MITRE-Bedford:
Description of the ALEMBIC System as Used for MUC-4

John Aberdeen, John Burger, Dennis Connolly, Susan Roberts, & Marc Vilain

{*aberdeen, john, decon, suzi, mbv*}*@mitre.org*

The MITRE Corporation
202 Burlington Road
Bedford, MA 01730

BACKGROUND

The ALEMBIC text understanding system fielded at MUC-4 by MITRE–Bedford is primarily based on natural language techniques. ALEMBIC[1] is a research prototype that is intended to explore several major areas of investigation:

- Error recovery, involving primarily issues of semi-parsing and recovery of plausible attachments.
- Robustness, involving primarily issues of uncertain reasoning and tractable inference.
- Self-extensibility, focusing primarily on machine learning of natural language and user-configurable semantics.
- System integration, through SGML (the Standard Generalized Markup Language), both at the level of meaning analysis and at the overall application level.

This investigation is part of an internally-funded research initiative towards processing open source texts (i.e., free natural language texts drawn from broadcast transcripts, news wires, *etc.*). This initiative has been underway for just over half a year, prior to which our group was focusing nearly exclusively on natural language interfaces to expert systems. We are thus newcomers to the MUC data extraction task, and our system is still very much in early phases of development. The system details we present here should thus be taken as preliminary.

OVERALL ARCHITECTURE

The system's underlying architecture, shown in Figure 1, follows a task breakdown used in several other systems that have recently participated in MUC (e.g., PLUM [10] or NLTOOLSET [4]). Processing occurs in three distinct phases: preprocessing, natural language analysis, and application-specific output generation. One of the ways ALEMBIC differs from other MUC systems, however, is in exploiting SGML as the interchange lingua franca between these three processing phases. The intention is to allow system modules whose invocation occurs early in the analysis of a document to record processing results directly in the document through SGML markup. This information then becomes available to subsequent modules as meta-data.

As a result of this SGML-based architecture, the system's overall flow of control is governed from an object-oriented document manager built on top of a Common Lisp port of Goldfarb's public domain SGML parser. For MUC-4, the pre-processing phase thus takes a FBIS message file and normalizes it by recoding it in SGML. The document manager then builds an internal document object by parsing the resulting SGML. The actual content

---

[1] **alembic 1 :** an alchemical apparatus used for distillation **2 :** something that refines or transmutes as if by distillation.

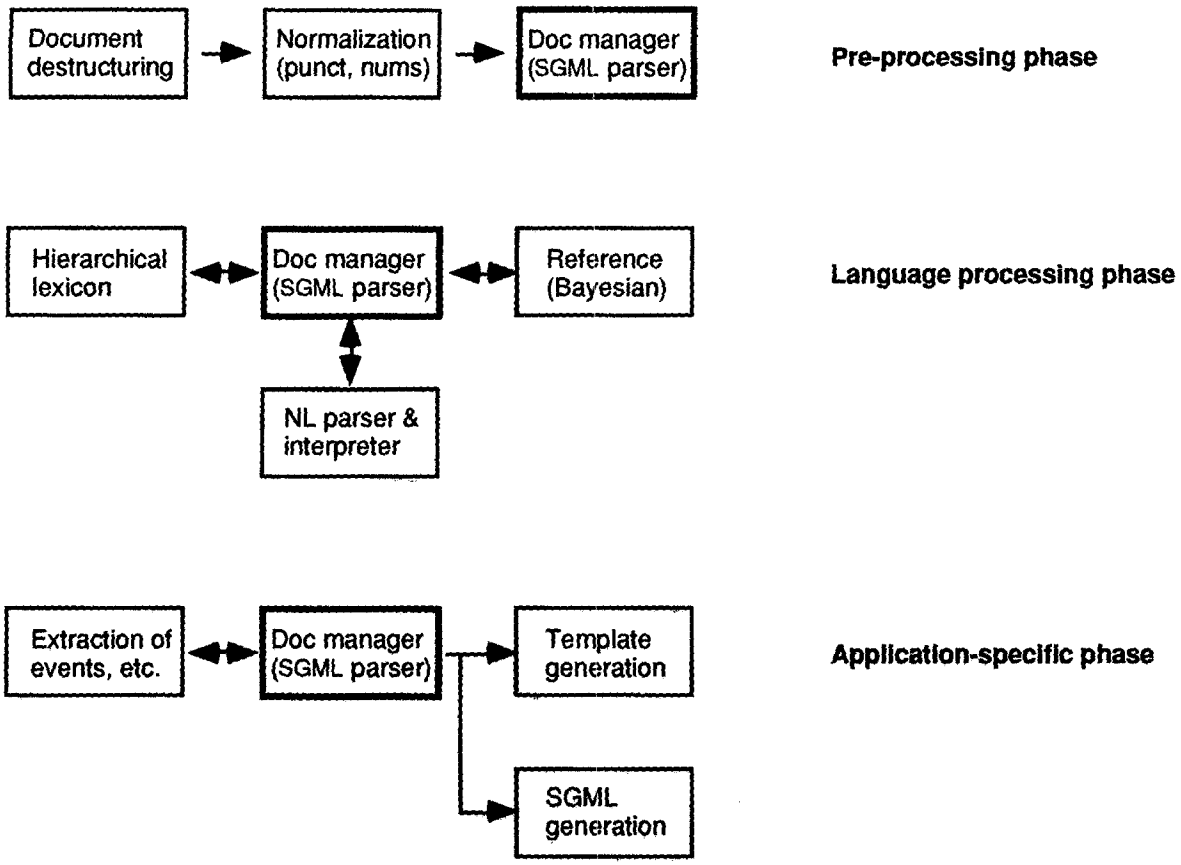

Figure 1: Overall System Architecture

analysis of the document is performed by invoking the natural language analysis modules on the internal document object, and the results of these analyses are stored as attributes of the document. The system's output is normally just another SGML file, in which the content analysis is simply encoded as additional (semantic) markup. For MUC-4, we also provided selective output that consisted solely of filled templates.

As an example of this overall flow of control, and its corresponding encoding in SGML, consider the first paragraph of message TST2-MUC4-0048:

```
SALVADORAN PRESIDENT-ELECT ALFREDO CRISTIANI CONDEMNED THE TERRORIST
KILLING OF ATTORNEY GENERAL ROBERTO GARCIA ALVARADO AND ACCUSED THE
FARABUNDO MARTI NATIONAL LIBERATION FRONT (FMLN) OF THE CRIME.
```

The SGML normalization of this paragraph produced by the pre-processor begins as follows.

```
<p><s>SALVADORAN PRESIDENT<punct loc="midword" type="hyphen">-
</punct>ELECT ALFREDO CRISTIANI CONDEMNED THE TERRORIST KILLING ...
```

The *p* and *s* tags stand respectively for paragraph and sentence delimiters, and the *punct* tag encodes normalized punctuation. In SGML parlance, the text bracketed by the *<punct...>* and *</punct>* delimiters is a punct element, and the equated terms in the *punct* tag are attributes of the overall element. For other details on SGML, see, e.g., [8].

Turning to the natural language phase, the structural markup for sentences, paragraphs, and quotes is exploited straightforwardly to dispatch text chunks to the linguistic parser. More interestingly, punctuation markup can also

appear as part of the actual definitions of lexical items, e.g., the possessive marker (*'s*) or hyphenated words. The lexicon entry for the title modifier *-elect,* for example, is the sequence (**mw-hyphen* elect*), in which **mw-hyphen** matches any SGML *punct* element with *loc* and *type* attributes respectively set to *midword* and *hyphen.*

As mentioned, when the natural language phase has been completed, ALEMBIC records its analysis of the document as further annotation. In the case of the MUC-4 version of the system, this markup simply encodes the templates that the system has produced, e.g.,

```
<p><template>
<slotname>0.  MESSAGE:ID</slotname> <slotval>TST2-MUC4-0048</slotval>
<slotname>1.  MESSAGE: TEMPLATE</slotname> <slotval>1</slotval>
...
</template>
<s>SALVADORAN PRESIDENT <punct loc="midword" type="hyphen>-
</punct>ELECT...
```

INDIVIDUAL PROCESSING MODULES

Pre-processing phase

As we alluded to above, this phase of processing is intended to normalize aspects of a document that are simply too messy to be dealt with during linguistic parsing. In the version of ALEMBIC used in MUC-4, this includes document structure, especially header structure, punctuation, and numeral strings. By handling the document structure in this preliminary phase, we gain all the usual advantages of separating tokenization from parsing, and additionally can introduce special-purpose error recovery strategies. These strategies address such problems as missing quote marks, missing periods at the end of paragraphs, and so forth. One advantage of using SGML is that it actually simplifies implementing these error recovery strategies. SGML allows the preprocessor to ommit issuing many structural tags, in particular some that are keyed directly off of punctuation. The document manager treats the missing markup as implicit, and fills it in from a document grammar instead.

A further motivation for using SGML is that it readily allows us to extend the scope of pre-processing through incremental addition of further modules. Among the modules that we have considered adding to the pre-processor are an SGML-native part-of-speech tagger, and a coarse subject identifier (based on Amsler's FORCE4 algorithm). Both of these have been implemented by our colleagues in MITRE–Washington.

Document Manager

The document manager provides an object-oriented framework for working with SGML documents. The manager is entirely CLOS-based, and SGML elements are thus made available as instances of CLOS objects. A sentence element (corresponding to the string bracketed by matching <*s*> and </*s*> tags) is mapped into an instance of the *S* object, and any *S*-specific code (e.g., the linguistic parser) is thus made applicable to the element.

As mentioned, the document manager is built around a public domain SGML parser/tokenizer written by Goldfarb, the godfather of SGML. The parser consists of C language routines that were made available through the Consortium for Lexical Research. On the Lisp side, there are several ways to use the parser. At the lowest level, one can simply get token types and character strings out of an SGML document. At the highest level, one can get a stream of CLOS objects representing SGML tokens. The parser takes care of canonicalizing a document by, e.g., inserting any tags left implicit by the preprocessor, or filling in the default attribute values of attributes.

Linguistic Lexicon

The design of our lexicon is hierarchical, with specific lexical classes (e.g., auxiliary verbs or mono-transitive verbs) being subsumed in the hierarchy by more abstract lexical classes (e.g., verbs in general). This allows for significant sharing of lexical information between lexical classes, and reduces the complexity of the lexicon.

Lexicon entries correspond to base stems, which are automatically expanded into the lexical forms that are actually used by the system. Our syntactic framework closely follows Steedman's combinatory categorial grammars (CCG's), and as a result the expansion of a stem occurs along two dimensions.

- Lexical word forms, that is, the surface forms of the stem. For count nouns this is just the singular and plural form of the noun; for main verbs, this includes the usual participial forms, tensed forms, and infinitive, as well as adjectival forms, and various nominalizations.
- Lexical category types, that is, the underlying syntactic categories that are sanctioned by a given word form. In the case of a mono-transitive verb's past tense/participle (e.g., "condemned," the first verb in TST2-MUC4-0048), this includes the active voice (e.g., "Cristiani. condemned the terrorist killing"), the passive voice, and ancillary forms such as the detransitivized active voice and verbal adjectives.

In our variant of CCG's, lexical categories are treated either as complex vectors of features, or as mathematical functions over such vectors. For example, stripping away all syntactic and semantic features, the syntactic category corresponding to a transitive verb is the function *S\NP/NP*, i.e., a function that will produce an *S* given an object *NP* on its right (the */NP* term) and a subject *NP* on its left (the *\NP* term). To accomodate syntactic and semantic features, categories are actually implemented in a standard unification framework (as in [11]). Some features can be themselves category-valued, and so the overall category structure is actually a reentrant graph that can become fairly involved, as attested to by a partial expansion of "condemned" in the active voice:

```
[[res [[res [[syn :S]
             [sem [[head :CONDEMN
                    [args #( SUBJ-SEM OBJ-SEM ) ]]]
             [tense :PAST]
             [synform :FINITE]
             [by-pp-obj NIL]
             [for-to-pp-obj NIL]]]
       [dir :\]
       [arg [[syn :NP]
             [sem SUBJ-SEM]
             [num-pers *any*]]]]
 [dir :/]
 [arg [[syn :NP]
       [sem OBJ-SEM]]]]
```

This encoding is based on Pareschi and Steedman's category notation [6], wherein the *res*, *arg*, and *dir* features are used to encode a syntactic function's result, argument, and direction of application. To reduce the complexity of defining tens of thousands of such lexical entries, we associate to each category type (such as the active voice of a transitive verb) a lexical skeleton, that is, a partial lexical entry that is fleshed out during the expansion of a stem. The fleshing out of skeletons does not actually occur until run time, when a word form is actually found in a document. Since category data structures are fairly substantial, this yields tremendous memory savings.

The lexicon was populated in part by adapting knowledge bases provided to us by Richard Tong of ADS.

Geographical Names, Personal Names, and Unknown Words

For MUC-4, we used a number of strategies for handling open classes of proper names. For geographical names, we relied primarily on a listing of such names that had been compiled by previous MUC participants, and which was forwarded to us by ADS. As a back up, we also encoded a small grammar of Spanish geographical names—for example, "san" has a reading as a functor that produces geographical names given a personal name on its right.

For personal names, we relied primarily on a cross-cultural database of 15,000 names obtained from various public domain databases. Most of these are first names, with only about 2,000 family names covered by the database. In order to fill inevitable gaps in the database, we allowed unknown words to be given, among others, a definition as a personal proper name. Separately, we provided a grammatical type-shifting rule that turns personal

names into functors that combine with any personal name on their right. In non-CCG terms, this amounts to a grammar rule of form:

```
PERS-NAME -> PERS-NAME PERS-NAME
```

All the names in TST2-MUC4-0048 turned out to be in our database, in part because we had already extended it with a list of VIP names provided by ADS.

CCG Parser

We chose to use categorial grammars in ALEMBIC for a number of reasons. First and foremost, we expected our syntactic coverage to be necessarily incomplete, and knew that we would have to rely heavily on bottom-up parsing. In this light, categorial grammars are effectively the *uhr*-bottom-up linguistic framework, as one cannot meaningfully speak of top-down parsing in this framework. We also wanted a framework that was strongly lexically governed, as in CCG's, in order to reduce the parsing search space. Finally, in anticipation of eventually wanting to provide fairly comprehensive coverage of syntax, we chose one of the recent mildly context sensitive frameworks, in the hope that we could exploit the linguistic strengths of the framework at some future point.

Our current CCG parser is based upon Pareschi and Steedman's algorithm for left-to-right parsing of combinatorial categorial grammars [6]. Their approach is nominally only intended to produce full chart-based parses. Because we anticipated our syntactic coverage to be incomplete, we extended the basic algorithm into a heuristic semi-parser. The semi-parser heuristics are used to provide a spanning segmentation of the input string.

In addition, we extended the CCG framework per se in order to cover optional and displaced arguments, which are typically weaknesses of traditional categorial grammar frameworks. The approach we've taken involves introducing a pair of features for each optional argument, one feature to encode the type of argument that is expected, and the second to encode the disposition of the argument's semantics. For instance, consider the passive voice of a transitive verb—*kill* is a canonical example. A partial encoding would be as follows:

```
[[res [[syn :S]
       [sem [[head :KILL
              [args #( AGENT-SEM OBJ-SEM ) ]]]
       [by-pp-prep :BY]
       [by-pp-obj AGENT-SEM]
 [dir :\]
 [arg [[syn :NP]
       [sem OBJ-SEM]]]]
```

The *by-pp-prep* feature indicates that the category can be modified by a PP headed by *by;* the *by-pp-obj* feature indicates that the embedded semantics of the PP is then indirectly unified with the semantic agent of the sentence.

At the time we fielded our system for MUC-4, both our syntactic coverage and semi-parsing heuristics were still very preliminary, and our overall parses were thus extremely fragmentary. For example, the first sentence in TST2-MUC4-0048 ended up being bracketed roughly as follows:

```
[SALVADORAN PRESIDENT] [-] [ELECT] [[ALFREDO CRISTIANI] CONDEMNED
[THE TERRORIST KILLING OF ATTORNEY]] [GENERAL ROBERTO GARCIA
ALVARADO] [AND] [ACCUSED [THE FARABUNDO MARTI NATIONAL LIBERATION
FRONT]] [(] [FMLN] [)] [OF] [THE CRIME] [.]
```

This particular bracketing illustrates several early shortcomings of our grammar, many of which have been addressed in the months since the MUC-4 evaluation. First, the MUC-4 version of the title sub-grammar was weak: title modifiers such as *-elect* or *general* were simply absent from the grammar. Second, prepositional phrase coverage was incomplete: PP's that appeared as optional arguments of categories would parse, but those that should be treated as modifiers failed to do so. In addition, many verbs simply lacked the appropriate subcategorization frame for PP arguments, as in this case with *accused*. Finally, as with many semi-parsers, ALEMBIC currently punts on coordination.

Semantic Interpretations

As might be gleaned from the category definitions given above, ALEMBIC produces semantic interpretations concurrently with parsing. The meaning representation language that we use is directly descended from our earlier work on the King Kong interface [2], whose underlying approach is similar to that in the core language engine [1]. Meaning representations are given at the so-called *interpretation level*, where quantifiers are not scoped with respect to each other, but are simply left "in place," i.e., attached to their associated noun phrases. For example, the interpretation of the fragment "the terrorist killing" in message TST2-MUC4-0048 is:

```
[[head :KILL]
 [args #( [[head :TERRORISM-AGENT]
           [quant NIL]]
          OBJ-VAR )]
 [quant :DEFINITE]]
```

In addition, the representation maintains an implicitly Davidsonian representation of events and other relations. That is, aside from their underlying arguments, the relations may be modified through a proxy variable, as in the following encoding of a later sentence in the message, "guerillas attacked ... five days ago."

```
[[head :ATTACK]
 [args #([[head :TERRORISM-AGENT]] … )]
 [proxy PROX179]
 [mods {[[head :TIME-OF]
         [args #(PROX179
                 [[head :DATE]
                  [proxy PROX180]
                  [mods {[head :BEFORE]
                         [args #(PROX180 *NOW* [[head :DAY]
                                                [quant 5]])]}]]}]]}]]
```

Reference Resolution

The approach we have taken towards reference resolution [3] attempts to integrate several disparate approaches towards the problem. The literature on reference resolution identifies a number of sources of linguistic evidence that can be applied towards resolving anaphoric references, but few attempts have been made at combining these evidence sources in a principled way (for an exception, see [5]). The approach embodied in our system attempts to perform the integration by exploiting a Bayesian belief network.

The network combines a number of evidence sources that bear upon whether an anaphor (either a definite reference or a pronoun) can be resolved to a particular candidate referent. Because of the fragmentary nature of our parses, the reference resolution network only considered non-grammatical features of the anaphor and candidate. In particular, these included:

- Agreement on number, person, and gender
- Compatibility vis-à-vis the semantic hierarchies
- Recency
- Reflexivity
- Phrase type (pronominal, definite, or otherwise)

We experimented with a number of such networks prior to the MUC-4 evaluation run, including hand-built networks and networks derived by machine learning algorithms. We ended up selecting a simple flat network in which all evidence sources were directly allowed to bear upon the root node (which stood for coreference of the anaphor and candidate).

To apply the network, our system first collects a set of *mentions* from the parsed document: these amount roughly to noun phrases and to event verbs with any arguments that might have been attached by the parser. Anaphoric mentions are then compared to mentions preceding them in the document. The comparison is performed by populating the evidence nodes of the network according to the characteristics of the anaphor and candidate. Mentions that are found to co-refer are grouped together and assigned to a unique discourse entity (called a *peg*).

In the case of our actual run on TST2-MUC4-0048, for example, the bracketing of the first sentence produced by the semi-parser lead to identifying as mentions (among others) Alfredo Cristiani, and the murder event introduced by the nominalization of "kill." The second of these phrases was then taken as potentially anaphoric and compared to earlier mentions in the sentence, including that for Cristiani. In the case, of Cristiani the mentions were found not to co-refer, reflecting the importance of KR compatibility. Nevertheless, the fragmentary nature of the parses, coupled with the relative lack of grammatical features in the Bayesian network, led to disapointing reference resolution performance overall. As we describe below and elsewhere in these proceedings, this led indirectly to our relatively low precision scores.

Extraction of Significant Events and Template Generation

Once reference resolution has been performed, the system enters a MUC-4-specific processing phase. The first step towards producing templates consists of identifying significant violent events, which is performed by searching the document for event pegs whose semantic heads are subsumed by the KR node for violence. In our actual run on TST2-MUC4-0048, two such pegs were found in the first paragraph: one for "killing" and one for "the crime." The fact that two separate pegs were found for these phrases reflects a failure on the part of our reference resolution mechanism, as these two phrases should properly have been determined to be co-referential.

In the MUC-4 version of ALEMBIC, the actual generation of templates is keyed off of the pegs for violent events. Each such peg is taken to indicate a separate act of violence for which a template should be generated. This strategy is very straightforward, but it places a tremendous burden on the system's ability to identify coreferential events. As reference resolution is actually a weak point in the MUC-4 version of the system, this leads us to generate multiple templates for what is really the same event. As a result, we paid a significant toll in our precision scores.

Turning to the slot-filling mechanism for a particular template, once a significant event peg has been identified, ALEMBIC then attempts to locate the participants in the event. Preferentially, the system attempts to use the syntactic arguments of the event phrase, if the parser succeeded in identifying them. In those cases where the parser failed to provide arguments to a verb or a nominalized event, the system assumes that the parse must have been fragmented, and attempts to locate potential arguments elsewhere in the sentence. This search is clearly heuristic and application-specific. In the case of the actual perpetrator of the event, the system attempts to find phrases with the appropriate agentive heads—this includes military organizations, terrorist organizations, and known terrorists. A similar process is performed to identify entities that might bear on other template slots. For example, targets and instruments are identified by searching for phrases headed by KR relations denoting damage, injury, or weapons.

The heuristic nature of this process yields mixed results. At times it reunites arguments that had been separated from their verbs due to fragmentary parsing, and at times it simply results in unprincipled filling of slots (with results that can be incorrect, and even humorous.)

The final step taken by the system towards analyzing a message is also the most uninteresting. Once the message has been fully analyzed, it is simply dumped back out to a file, along with any relevant markup of meaning analysis. In the case of the MUC-4 task, this amounts to associating SGML template tags to relevant paragraphs of text, or more simply, to ignoring the production of an SGML output file, and just printing the templates on their own.

CONCLUDING THOUGHTS

As we mentioned at the beginning of this note, ALEMBIC is still in a very early stage of development. Although we are satisfied with the system's (fairly humble) performance given its relative youth, we are also painfully aware of the compromises performed in fielding a message processing system on such a tight development schedule. Many of the shortcuts we took are unsatisfyingly heuristic. In addition, many interesting ideas that seemed promising in paper studies were never included in the fielded system. It was our original intent, for example, to use

a completely different parsing algorithm that supports nearly-semantic parse rules; this class of rules is related to the nearly-syntactic extraction rules of some recent MUC-class systems, e.g., FASTUS and CIRCUS. We had also intended to extend the semantic interpretation process with a terminological inference mechanism based on [9].

These modules were not implemented in time for MUC-4—nor were a host of other improvements detailed in our companion results and analysis paper. It is to these ideas that we now turn, in the expectation that the next version of ALEMBIC that we apply to the MUC data extraction task will dramatically outperform the version presented here.

Acknowledgments

Principal funding for this work was provided by the MITRE Corporation. Funding for our participation in MUC-4 was provided by the Defense Advanced Research Projects Agency through the MUC conference committee. We would also like to express our gratitude to Richard Tong and his colleagues at ADS for providing us with their helpful lexicon and taxonomies. Finally, we would like to thank Beth Sundheim for her ongoing support.

REFERENCES

[1] Alshawi, H and van Eijck, J, "Logical forms in the core language engine," in *Proceedings of the 27th Annual Meeting of the ACL*, Vancouver, British Columbia. 1989.

[2] Bayer, S, and Vilain, M, "The relation-based knowledge representation of King Kong," *SIGART Bulletin* **2**(3), 15-21. 1991.

[3] Burger, J and Connolly, D, "Probabilistic resolution of anaphoric reference," To appear in *Proceedings of the 1992 AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, Boston, MA. 1992.

[4] Krupka, G, Jacobs, P, Rau, L, and Iwanska, L, "GE: Description of the NLTOOLSET system as used in MUC-3," in [7]. 1991.

[5] Luperfoy, S, *Discourse Pegs: A Computational Analysis of Context-dependent Referring Expressions*, doctoral dissertation, Dept. of Linguistics, University of Texas at Austin. 1991.

[6] Pareschi R and Steedman, M, "A lazy way to chart-parse with categorial grammars," in *Proceedings of the 25th Annual Meeting of the ACL*, Stanford, CA. 1987.

[7] Sudheim, B, *ed*, *Proceedings of the Third Message Understanding Conference*, Morgan Kaufman. 1991.

[8] van Herjwinen, E *Practical SGML*, Kluwer Academic Publishers. 1990.

[9] Vilain, M, "Deduction as parsing: tractable classification in the KL-ONE framework," in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI91)*, Anaheim, CA. 1991.

[10] Weischedel, R, Ayuso, D, Boisen, S, Ingria, R, and Palmucci J, "BBN: Description of the PLUM system as used in MUC-3," in [7]. 1991.