. Berwick, eds., *Computational Models of Discourse,* MIT Press, 1983.

[McDonald 1985] McDonald, David D., and Pustejovsky, J.D., "Description-directed natural language generation", in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 799-805, Los Angeles, CA, August 1985.

[Moens 1988] Moens, Marc, and Mark Steedman, *Temporal Ontology and Temporal Reference*, Computational Linguistics 14:2, Special Issue on Tense and Aspect, June 1988.

[Nakhimovsky 1988] Nakhimovsky, Alexander, *Aspect, Aspectual Class, and the Temporal Structure of Narrative*, Computational Linguistics 14:2, Special Issue on Tense and Aspect, June 1988.

[Nyberg 1988] Nyberg (3d), Eric H., *Generation Mapping Rules Version 1.0*, CMU-CMT-88-MEMO.

[Passonneau 1988] Passonneau, Rebecca J., *A Computational Model of the Semantics of Tense and Aspect*, Computational Linguistics 14:2, Special Issue on Tense and Aspect, June 1988.

[Radford 1988] Radford, Andrew, *Transformational Grammar,* Cambridge University Press, 1988.

[Reichenbach 1947] Reichenbach, Hans, *Elements of Symbolic Logic,* The Free Press, New York, 1947 (reprinted 1966).

[Shieber 1983] Shieber, S.M., H. Uszkoreit, F.C.N. Pereira, J.J. Robinson, and M. Tyson, *The Formalism and Implementation of PATR-II,* in "Research on Interactive Acquisition and Use of Knowledge", SRI International, Menlo Park, CA, 1983.

[Smith & Bates 1989] Smith, Sean, and Joe Bates, *Towards a Theory of Narrative for Interactive Fiction*, CMU Technical Report CMU-CS-89-121, February 1989.

[Tomita 1988] Tomita, Masaru, and Eric H. Nyberg, 3rd, *Generation Kit and Transformation Kit, Version 3.2*, CMU-CMT-88-MEMO.

[Van Riemsdijk 1986] Van Riemsdijk, Henk, and Edwin Williams, *An Introduction to the Theory of Grammar*, MIT Press, 1986.

[Webber 1988] Webber, Bonnie L., *Tense as Discourse Anaphor*, Computational Linguistics 14:2, Special Issue on Tense and Aspect, June 1988.

Table of Contents

List of Figures