GENESIS-II: A VERSATILE SYSTEM FOR LANGUAGE GENERATION IN CONVERSATIONAL SYSTEM APPLICATIONS[1]

Lauren Baptist and Stephanie Seneff

Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA
{lmb, seneff}@sls.lcs.mit.edu

ABSTRACT

Language generation is a fundamental component of dialogue systems. Over the past year, we have developed a new generation module for conversational systems developed at MIT using the GALAXY architecture. Our generator, which we call GENESIS-II, resolves many of the shortcomings of its predecessor, GENESIS. GENESIS-II makes it substantially easier for users to specify generation, and the generation output is often of a higher quality. In particular, GENESIS-II has improved the ease and quality of generation in foreign languages (Japanese, Chinese, Spanish) and non-traditional languages (SQL, HTML, speech waveforms). In this paper, we focus on the more advanced features of our system.

1. Introduction

Language generation can be defined as the process of transforming a meaning representation into a target string. In conversational systems, this most obviously means preparing a well-formed string to be spoken to the user, as a response to a question. However, a versatile generation module in a conversational system can also be useful on a broader spectrum, for such tasks as the generation of database queries in the formal language SQL and the preparation of a hyperlinked HTML table of a list of items being displayed in a display window [7]. Furthermore, a generation system can play a critical role in language translation tasks, particularly when an interlingual approach is adopted. This is probably the most challenging generation task, since, in some cases, the meaning representation may need to be transformed in some way in order for it to accurately accommodate the target language's generation requirements [8].

In conversational systems, it is unclear how to best represent the knowledge contained in a user's query. A hierarchical representation that preserves the syntactic structure[2] is a very accurate encoding of the user's query, but often the detailed structure is not really needed for the task of answering the question. For pragmatic reasons, many researchers have adopted a flattened "*e*-form" (electronic form) representation of the facts contained in users' queries, in order to simplify the process of further decoding in later stages of analysis. Similarly, the information that the system needs to provide to the user can be conveniently represented as an *e*-form. An example of an *e*-form in the flight domain is shown in Figure 1.

```
{c flight_leg
   :airline "UA"
   :flight_number 94
   :source "BOS"
   :depart_time "3:00 p.m."
}
```

Figure 1: Example of an *e*-form in the flight domain.

As suggested in a review paper by Hovy [4], available generation systems have generally divided along the dimension of "linguistic" vs. "nonlinguistic" approaches, which are critically tied to the strategy for encoding the linguistic information (essentially, linguistic hierarchy v.s. *e*-form). In developing conversational systems, our approach has been to first parse a user query into a hierarchical linguistic structure that we call a "semantic frame." Our discourse inheritance mechanism depends upon the hierarchy to accurately carry out its tasks. Hierarchy is also necessary to produce an accurate paraphrase of the user query to be presented in a display window. After inheritance, the frame is converted into a flattened *e*-form, which is then sent to the turn manager for further analysis.

In our multilingual JUPITER weather domain, we find that a linguistic representation of the weather reports is absolutely essential in order to obtain accurate translations into the three target languages of our current research systems: Mandarin [8], Japanese [6], and Spanish. An example semantic frame for a sentence in the weather domain is given in Figure 2.

Because our systems have always included both linguistic and nonlinguistic representations, our generation server has evolved to be able to handle both styles of meaning representation. In fact, one of its functions is to convert a semantic frame into an equivalent *e*-form.

We have been developing multilingual conversational systems for over ten years, and the various generation activities outlined above have been carried out by a generation system we call GENESIS [3]. As new needs arose, GENESIS was augmented to be able to handle them, and thus, it slowly evolved to have increasing capability. As a consequence of this evolutionary process, however, GENESIS became quite idiosyncratic in its usage specifications, and we eventually reached a point where new requirements for translingual generation could not be met without a substantial overhaul of the core engine. We therefore made the decision to completely redesign the system, with the aim of enhanc-

[1]This work was supported by a fellowship and contract from Nippon Telegraph & Telephone.

[2]For example, distinguishing attributes which modify a noun phrase versus those that are contained in a top-level main predicate or in a subordinate clause.

Sentence:
Rain likely in the morning followed by light sleet and snow ending in the early evening

Semantic Frame:

```
{c weather_event
   :topic
     {q precip_act :name "rain"
       :pred {p probability
                :temp_qualifier "likely"
                :pred {p in_time
                         :topic
                           {q time_of_day
                              :name "morning"
                              :quantifier "def"
                           } } } }
   :pred {p followed_by
            :topic
              {q precip_act :name "sleet"
                 :qualifier "light"
                 :and {q precip_act
                         :name "snow" } }
            :pred {p ending
                     :pred
                        {p in_time
                           :topic
                              {q time_of_day
                                 :modifier "early"
                                 :name "evening"
                                 :quantifier "def"
                              } } } } }
```

Figure 2: Example of a semantic frame preserving syntactic structure as well as semantics, for a weather report. GENESIS-II exactly reproduces this sentence in the English paraphrase.

ing its ease of use and enabling more sophisticated capabilities.

The resulting system, GENESIS-II, has fulfilled our main goals, which were to provide very straightforward methods for simple generation tasks, while also supporting the capability of handling more challenging generation requirements, such as movement phenomena, propagation of linguistic features, structural reorganization, and the specification of word sense. We have focused on creating an expressive language with generalized mechanisms by carefully designing notations and commands in a generic way, so that they would enjoy wider utility.

We have thus far used GENESIS-II in a number of specific domains and languages, both formal and natural. As mentioned previously, in our JUPITER domain, weather reports are being translated into three languages besides English [8, 6]. In the realm of formal languages, we use GENESIS-II both to convert a linguistic frame into an *e*-form and to generate database queries, often represented in SQL [7]. GENESIS-II is also able to create hyperlinked HTML tables of lists of database retrievals. Finally, it is used to produce, along with a text of the system responses, a marked-up string for our ENVOICE corpus-based speech synthesis [9].

We begin this paper with a high-level overview of the GENESIS-II system. We follow with an example showing the mechanisms for response generation from a simple *e*-form. We then describe some of GENESIS-II's features that enable it to handle hard problems in generation. We conclude with a discussion of the difficult problem of evaluating generation systems. Due to space limitations, we will only be able to highlight some of GENESIS-II's features here. The interested reader is referred to [2] for a more thorough overview of the system.

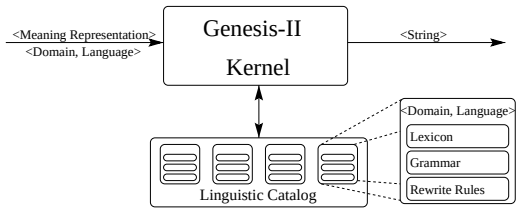

Figure 3: GENESIS-II's Architecture

(1)	flight_leg	>airline_flight >leaves_from
(2)	airline_flight	:airline "flight" :flight_number
(3)	leaves_from	"leaves" >from_source >at_dpt_time
(4)	at_dpt_time	"at" :depart_time
(5)	from_source	"from" :source

Figure 4: Example rules governing generation of an English string from the *e*-form of Figure 1. Rules are numbered for expository purposes.

2. Overview and Example

Like its predecessor, GENESIS-II's high-level architecture consists of a kernel and a linguistic catalog, as depicted in Figure 3. For each unique domain and language, the linguistic catalog provides a lexicon, a grammar, and a list of rewrite rules, which together control the string generation for frames in that domain. The kernel is the core C code that, at run time, converts a meaning representation into a string by recursively evaluating the grammar rules and using the lexicon to determine surface realizations, which include inflectional endings (gender, number, mode, etc.) and word-sense disambiguation.

Figure 4 shows a set of grammar rules that govern the generation of the English string, "United flight 94 leaves from Boston at 3:00 p.m.," from the *e*-form of Figure 1. Generation begins with (1), a lookup on the clause named `flight_leg`, which instructs it to first apply the `airline_flight` rule, followed by the `leaves_from` rule. In (2), the system generates "United flight 94," by looking up the expansion for "UA" in the vocabulary file. Similarly, (3) produces "leaves from Boston at 3:00 p.m." If the `:source` were missing, the entire `from_source` message would be omitted, and likewise for the `at_dpt_time` message, so that well-formed substrings would still be generated. GENESIS-II is extremely efficient, because each generation step is controlled by a lookup on an index in a binary search, whether in the rule file or the lexicon. GENESIS-II's mechanisms for generating from *e*-forms closely resemble those described in [1].

3. Generation Methodology

As suggested by the example in Section 2, GENESIS-II generates a string by executing grammar rules in a top-down fashion, beginning with the highest level clause and following the threads set out by the recursive grammar rules. We begin here by describing the unified generation mechanisms that make the language of GENESIS-II much more straightforward than that of the original system. This is followed by a discussion of mechanisms for grouping grammar rules. Finally, we discuss briefly

the internals of GENESIS-II's generation mechanisms for feature propagation.

Unified Generation Mechanisms The original GENESIS system has several shortcomings in terms of both irregularities and inflexibilities. First of all, it has unique generation mechanisms for each of the types of constituents it encounters. Specifically, clauses, predicates, topics, lists, and keywords all have different generation mechanisms. Therefore, application developers must learn several metalanguages in order to specify the generation of all types of constituents. Furthermore, conjunction generation includes rigid aspects that are embedded in the code, and the ordering of modifiers cannot be specified in a context-dependent way.

GENESIS-II resolves all of these deficiencies. First, it provides a unified generation framework for all types of constituents, making it much easier to learn how to write the generation rules. Furthermore, it is possible to specify context-dependent ordering of modifiers relative to the name of the modified constituent. Conjunctions are handled in a natural way using the standard mechanisms for keys and frames.

Grouping Grammar Rules The rule files for GENESIS are generally large, because rules governing the generation of almost every possible constituent must be explicitly defined. GENESIS-II allows generation patterns to be generalized in several ways. First, every major linguistic constituent has a default generation rule, which applies whenever no explicit entry is available under the constituent's own name. Furthermore, groups of constituents which generate according to identical rule patterns can be affiliated and thus linked to a single shared generation rule.

Feature Propagation To propagate critical information among the frame's constituents, a linguistic environment, called the "`info frame`," is updated and referenced throughout the generation process. The `info frame` serves as a conduit to communicate syntactic, semantic, and prosodic contexts among the frame's hierarchical constituents. For example, if a constituent is plural, GENESIS-II updates the `info frame` to reflect this number information. A subsequently processed frame can then use this information, thus enforcing, for example, number agreement between a topic and its main predicates. GENESIS-II also allows grammar rules to set context-sensitive variables directly in the `info frame`, typically to establish context for the descendents. This is useful for setting both semantic and prosodic context for vocabulary selection. For example, a rule in Spanish might set a "`$wind`" variable when the constituent "wind" is encountered, so that translations for various words can be tailored for the wind "environment". In this way and others, information can be propagated through the hierarchy of the meaning representation.

4. Advanced Features

In this section, we briefly discuss three of GENESIS-II's more sophisticated features. The first one is useful for enumerating lists of items, typically retrieved from a database. The other two concern issues that are particularly relevant for translingual generation. These include techniques for word-sense disambiguation and for major restructuring of information in a frame when it is necessary for fluency.

```
{c speak_departure
   :common_airline "US"
   :fl_list ( {c departing_flight
                 :departure_time "6:15"
                 :depart_xm "a.m."
                 :arrival_time "8:03"
                 :arrive_xm "a.m." }
              {c departing_flight
                 :departure_time "8:50"
                 :depart_xm "a.m."
                 :arrival_time "10:45"
                 :arrive_xm "a.m." }
              {c departing_flight
                 :departure_time "11:20"
                 :depart_xm "a.m."
                 :arrival_time "1:11"
                 :arrive_xm "p.m." } )
   :num_nonstops 3 }
```

Figure 5: An *e*-form response from the MERCURY flight-reservation domain, showing a typical list structure.

Generating from Lists GENESIS-II provides a simple framework for generating strings from lists. Lists are prevalent in database retrieval domains, and therefore, convenient mechanisms for speaking about them are essential. The position of an item in a list often influences its surface realization. For example, for many lists, all but the first and last entries are preceded by a comma, and the last is preceded by the word "and." GENESIS-II handles these distinctions by positionally tagging list items.

Consider, for example, the *e*-form in Figure 5. The key value associated with the keyword `:fl_list` in the frame is a list of `departing_flight` frames, and the English generation string might be of the form, "I have 3 flights on U.S. Airways: *one* departing at 6:15 a.m. and arriving at 8:03 a.m., *another* departing at ..., and *the last* departing at ... p.m." The different references to the three items, "one," "another," and "the last," can be specified through tags marking special positions such as "first" and "last."

Context-dependent Selectors As suggested earlier, GENESIS-II provides the ability to set context-sensitive selectors. Conceptually, a selector is a variable that a grammar rule or a vocabulary item sets, for reference by grammar rules and vocabulary items later in the generation process.

We motivate the need for selectors by discussing word-sense disambiguation—a problem in language generation in general, but especially in machine translation. That is, a word in one language may have several context-dependent translations into another language, and a translator—machine or otherwise—must select the appropriate word sense. For example, quite often, a preposition in one language has several distinct translations in another language. To illustrate selection, consider a specific example in the Spanish JUPITER domain, where selectors are essential for specifying the correct generation for the preposition "into." The Spanish lexical entry for "into" is as follows:

into "a" $:temp "hasta" $:until "hasta" $:loc "hacia"

```
{c truth
   :mode "finite"
   :number "third"
   :aux "do"
   :topic {q flight
             :quantifier "def" }
   :pred {p arrival_time
            :topic {q time
                      :quantifier "what" }}}
```

Figure 6: A semantic frame representation for "*What time* does the flight arrive?"

By default, "into" translates into "a." However, a generation rule for a location frame would set a `$:loc` selector directly in its grammar rule, thus selecting for "hacia" as the appropriate translation. GENESIS-II maintains an ordered list of active selectors, established through the semantic frame hierarchy, so that the most specific relevant selector applies for any particular situation.

With such techniques, GENESIS-II's users can accurately resolve the semantic ambiguities that arise in translation. Furthermore, we were able to use selectors not only for *semantic* disambiguation, but also for *prosodic* selection, choosing a `$low` or `$high` prosodic context as appropriate.

Reorganizing the frame hierarchy Our experience with translating a semantic frame produced from English into other languages has shown that meaning representations are often difficult to interpret when working with translingual situations. One of the primary difficulties arises from the word-ordering differences between languages. A constituent that is deep in the meaning representation may need to appear at an unusual location in the generation string in some domains and languages, including English. In English wh-queries, for example, an embedded noun phrase often appears at the beginning of the generation string. See, for example, Figure 6, where the wh-quantified noun phrase, "what time," appears sentence-initially in the English generation string.

To resolve such issues, we created several mechanisms for restructuring the actual frame hierarchy. One set of mechanisms contains `pull` commands, which allow higher-level frames to extract constituents from *deeper* in the frame structure. For example, in the semantic frame of Figure 2, a `pull` mechanism can be invoked to pre-generate the "likely" qualifier so as to place it at the beginning of the generation string for `precip_act`, an appropriate position in Spanish.

Another set of mechanisms allows constituents to `push` aside certain generation substrings for deferral to a later stage, often at the time of construction of a *higher* constituent in the frame structure. This mechanism is particularly effective for wh-query movement. Such mechanisms give users the power they need to generate correctly in what were once impossible situations.

5. Evaluation and Summary

The task of evaluating a generation system remains a challenging research problem. Evaluation can be assessed along several dimensions, such as the quality of the strings generated, the flexibility and ease of use of the system, and the speed and memory requirements. At this point, the strongest evidence of our system's superiority over the original GENESIS system is the fact that researchers in our group now generally prefer it over its predecessor. The many domains, languages, and frame representation styles for which it is being used can be taken as evidence of its general utility. We have developed knowledge bases in English, Spanish, Chinese, and Japanese, as well as in non-traditional languages, such as speech waveforms and HTML.

It is clear that GENESIS-II has at the least two quantitative advantages. First, as mentioned in the introduction, the GENESIS-II framework allows more succinct rule expressiveness. In the SQL language for JUPITER, for example, the grammar size was reduced by 50%. A second clear advantage is that GENESIS-II looks up rules in the lexicon and grammar using a binary search, which, in theory, should be faster than the processing requirements of a statistically-based system, such as the one in [5].

With its simple, yet flexible, framework GENESIS-II is a powerful generator for conversational systems. Its mechanisms for a unified treatment of all constituents, for grouping elements to share common grammar rules, for conveniently generating lists of items, for frame manipulation, and for context-sensitive selection make even the challenging task of translingual generation feasible.

6. REFERENCES

1. S. Axelrod, "Natural Language Generation in the IBM Flight Information System," *Proceedings of the Workshop on Conversational Systems at ANLP-NAACL*, 2000.

2. L. Baptist, "GENESIS-II: A Language Generation Module for Conversational Systems," SM. Thesis, MIT, 2000.

3. J. Glass, J. Polifroni, and S. Seneff, "Multilingual Language Generation Across Multiple Domains," *Proceedings of ICSLP*, 1994.

4. E.H. Hovy, "Language Generation," *Survey of the State of the Art in Human Language Technology*, 1996.

5. A.H. Oh and A.I. Rudnicky, "Stochastic Language Generation for Spoken Dialogue Systems," *Proceedings of the Workshop on Conversational Systems at ANLP-NAACL*, 2000.

6. S. Seneff, J. Glass, T.J. Hazen, Y. Minami, J. Polifroni, and V. Zue, "A Japanese Spoken Dialogue System in the Weather Domain," *NTTR&D* Vol. 49, pp.365 – 371, 2000.

7. S. Seneff and J. Polifroni, "Formal and Natural Language Generation in the MERCURY Conversational System," *These proceedings*, 2000.

8. C. Wang, S. Cyphers, X. Mou, J. Polifroni, S. Seneff, J. Yi and V. Zue, "MuXing: A Telephone-access Mandarin Conversational System in the Weather Domain," *These proceedings*, 2000.

9. J. Yi, J. Glass, and I. Hetherington, "A Flexible, Scalable Finite-State Transducer Architecture for Natural-Sounding Speech Synthesis," *These proceedings*, 2000.