NEW YORK UNIVERSITY
DESCRIPTION OF THE PROTEUS SYSTEM AS USED FOR MUC-3

Ralph Grishman, John Sterling, and Catherine Macleod

The PROTEUS Project
Computer Science Department
New York University
715 Broadway, 7th Floor
New York, NY 10003

{grishman,sterling,macleod}@cs.nyu.edu

HISTORY

The PROTEUS system which we have used for MUC-3 has three main components: a syntactic analyzer, a semantic analyzer, and a template generator.

The PROTEUS Syntactic Analyzer was developed starting in the fall of 1984 as a common base for all the applications of the PROTEUS Project. Many aspects of its design reflect its heritage in the Linguistic String Parser, previously developed and still in use at New York University. The current system, including the Restriction Language compiler, the lexical analyzer, and the parser proper, comprise approximately 4500 lines of Common Lisp.

The Semantic Analyzer was initially developed in 1987 for the MUCK-I (RAINFORMs) application, extended for the MUCK-II (OPREPs) application, and further revised for the current evaluation. It currently consists of about 3000 lines of Common Lisp (excluding the domain-specific information).

The Template Generator was written from scratch for the MUC-3 task; it is about 1200 lines of Common Lisp..

STAGES OF PROCESSING

The text goes through the five major stages of processing: lexical analysis, syntactic analysis, semantic analysis, reference resolution, and template generation (see Figure 1). In addition, some restructuring of the logical form is performed both after semantic analysis and after reference resolution (only the restructuring after reference resolution is shown in Figure 1). Processing is basically sequential: each sentence goes through lexical, syntactic, and semantic analysis and reference resolution; the logical form for the entire message is then fed to template generation. However, semantic (selectional) checking is performed during syntactic analysis, employing essentially the same code later used for semantic analysis.

Each of these stages is described in a section which follows.

LEXICAL ANALYSIS

Dictionary Format

Our dictionaries contain only syntactic information: the parts of speech for each word, information about the complement structure of verbs, distributional information (e.g., for adjectives and adverbs), etc. We follow closely the set of syntactic features established for the NYU Linguistic String Parser. This information is entered in LISP form using `noun`, `verb`, `adjective`, and `adverb` macros for the open-class words, and a `word` macro for other parts of speech:

```
(ADVERB "ABRUPTLY" :ATTRIBUTES (DSA))
(ADJECTIVE "ABRUPT")
(NOUN :ROOT "ABSCESS" :ATTRIBUTES (NCOUNT))
```

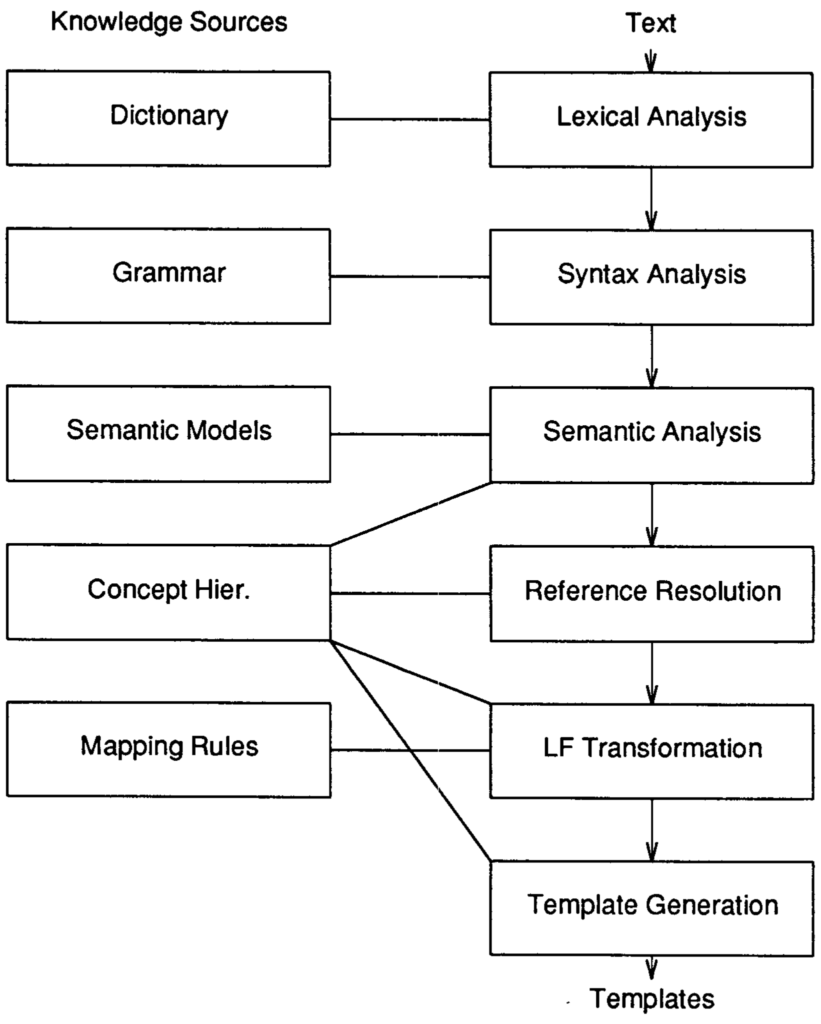

Figure 1. Structure of the Proteus System as used for MUC-3

```
(VERB :ROOT "ABSCOND" :OBJLIST (NULLOBJ PN (PVAL (FROM WITH))))
```

The `noun` and `verb` macros automatically generate the regular inflectional forms.

Dictionary Files

The primary source of our dictionary information about open-class words (nouns, verbs, adjectives, and adverbs) is the machine-readable version of the Oxford Advanced Learner's Dictionary ("OALD"). We have written programs which take the SGML (Standard Generalized Markup Language) version of the dictionary, extract information on inflections, parts of speech, and verb subcategorization (including information on adverbial particles and prepositions gleaned from the examples), and generate the LISP-ified form shown above. This is supplemented by a manually-coded dictionary (about 500 lines) for closed-class words and a few very common words.

For MUC-3 we used several additional dictionaries. There was a dictionary (about 800 lines) for English words not defined in the OALD, or not adequately defined or too richly defined there. In addition, we extracted from the text and templates lists of organizations, locations, and proper names, and prepared small dictionaries for each (about 2000 lines total).

Lookup

The text reader splits the input text into tokens and then attempts to assign to each token (or sequence of tokens, in the case of an idiom) a definition (part of speech and syntactic attributes). The matching process proceeds in four steps: dictionary lookup, lexical pattern matching, spelling correction, and prefix stripping. Dictionary lookup immediately retrieves definitions assigned by any of the dictionaries (including inflected forms), while lexical pattern matching is used to identify a variety of specialized patterns, such as numbers, dates, times, and possessive forms.

If neither dictionary lookup nor lexical pattern matching is successful, spelling correction and prefix stripping are attempted. Based on an analysis of the errors we found, we have used for MUC-3 a rather conservative spelling corrector, which identifies an input token as a misspelled form of a dictionary entry only if one of the two has a single instance of a letter while the other has a doubled instance of the letter (e.g., "mispelled" and "misspelled").[1]

The prefix stripper attempts to identify the token as a combination of a prefix and a word defined in the dictionary. We currently use a list of 17 prefixes, including standard English ones like "un" and MUC-3 specials like "narco-".

If all of these procedures fail, the word is tagged as a proper noun (name), since we found that most of our remaining undefined words were names.

Filtering

In order to avoid full processing of sentences which would make no contribution to the templates, we perform a keyword-based filtering at the sentence level: if a sentence contains no key terms, it is skipped. This filtering is done after lexical analysis because the lexical analysis has identified the root form of all inflected words; these root forms provide links into the semantic hierarchy. The filtering can therefore be specified in terms of a small number of word classes, one of which must be present for the sentence to be worth processing.

SYNTACTIC ANALYSIS

Syntactic analysis involves two stages of processing: parsing and syntactic regularization. At the core of the system is an active chart parser. The grammar is an augmented context-free grammar, consisting of BNF rules plus procedural restrictions which check grammatical constraints not easily captured in the BNF rules. Most restrictions are stated in PROTEUS Restriction Language (a variant of the language developed for the Linguistic String Parser) and translated into LISP; a few are coded directly in LISP [1]. For example, the count noun restriction (that singular countable nouns have a determiner) is stated as

```
WCOUNT = IN LNR AFTER NVAR:
   IF BOTH CORE Xcore IS NCOUNT AND Xcore IS SINGULAR
   THEN IN LN, TPOS IS NOT EMPTY.
```

Associated with each BNF rule is a regularization rule, which computes the regularized form of each node in the parse tree from the regularized forms of its immediate constituents. These regularization rules are based on lambda-reduction, as in GPSG. The primary function of syntactic regularization is to reduce all clauses to a standard form consisting of aspect and tense markers, the operator (verb or adjective), and syntactically marked cases. For example, the definition of `assertion`, the basic S structure in our grammar, is

```
<assertion>     ::= <sa> <subject> <sa> <verb> <sa> <object> <sa>
                        :(s !(<object> <subject> <verb> <sa*>)).
```

Here the portion after the single colon defines the regularized structure.

[1] The more standard corrector we used for MUCK-2, which allowed for any single insertion, deletion, transposition, or substitution, gave too many incorrect matches.

Coordinate conjunction is introduced by a metarule (as in GPSG), which is applied to the context-free components of the grammar prior to parsing. The regularization procedure expands any conjunction into a conjuntion of clauses or of noun phrases.

The output of the parser for the first sentence of DEV-0099, "POLICE HAVE REPORTED THAT TERRORISTS TONIGHT BOMBED THE EMBASSIES OF THE PRC AND THE SOVIET UNION.", is

```
(SENTENCE
 (CENTERS
  (CENTER
   (ASSERTION (SUBJECT (NSTG (LNR (NVAR (N "POLICE")))))
    (VERB (LTVR (TV "HAVE")))
    (OBJECT
     (VENO (LVENR (VEN "REPORTED"))
      (OBJECT
       (THATS ("THAT" "THAT")
        (ASSERTION (SUBJECT (NSTG (LNR (NVAR (N "TERRORISTS")))))
         (SA (SA-VAL (NSTGT (NSTG (LNR (NVAR (N "TONIGHT")))))))
         (VERB (LTVR (TV "BOMBED")))
         (OBJECT
          (NSTGO
           (NSTG
            (LNR (LN (TPOS (LTR (T "THE")))) (NVAR (N "EMBASSIES"))
             (RN
              (RN-VAL
               (PN (P "OF")
                (NSTGO
                 (NSTG
                  (LNR
                   (LNR (LN (TPOS (LTR (T "THE"))))
                    (NVAR (NAMESTG (LNAMER (N "PRC")))))
                   (CONJ-WORD ("AND" "AND"))
                   (LNR (LN (TPOS (LTR (T "THE"))))
                    (NVAR (NAMESTG (LNAMER (N "SOVIET" "UNION")))))))))))))))))))))
 (ENDMARK ("." ".")))
```

and the corresponding regularized structure is

```
(S PERF REPORT (SUBJECT (NP POLICE PLURAL (SN NP1227)))
 (OBJECT
  (S PAST BOMB (SUBJECT (NP TERRORIST PLURAL (SN NP1238)))
   (OBJECT
    (NP EMBASSY PLURAL (SN NP1242) (T-POS THE)
     (OF
      (AND (NP PRC SINGULAR (SN NP1231) (T-POS THE))
           (NP USSR SINGULAR (SN NP1235) (T-POS THE))))))
   (PREP (NP TONIGHT SINGULAR (SN NP1237))))))
```

The system uses a chart parser operating top-down, left-to-right. As edges are completed (i.e., as nodes of the parse tree are built), restrictions associated with those productions are invoked to assign and test features of the parse tree nodes. If a restriction fails, that edge is not added to the chart. When certain levels of the tree are complete (those producing noun phrase and clause structures), the regularization rules are invoked to compute a regularized structure for the partial parse, and selection is invoked to verify the semantic well-formedness of the structure (as noted earlier, selection uses the same "semantic analysis" code subsequently employed to translate the tree into logical form).

One unusual feature of the parser is its weighting capability. Restrictions may assign scores to nodes; the parser will perform a best-first search for the parse tree with the highest score. This scoring is used to implement various preference mechanisms:

- closest attachment of modifiers (we penalize each modifier by the number of words separating it from its head)
- preferred narrow conjoining for clauses (we penalize a conjoined clause structure by the number of words it subsumes)
- preference semantics (selection does not reject a structure, but imposes a heavy penalty if the structure does not match any lexico-semantic model, and a lesser penalty if the structure matches a model but with some operands or modifiers left over) [2,3]
- relaxation of certain syntactic constraints, such as the count noun constraint, adverb position constraints, and comma constraints
- disfavoring (penalizing) headless noun phrases and headless relatives (this is important for parsing efficiency)

The grammar is based on Harris's Linguistic String Theory and adapted from the larger Linguistic String Parser (LSP) grammar developed by Naomi Sager at NYU [4]. The grammar is gradually being enlarged to cover more of the LSP grammar. The current grammar is 1200 lines of BNF and Restriction Language plus 300 lines of Lisp; it includes 150 non-terminals, 365 productions, and 103 restrictions.

Over the course of MUC-2 and MUC-3 we have added several mechanisms for recovering from sentences the grammar cannot fully parse; these are described in our site report.

SEMANTIC ANALYSIS AND REFERENCE RESOLUTION

The output of syntactic analysis goes through semantic analysis and reference resolution and is then added to the accumulating logical form for the message. Following both semantic analysis and reference resolution certain transformations are performed to simplify the logical form. All of this processing makes use of a concept hierarchy which captures the class/subclass/instance relations in the domain.

Semantic analysis uses a set of lexico-semantic models to map the regularized syntactic analysis into a semantic representation. Each model specifies a class of verbs, adjectives, or nouns and a set of operands; for each operand it indicates the possible syntactic case markers, the semantic class of the operand, whether or not the operand is required, and the semantic case to be assigned to the operand in the output representation. For example, the model for "<explosive-object> damages <target>" is

```
(add-clause-model :id 'clause-damage-3
                  :parent 'clause-any
                  :constraint 'damage
                  :operands (list (make-specifier
                                   :marker 'subject
                                   :class 'explosive-object
                                   :case :instrument)
                                  (make-specifier
                                   :marker 'object
                                   :class 'target-entity
                                   :case :patient
                                   :essential-required 'required)))
```

The models are arranged in a shallow hierarchy with inheritance, so that arguments and modifiers which are shared by a class of verbs need only be stated once. The model above inherits only from the most general clause model, `clause-any`, which includes general clausal modifiers such as negation, time, tense, modality, etc. The evaluated MUC-3 system had 98 clause models, 14 nominalization models, and 31 other noun phrase models, a total of about 2000 lines. The class `explosive-object` in the clause model refers to the concept in the concept hierarchy, whose entries have the form:

```
(defconcept explosive-object     :typeof instrument-type)
(defconcept explosive            :typeof explosive-object
                                 :muctype explosive)
(defconcept grenade              :typeof explosive)
(defconcept explosive-charge     :typeof explosive
                                           :alias (dynamite-charge))
(defconcept bomb                           :typeof explosive-object
                                           :muctype bomb)
(defconcept |VEHICLE BOMB|                 :typeof explosive-object
                                           :muctype |VEHICLE BOMB|)
(defconcept car-bomb                       :typeof |VEHICLE BOMB|)
(defconcept bus-bomb                       :typeof |VEHICLE BOMB|)
(defconcept dynamite                       :typeof explosive-object
                                           :alias tnt
                                           :muctype DYNAMITE)
```

There are currently a total of 2098 concepts in the hierarchy, of which 1439 are place names.

The output of semantic analysis is a nested set of entity and event structures, with arguments labeled by keywords primarily designating semantic roles. For the first sentence of DEV-0099, the output is

```
(EVENT
 :TENSE-ASPECT (PERF)
 :IDENTIFIER E0000000085
 :TOP-LEVEL-FLAG T
 :PREDICATE REPORT
 :AGENT (ENTITY
         :SN NP1227
         :SET T
         :IDENTIFIER N0000000086
         :CLASS POLICE)
 :EVENT (EVENT
         :TENSE-ASPECT (PAST)
         :IDENTIFIER E0000000087
         :TOP-LEVEL-FLAG NIL
         :PREDICATE BOMB-EVENT
         :AGENT (ENTITY
                 :SN NP1238
                 :SET T
                 :IDENTIFIER N0000000089
                 :CLASS TERRORIST)
         :PATIENT (ENTITY
                   :NATIONALITY (ENTITY
                                 :IDENTIFIER N0000000093
                                 :CLASS (COUNTRY-NAME COUNTRY)
                                 :SET T
                                 :MEMBERS ((ENTITY
                                            :SN NP1231
                                            :IDENTIFIER N0000000091
                                            :CLASS PRC)
                                           (ENTITY
                                            :SN NP1235
                                            :IDENTIFIER N0000000092
                                            :CLASS USSR)))
                   :SN NP1242
                   :SET T
                   :IDENTIFIER N0000000090
                   :CLASS EMBASSY)
         :INSTRUMENT (ENTITY
                      :ZEROED-P T
                      :IDENTIFIER N0000000088
                      :CLASS EXPLOSIVE-OBJECT)
         :DATE (ENTITY
                        :SN NP1237
                        :IDENTIFIER N0000000094
                        :CLASS TONIGHT)))
```

Reference Resolution

Reference resolution is applied to the output of semantic analysis in order to replace anaphoric noun phrases (representing either events or entities) by appropriate antecedents. Each potential anaphor is compared to prior entities or events, looking for a suitable antecedent such that the class of the anaphor (in the concept hierarchy) is equal to or more general than that of the antecedent, the anaphor and antecedent match in number, the restrictive modifiers in the anaphor have corresponding arguments in the antecedent, and the non-restrictive modifiers (e.g., apposition) of the anaphor are not inconsistent with those of the antecedent. Special tests are provided for names (people may be referred to a subset of their names) and for referring to groups by typical members ("terrorist force" ... "terrorists"). Some further discussion of reference resolution and the subsequent process of template merging is included in a separate paper on discourse analysis in this volume ("Computational Aspects of Discourse in the Context of MUC-3").

Logical Form Transformations

The transformations which are applied after semantic analysis and after reference resolution simplify and regularize the logical form in various ways. For example, if a verb governs an argument of a nominalization, the argument is inserted into the event created from the nominalization: "x conducts the attack", "x claims responsibility for the attack", "x was accused of the attack" etc. are all mapped to "x attacks" (with appropriate settings of the confidence slot). For example, the rule to take "X was accused of Y" and make X the agent of Y is

```
(((event :predicate accusation-event
         :agent ?agent-1
         :event (event :identifier ?id-1 . ?R2)
         . ?R1)
  (event :identifier ?id-1 . ?R4))
 ->
 ((modify 2 '(:agent ?agent-1 :confidence |SUSPECTED OR ACCUSED|))
  (delete 1)))
```

Transformations are also used to expand conjoined structures. For example, the rule to take "<entity> of (<country-1> and <country-2>)" and expand it to "(<entity> of <country-1>) and (<entity> of <country-2>)" is

```
(((entity :nationality (entity :members (?P1 ?P2) . ?R1) . ?R2))
 ->
 ((modify 1 '(:nationality nil
              :set t
              :members ((entity :nationality ?P1 :set nil . ?R2)
                        (entity :nationality ?P2 :set nil . ?R2))))))
```

This rule is used in message TST1-0099, for example, to expand THE EMBASSIES OF THE PRC AND THE SOVIET UNION into THE EMBASSY OF THE PRC AND THE EMBASSY OF THE SOVIET UNION.

There are currently 32 such rules. These transformations are written as productions and applied using a simple data-driven production system interpreter which is part of the PROTEUS system.

TEMPLATE GENERATOR

Once all the sentences in an article have been processed through syntactic and semantic analysis, the resulting logical forms are sent to the template generator. The template generator operates in four stages. First, a frame structure resembling a simplified template (with incident-type, perpetrator, physical-target, human-target, date, location, instrument, physical-effect, and human-effect slots) is generated for each event. Date and location expressions are reduced to a normalized form at this point. In particular, date expressions such as "tonight", "last month", "last April", "a year ago", etc. are replaced by explicit dates or date ranges, based on the dateline of the article. Second, a series of heuristics attempt to merge these frames, merging

- frames referring to a common target
- frames arising from the same sentence
- an effect frame following an attack frame (e.g., "The FMLN attacked the town. Seven civilians died.")

This merging is blocked if the dates or locations are different, the incident types are incompatible, or the perpetrators are incompatible. Third, a series of filters removes frames involving only military targets and those involving events more than two months old. Finally, MUC templates are generated from these frames.

REFERENCES

[1] Grishman, R. PROTEUS Parser Reference Manual. PROTEUS Project Memorandum #4-C, Computer Science Department, New York University, May 1990.

[2] Grishman, R., and Sterling, J. Preference Semantics for Message Understanding. *Proc. DARPA Speech and Natural Language Workshop*, Morgan Kaufman, 1990 (proceedings of the conference at Harwich Port, MA, Oct. 15-18, 1989).

[3] Grishman, R., and Sterling, J., Information Extraction and Semantic Constraints. *Proc. 13th Int'l Conf. Computational Linguistics (COLING 90)*, Helsinki, August 20-25, 1990.

[4] Sager, N. *Natural Language Information Processing*, Addison-Wesley, 1981.

SPONSORSHIP

The development of the entire PROTEUS system has been sponsored primarily by the Defense Advanced Research Projects Agency as part of the Strategic Computing Program, under Contract N00014-85-K-0163 and Grant N00014-90-J-1851 from the Office of Naval Research. Additional support has been received from the National Science Foundation under grant DCR-85-01843 for work on enhancing system robustness.