UNISYS:

Description of the CBAS System Used for MUC-5

Carl Weir and Rich Fritzson
Unisys Corporation
70 East Swedesford Road
Paoli, PA 19301

INTRODUCTION

This paper describes CBAS, a data extraction system with rule-based reasoning modules.[1] The CBAS architecture depicted in Figure 1 emphasizes the use of multiple processors to detect significant primitive facts which are then processed by reasoning modules implemented as collections of forward-chaining rules to infer additional information. A guiding principle behind the architecture is to rely as much as possible on initial processors with relatively simple internal structure in order to insure greater robustness. However, the model does allow for the use of sophisticated initial processors, processors which embody linguistic analysis techniques. This emphasis on collections of multiple preprocessors which provide sets of primitive facts to be reasoned about is reminiscent of the standard architecture proposed for multisensor data fusion systems (where preprocessor = sensor) [5].

APPROACH AND SYSTEM DESCRIPTION

The data extraction process performed by CBAS takes place in three processing phases. An initial *tokenization* phase generates a set of primitive facts. A second, *intensional reasoning* phase involves the use of forward-chaining rules to infer information about possible events and their component objects and attributes from the basic facts generated in the initial phase. A third and final phase involves *extensional reasoning* activities in which actual events and their component objects and attributes are inferred from the set of possible entities introduced during the intensional reasoning phase.

Tokenization

The initial, *tokenization* phase consists of a collection of processors, each of which contributes what it can to the set of primitive facts which form the basis for higher-level reasoning. In the MUC-5 version of CBAS, three different tokenization processors were used, all of which were integrated together using PERL, a programming language specifically designed for manipulating textual data.

The most basic of the three processors in the MUC-5 implementation is used to do *text zoning*, which is the detection of regions of text corresponding to words, sentences, paragraphs, punctuation, and other regions which frequently arise in newswire text, such as date, source, and title headers, and remarks about the location of graphic images. A text zoning processor must be able to recognize the types of documents it is processing in order to properly identify regions of text, since the conventions and/or reliable clues for delimiting zones vary across document types.

The two other tokenization processors in the MUC-5 implementation require the output of the text zoning processor to perform their tasks; however they may do their processing asynchronously with respect to one another. The first of these processors determines the part-of-speech of word tokens. Currently a tagger developed by Eric Brill is being used.[2] The second of the processors searchs the word tokens which have been delimited for combinations which possibly correspond to company names.

[1] CBAS (pronounced "Sea Bass") is an acronym for *Concept-Based Analysis System*. For additional information on the system, including its availability, contact Carl Weir, 215-648-2369, weir@vfl.paramax.com.

[2] This tagger, which is implemented in C, is available from the *Linguistic Data Consortium*.

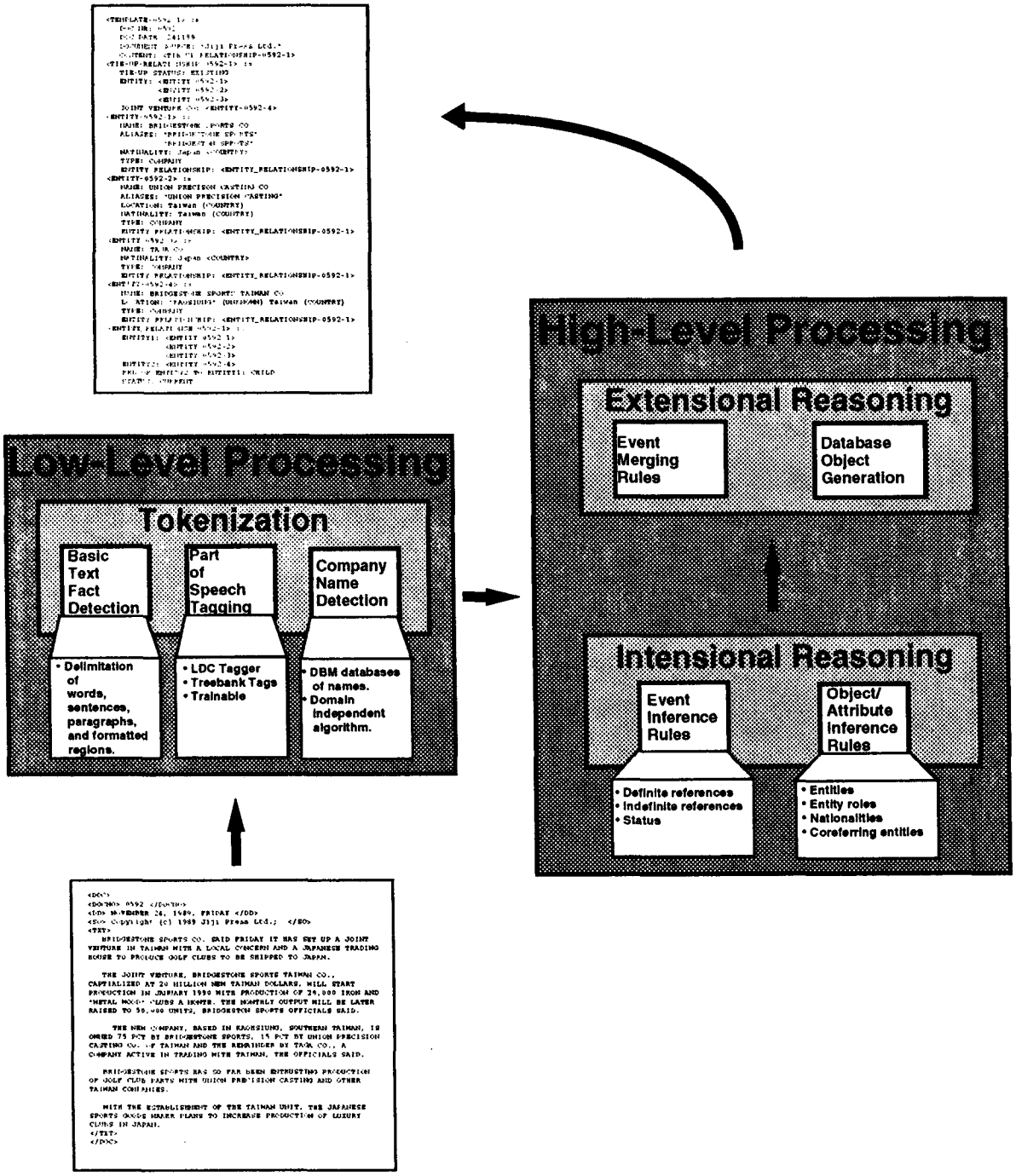

Figure 1: The CBAS Architecture.

Two types of problems were encountered in using the part-of-speech tagger for the MUC-5 task. First, in some cases the tagger did not make sufficiently fine-grained distinctions. A good example of this type of case is the lack of a class distinction between the definite article *the* and the indefinite article *a*, both of which are assigned the tag DT. And second, in cases where the accuracy of a given tag was crucial, often the tagger was not accurate enough. This latter type of problem has arisen in rules which depend on the identification of possessive "s" tokens. In general, part-of-speech tagging did not play as significant a role as it was anticipated to, and given that it consumed 25% of the time required to process a message, was more trouble than it was worth in the MUC-5 task.

The company name parser used in CBAS was a major success. The parser, which is implemented in C, is fast, taking on average about 4 seconds per text to do its job. The parser incorporates three procedures for detecting company names. First, it searches for known company names, looking for matches of token sequences against a company name database in Unix DBM format. The matches are not required to be exact; for example, trailing designators don't need to match—all three of the following sequences would be matched against the "Ford Motor" entry in the DBM database:

- Ford Motor Co.
- Ford Motor Inc.
- Ford Motor Ltd.

When looking for matches, lowercase names are not recognized, and names preceded by the preposition "in" are not recognized, since so many company names are also the names of places. DBM databases are capable of containing large quantities of data—as many as a billion blocks. (Currently the company name database contains about 8 MB of entries.) Moreover, DBM databases can be accessed very quickly, making them especially attractive in a data extraction task.[3]

In addition to the search for word token sequences corresponding to known company names, the company name parser also searches for sequences of capitalized words. This procedure does not attempt to detect sequences which start at the beginning of a sentence, or to detect sequences in "all caps" text. Also, sequences of tokens which correspond to place names or months are not recognized as possible company names.

A third and final procedure used by the company name parser is to look for sequences of tokens which end in company designators. The basic strategy here is to first locate a company designator and then work backwards until the sequence meets one or more delimiting criteria, including the presence of a sentence boundary, a punctuation marker, a preposition, another company designator, or something in lower case (other than "and").

The CBAS company name parser is a good example of the sort of processor which one wants to develop in a data extraction system: the procedures it embodies are simple; the facts it extracts have a consistent level of reliability; it relies minimally on other processors (just the text zoner) to perform its task; it performs its task quickly; and finally, there are many domains for which the detection of company names is required, and so it will be a useful preprocessor in many applications.

During the early stages of developing the MUC-5 version of CBAS, an effort was made to incorporate an NLP parser as yet another sensor invoked during tokenization. Tomek Strzalkowski's *Tagged Text Parser*

[3]Processing speed is not directly factored into the scores assigned to systems in MUC evaluations. However, it is a critical issue in performing well on the evaluations, since a rapid rule development cycle is needed for development purposes - a failure to consider the need for a rapid rule development cycle is one of the more common errors among less experienced participants in such efforts. Government sponsors have also begun to realize that a data extraction system which can process, say, 100 messages in 15 minutes is useful as an interactive analysis tool, which is a very desirable attribute. A few extraction systems are capable of this level of performance—systems relying heavily on linguistic analysis techniques take much longer, in the neighborhood 8-10 hours. Typical extraction systems which do not rely heavily on linguistic analysis techniques require 2-5 hours (1-3 minutes per text) to process 100 messages, depending on the texts being processed. However, no existing data extraction system is truly interactive in the sense that extraction queries can be formulated "on the fly"; all implementations of existing extraction architectures are custom-built to answer a single query.

(TTP) was acquired for this purpose [4]. However, after the parser was integrated into the system, it was determined that the structures returned by the parser did not preserve enough information about the regions of actual text corresponding to recognized syntactic structures to be useful, and that to modify the parser to return suitable output structures would not be possible, given the staffing resources available for the MUC-5 effort.[4] Aside from the fundamental problem with output structures inappropriate for the MUC-5 task, the TTP parser, despite its speed compared to other parsers examined, was nevertheless more than doubling the amount of time required to process a text. Consequently, the effort to incorporate an NLP parser into CBAS for the MUC-5 evaluation was abandoned.[5]

Unlike the situation in canonical NLP systems, the tokenization phase in the CBAS architecture involves a great deal of processing. Indeed, in CBAS any processors incorporating linguistic analysis techniques are viewed as components of the tokenization phase. What counts as a "primitive fact" versus a "derived fact" is a fairly arbitrary decision, and similarly what counts as a tokenization phase component versus a component of some higher-level processing phase is also arbitrary. However, the direction which is being taken in CBAS —pushing more and more analysis "up front" in the form of multiple, specialized, relatively asynchronous processors—is one which other research groups are also finding to be advantageous.[6] We believe there is a trend underway in which NLP systems applied to information extraction tasks are beginning to look more and more like standard multisensor data fusion engines.

Intensional Reasoning

After the tokenization phase has generated a collection of primitive facts, "higher-level" processing phases of the CBAS architecture are invoked to derive additional information. Two such phases exist in the current implementation of CBAS, and the first of these involves *intensional* reasoning, so-named because the general idea at this stage is to detect *possible* events being referred to, along with their component objects and attributes, without firmly committing to their existence.

Both of the higher-level processing phases are realized as collections of forward-chaining rules. The decision to use forward-chaining as the default reasoning method was motivated by an overall desire in CBAS to maintain as asynchronous a reasoning process as possible, imposing control only when necessary. CLIPS, a popular forward-chaining system, was used to implement the higher-level phases.[7] It is easy in CLIPS to incorporate calls to external programs via C procedures, and this capability makes it possible to escape from the default forward-chaining reasoning method whenever it is desirable to engage in a different style of analysis. In CBAS, calls are made within CLIPS images to external UNIX DBM databases, which are used to store static knowledge (just like the company name parser stores relatively static knowledge about known companies). This use of DBM databases greatly reduces the size of internal CLIPS factbases without a penalty in access time.

A number of other MUC-5 systems have architectures similar to that of CBAS in that pattern-matching plays a key role in their reasoning phases.[8] However, CBAS is distinguished from these systems in that the pattern-matching process in CBAS is implemented using general-purpose expert system software whereas the other systems rely on custom-built code, and in most cases the custom-built code involves the use of

[4] In MUC evaluation tasks, there is a need to supply the actual text substrings corresponding to an analysis structure when instantiating output data structures (templates), and it has been our experience that the representations generated by some linguistic analysis components (of which TTP is just one example) do not provide a straightforward means of satisfying this requirement.

[5] Independent of speed and the accessibility of data in the output structures generated by linguistic analysis components, another problem which may be lurking about is a highly inconsistent level of reliability: it could be that the accuracy of results are so unpredictable, that incorporating linguistic analysis results in the contexts of intensional and extensional reasoning is too much of a rule-writing burden to be manageable.

[6] Lisa Rau (GE) has expressed this view in discussions.

[7] CLIPS is a "GOTS" product developed and maintained at NASA's Johnson Space Center. Rule-based systems similar to CLIPS have been used before to implement data extraction systems; two well-known implementations of this sort are the Carnegie Group's *Text Categorization Shell* [3] and the ADS *Rubric* system, which is a subcomponent of the Codex system evaluated at MUC-3 [1].

[8] A distinction is being made here between *pattern-matching* and various forms of *NLP-based syntactic analysis*, including systems which don't make a strong attempt to derive full sentential parses.

a formalism which is less familiar to ordinary users than standard production rules.

A fundamental feature of the forward-chaining rules used in CBAS is that the facts which the rules infer are associated with specific regions of text in very much the same way that edges in a parser's well-formed substring table are assigned to specific regions of an input string. However, unlike typical parsers, which contain an implicit constraint that adjacent constituents in a rule must be realized by contiguous strings of text in the input, all constraints in CBAS inference rules are explicitly encoded via attributes of facts—contiguity is not assumed.

A brief digression is needed at this point to provide a basic understanding of the structure of a CLIPS forward-chaining rule. First, any forward-chaining system, CLIPS included, has two basic data types: facts and rules. Facts represent what is already known, and rules describe how to infer new facts, given whatever facts currently exist. Forward-chaining rules have a "left-hand side" (LHS) and a "right-hand side" (RHS), which are delimited from each other by an arrow symbol, ==>. The LHS of a rule consists primarily of patterns that facts in the factbase might satisfy, and the RHS of a rule consists of actions to be performed if all the expressions constituting the LHS of the rule do match existing facts, and of course a common action performed on the RHS of a rule is to assert new facts and/or to remove existing facts which match the patterns on the rule's LHS. Pattern-matching never occurs on the RHS of a rule, only actions. In CLIPS, rules are defined using a **defrule** construct, which is fairly transparent in format.

It is easier to grasp the nature of a forward-chaining rule by looking at concrete examples. The following CBAS rule used in the intensional reasoning phase states that if a company name has been predicted by the company name parser, and if this company name consists of one word token whose part-of-speech category is **PP$**, **VB**, **RB**, **IN**, or **CC**, then the predicted company name is not really describing a company object and should be eliminated from consideration.[9]

```
(defrule delete-company-name-with-wrong-cat
  (declare (salience 400))
  (control-fact (phase corp))
  ?A <- (company-name (l ?v1)(r ?v2))
  (txt_token (l ?v1)(r ?v2)(cat "PP$"|"VB"|"RB"|"IN"|"CC"))
  ==>
  (retract ?A))
```

Note in this example how the "l" (left) and "r" (right) attributes, whose values are pointers to locations in the text, are used to capture the fact that the company-name "concept" and the word token span the same region of text. Typically in forward-chaining formalisms an expression beginning with a question mark is a variable to be instantiated by a value in an actual fact in the factbase. Note that for the "cat" attribute, alternative literal string values are provided—a given actual fact would need to have a value for its "cat" attribute which corresponds to one of the literal strings. The CLIPS facts used in CBAS are defined to be "template" structures, which means that the order in which attributes are specified is irrelevant, and templates will match a pattern on the LHS of a rule even if the template has attributes not specified in the pattern—the only requirement is that attributes explicitly mentioned in the pattern match the template.[10] Finally, the **?A <-** notation is used to provide a way of pointing to the fact instantiating a given pattern on the LHS so that on the RHS the fact can be modified or deleted.

In the following rule, the l(eft) and r(ight) attributes of **txt_token** facts are used to require two word tokens to be contiguous. This rule illustrates a rudimentary form of syntactic analysis in which words in domain-specific classes are combined to infer constituent structures. Constraining the tokens to specific word classes is done by unifying the **reg** attributes of **txt_token** and **word** facts, where **word** facts encode the class information. In this particular case, the only words of type "joint" are *joint*, *co-operative*, and *new*, and the only words of type "venture" are *venture*, *project*, *plan*, *deal*, *firm*, *concern*, and *development*. And the set of possible phrases recognized by this rule is the Cartesian product of these two word classes.

[9] TreeBank part-of-speech labels are assigned by the tagger used in CBAS.

[10] Do not confuse the use of the term "template structure" in CLIPS with the use of the same term in MUC applications—in the latter case, it refers to output structures which are intended to represent generalized data base records.

```
(defrule const-joint-venture
  (control-fact (phase const))
  (txt_token (p ?p)(s ?s)(l ?v1)(r ?v2)(cite ?t1)(reg ?r1))
  (word (type "joint")(reg ?r1))
  (txt_token (p ?p)(s ?s)(l ?v2)(r ?v3)(cite ?t2)(reg ?r2))
  (word (type "venture")(reg ?r2))
  ==>
  (bind ?cid (gensym*))
  (bind ?new-r (format nil "%s %s" ?r1 ?r2))
  (bind ?new-c (format nil "%s %s" ?t1 ?t2))
  (assert (const (cid ?cid)(type "venture")(p ?p)(s ?s)(l ?v1)(r ?v3)(reg ?new-r)(cite ?new-c))))
```

Surely the above rule represents the sort of formalism that gives linguists nightmares—subconstituents are domain-specific, not embodying any linguistic generalizations.[11] Nevertheless, such rules are much simpler to compose and maintain, despite their superficially complex appearance, than standard collections of grammar rules for large-scale systems. Moreover, they are are much more robust—grammar rules are so interdependent that robustness is a chronic problem—and they are much faster to execute, simply because they do not constitute an effort to reach a complete constituent analysis.

In the following forward-chaining rule a distinction is made between definite and indefinite references to joint ventures. In this case, explicit strings corresponding to definite and indefinite articles must be accessed, since no part-of-speech distinction is available between definite and indefinite determiners. In the MUC-5 version of CBAS only non-definite references to joint ventures permit the inference of a joint venture reference.

```
(defrule et_16_DT_VENTURE_WITH
  (control-fact (phase et))
  (txt_token (p ?p)(s ?s)(l ?v1)(r ?v2)(cat "DT")(reg ?r1)(cite ?t1))
  (txt_token (s ?s)(l ?v3&:(> ?v3 ?v2))(r ?v4)(reg "venture")(cite ?t2))
  (not (txt_token (l ?v5&:(>= ?v5 ?v2))(r ?v6&:(<= ?v6 ?v3))(cat ?c&:(neq ?c "JJ")&:(neq ?c "CD"))))
  ==>
  (bind ?new-id (gensym*))
  (bind ?new-r (get-region-string ?*REG-DBM* ?v1 ?v4))
  (bind ?new-c (get-region-string ?*CITE-DBM* ?v1 ?v4))
  (if (eq ?r1 "the") then
    (assert (eref (id ?new-id)(rid 16)(p ?p)(s ?s)(l ?v1)(r ?v4)(reg ?new-r)(cite ?new-c)))
    else
    (assert (etrigger (id ?new-id)(rid 16)(p ?p)(s ?s)(l ?v1)(r ?v4)(reg ?new-r)(cite ?new-c)))))
```

In the above rule, the word represented by the first **txt_token** fact is not required to be contiguous with the word represented by the second **txt_token** fact. However, the first word is required to be to the left of the second word. The negated pattern ensures that any words occurring between the first and second words must be adjectives or numeric expressions—ie, modifying expressions. The **&:** notation introduces "in-line" functional contraints on variables in patterns. It should be possible to hide a great deal of the explicit encoding of constraints on location pointers by introducing a slightly higher-level formalism which expands to the explicit notation currently being used. The primary reason this has not already been done is that while encoding the constraints may look complicated, it is actually a fairly straightforward task, and taking the time out to develop the higher-level formalism has not been justifiable.

A significant feature of the above rule is the use on the right-hand side of the function **get-region-string**. This function invokes a remote C procedure which accesses DBM databases. In this rule, the procedure is used to access regions of text both in their citation forms and in a regularized form (all lowercase). The ability to compute arbitrary regions of text in this fashion greatly simplifies the writing of CBAS forward-chaining rules, since it bypasses the need to do explicit pattern-matches on the left-hand side

[11] To be fair, it has been our experience that "industrial-strength" grammars tend to be very domain-specific as well, requiring a high overhead for rule maintenance.

of the rule to determine the strings corresponding to word tokens, a particularly problematic situation, given that in this particular case, the distance between the determiner and the "venture" constituent is arbitrary. This is a good example of when bypassing a default reasoning method is desirable.

Following standard practice in forward-chaining system development, the antecedent portions of CBAS forward-chaining rules include references to "control fact" statements (see the above rules for examples). These control facts are asserted and retracted during the processing of a text to enable or disable portions of the Rete network constructed out of the system's factbase.[12] The use of control facts is dependent on the ability to set the salience of a given forward chaining rule. The salience of a rule determines its position on the agenda CLIPS maintains of all rules whose left-hand side patterns have been satisfied. Below, for example, is a rule which retracts a control fact of the form `(control-fact (phase const))` and asserts a fact of the form `(control-fact (phase et))`. All rules whose LHS contains the pattern `(control-fact (phase const))` and which have a higher salience value than -500 will be activated before this rule has a chance to retract the fact, after which those rules will no longer be able to fire.[13] Each rule retracting a control fact generally asserts a new control fact in order to activate another portion of the Rete network.

```
(defrule const-phase-end
  (declare (salience -500))
  ?f <- (control-fact (phase const))
  =>
  (retract ?f)
  (assert (control-fact (phase et))))
```

The rules which are associated with a given portion of a Rete network which is activated or deactivated by a given control fact constitute a *rule module*. Three different types of rule modules arise in the intensional reasoning phase:

- Modules which consist of rules for locating possible references to events. There is only one module of this sort in the MUC-5 implementation of CBAS, since only one type of event is of interest, but multiple modules of this sort could exist. (In the MUC-4 terrorist domain, for example, different types of terrorist acts needed to be distinguished.)
- Modules which consist of rules for inferring facts describing possible objects and attributes of events. For example, a rule module exists which "promotes" predicted company names to the status of being denotations of company entities.
- Modules which consist of rules for associating possible objects and attributes of events with specific possible events. For example, modules exist for determining the roles played by objects associated with a given possible event.

During the intensional reasoning phase, data correlation is done across objects, but not across events. In the MUC-5 joint venture domain, this activity primarily involves reference resolution among company entities. The rules used to perform this task in CBAS are fairly primitive; the following rule does most of the work by insuring that if two company entities exist and one has a "reg" value which is a substring of the other, then the "cite" and "reg" attribute values of the entity with the shorter reg value are made the same as the longer cite and reg values. It also insures that both entities have the short cite value as an "alias", which is a requirement in the MUC-5 task.

[12] A Rete network is a data structure commonly used to encode information in forward-chaining systems. See Forgy [2] for an explanation of Rete networks.

[13] Activation of each rule will, of course, also depend on all other LHS patterns matching facts in the factbase as well.

```
(defrule assert-entity-aliases
  (control-fact (phase corp))
  ?A <- (entity (id ?i1)(l ?v1)(reg ?r1)(cite ?t1)(type "COMPANY"))
  ?B <- (entity (id ?i2&:(neq ?i1 ?i2))(l ?v2)(reg ?r2)(cite ?t2)(type "COMPANY"))
  (test (and (str-index ?r1 ?r2)(neq ?r1 ?r2)))
  (txt_token (l ?v1)(reg ?r3))
  (txt_token (l ?v2)(reg ?r3))
  ==>
  (modify ?A (reg ?r2)(cite ?t2)(aliases ?t1))
  (modify ?B (aliases ?t1)))
```

Determining coreference relations is a critical issue in data extraction technology. Unfortunately, the majority of work done by linguists in this area involves pronominal correference, whereas in the data extraction tasks which have been examined in MUC conferences, coreference among common noun descriptions is a more significant issue.[14]

The Extensional Reasoning Phase

The second "higher-level" processing phase in CBAS is called the *extensional* reasoning phase. The general purpose of this phase is to take the information about possible events and their component objects and attributes contributed by the intensional reasoning phase and to identify on the basis of this information a collection of actual event instances to be represented as database objects. In practice, rules in the intensional reasoning component have been responsible for data correlation at the object level, and rules in the extensional reasoning component have been responsible for data correlation at the event level.

For the MUC-5 version of CBAS there was not enough time to develop a set of rules for correlating descriptions of events. The most significant inference made during this phase is the elimination of joint venture event descriptions from consideration if the descriptions include references to fewer than two non-coreferential partners. One would expect that a failure to correlate event descriptions should result in a higher number of spurious actual events being reported. Fortunately, however, the generation of spurious events was not a serious problem in the MUC-5 task.[15]

The majority of rules constituting the extensional reasoning phase actually have very little to do with inferring information conveyed in an input text. Instead, the purpose of most rules in this phase is to generate the database objects which are to be returned as the system's output. From a knowledge-engineering perspective, this task is not terribly interesting, but it nevertheless takes a significant amount of effort to implement.[16]

AN EXTENDED EXAMPLE

In this section, we illustrate in a more concrete fashion how the MUC-5 version of CBAS goes about extracting information by examining in detail what happens during the processing of a specific text in the MUC-5 corpus. Our discussion will proceed through the three processing phases which have been identified. Figure 2 contains the sample message upon which the discussion is based.

[14] A commonly appearing form of coreference is "part-whole" reference. Here is an example from a MUC evaluation text:

> WE HAVE ALSO LEARNED THAT TWO VEHICLES OF THE SALVADORAN RED CROSS HAVE ALSO BEEN ATTACKED. ONE OF THEM WAS TOTALLY DESTROYED BY FIRE IN THE MEJICANOS SECTOR, AND AN AMBULANCE WAS ATTACKED NEAR THE NATIONAL UNIVERSITY.

In this case, it is desirable for a data extraction system to realize that only two vehicles were attacked, not four (i.e., that "TWO VEHICLES" corefers with "ONE OF THEM ..." and "AN AMBULANCE").

[15] The MUC-4 implementation of CBAS did contain a number of event merging rules. The rules implemented heuristics for merging events if there was significant overlap in the objects participating in the events.

[16] Making the generation of output templates a completely separate processing phase would be a straightforward and reasonable task; we haven't bothered to do so because of other, more significant issues which have needed addressing.

<doc>

<DOCNO> 0592 </DOCNO>

<DD> NOVEMBER 24, 1989, FRIDAY </DD>

<SO> Copyright (c) 1989 Jiji Press Ltd.; </SO>

<TXT>

BRIDGESTONE SPORTS CO. SAID FRIDAY IT HAS SET UP A JOINT VENTURE IN TAIWAN WITH A LOCAL CONCERN AND A JAPANESE TRADING HOUSE TO PRODUCE GOLF CLUBS TO BE SHIPPED TO JAPAN.

THE JOINT VENTURE, BRIDGESTONE SPORTS TAIWAN CO., CAPITALIZED AT 20 MILLION NEW TAIWAN DOLLARS, WILL START PRODUCTION IN JANUARY 1990 WITH PRODUCTION OF 20,000 IRON AND "METAL WOOD" CLUBS A MONTH. THE MONTHLY OUTPUT WILL BE LATER RAISED TO 50,000 UNITS, BRIDGESTON SPORTS OFFICIALS SAID.

THE NEW COMPANY, BASED IN KAOHSIUNG, SOUTHERN TAIWAN, IS OWNED 75 PCT BY BRIDGESTONE SPORTS, 15 PCT BY UNION PRECISION CASTING CO. OF TAIWAN AND THE REMAINDER BY TAGA CO., A COMPANY ACTIVE IN TRADING WITH TAIWAN, THE OFFICIALS SAID.

BRIDGESTONE SPORTS HAS SO FAR BEEN ENTRUSTING PRODUCTION OF GOLF CLUB PARTS WITH UNION PRECISION CASTING AND OTHER TAIWAN COMPANIES.

WITH THE ESTABLISHMENT OF THE TAIWAN UNIT, THE JAPANESE SPORTS GOODS MAKER PLANS TO INCREASE PRODUCTION OF LUXURY CLUBS IN JAPAN.

</TXT>

</doc>

Figure 2: A Sample MUC-5 Message.

Tokenization

Figure 3 contains a sampling of the basic facts created during the tokenization stage for the example message. These facts are in a format appropriate for processing by CLIPS. The **txt_token** facts identify the locations of lexical items, and the **sentence** and **paragraph** facts identify the locations of sentence and paragraph boundaries, respectively. The part-of-speech tagger is invoked during the delimitation of word tokens, and part-of-speech categories returned by the tagger are added to the other information collected in the tokenization process.[17] The company name parser, which is responsible for generating the **company_name** facts illustrated in Figure 3, relies upon the presence of word and sentence boundaries. (All of the **company_name** facts generated for this example are listed.)

Intensional Reasoning.

At the start of the second stage of processing, the set of basic facts detected during tokenization are asserted to the factbase of the CLIPS-based intensional reasoning component. Once this is done, the forward-chaining engine is invoked to infer information about possible events, objects, and attributes.

In the example text, only one reference to a joint venture event is detected—in the first sentence, the phrase *SET UP A JOINT VENTURE* triggers the inference that an event reference has occurred. The phrase *THE JOINT VENTURE* does not trigger an event reference because it is recognized as a definite reference; however, this definite reference is recorded.

The company name parser invoked in the tokenization phase has detected the presence of several possible company name references. Based on testing of the company name parser, it is known that whenever the metric it assigns to a possible name is less than 1.0, the likelihood that an actual company name is present is relatively low, and consequently, any possible company names with less than 1.0 likelihood are thrown out. This heuristic generally works very well (as a heuristic should), but in this example a company name is excluded that it would have been better to keep: *BRIDGESTONE SPORTS CO.* And because of this error, CBAS misses the identification of one of the parents of the detected joint venture. The heuristic also fails to rule out *TRADING HOUSE* as a plausible company name and consequently it is incorrectly inferred to be a reference to a parent company. The other two parents in the joint venture, *UNION PRECISION CASTING CO.* and *TAGA CO.*, are correctly identified.

Rules for determining the roles played by companies typically involve the detection of a company name in a syntactic context within which a relationship of a certain type is likely to be mentioned. For example, definite references to joint ventures followed by a comma followed by a company name typically signal that the company name denotes a company in a child role. It is for this reason that *BRIDGESTONE SPORTS TAIWAN CO.* in the context *THE JOINT VENTURE, BRIDGESTONE SPORTS TAIWAN CO.* is inferred to be referring to a child.

Extensional Reasoning

The extensional reasoning phase is implemented as a completely separate CLIPS process. During this processing phase, decisions are first made about which events are actual and which events are spurious. No effort is made in the MUC-5 version of CBAS to correlate events. The primary processing strategy is a simple one: do not instantiate events which do not have two or more non-coreferential partners. In the sample text, only one event is inferred, and since it has two or more partners, it is instantiated. The template generated by CBAS for this example is given in Figure 4.[18]

[17] In general, we have found it advantageous (both in terms of rule-writing convenience and processing speed) to have "fat" facts. That is, to include as much information in a single clause as is reasonable instead of distributing information across clauses. For this reason, the **sentence** and **paragraph** facts are actually used very little; instead, information about sentence and paragraph membership is built into the **txt_token** facts. (In the example rules and facts shown in this paper, a number of features irrelevant to the discussion have been eliminated to make the presentation more concise and lucid.)

[18] An important strategy which we employed in MUC-5 was simply not to try to extract every possible detail specified in the

```
(txt_token (l 0)(r 1)(cat NN)(cite BRIDGESTONE))
(txt_token (l 1)(r 2)(cat NNS)(cite SPORTS))
(txt_token (l 2)(r 3)(cat NN)(cite CO.))
(txt_token (l 3)(r 4)(cat VBD)(cite SAID))
(txt_token (l 4)(r 5)(cat RB)(cite FRIDAY))
(txt_token (l 5)(r 6)(cat PP)(cite IT))
(txt_token (l 6)(r 7)(cat VBZ)(cite HAS))
(txt_token (l 7)(r 8)(cat VBN)(cite SET))
(txt_token (l 8)(r 9)(cat IN)(cite UP))
(txt_token (l 9)(r 10)(cat DT)(cite A))
(txt_token (l 10)(r 11)(cat JJ)(cite JOINT))
(txt_token (l 11)(r 12)(cat NN)(cite VENTURE))
(txt_token (l 12)(r 13)(cat IN)(cite IN))
(txt_token (l 13)(r 14)(cat NN)(cite TAIWAN))
(txt_token (l 14)(r 15)(cat IN)(cite WITH))
(txt_token (l 15)(r 16)(cat DT)(cite A))
(txt_token (l 16)(r 17)(cat JJ)(cite LOCAL))
(txt_token (l 17)(r 18)(cat NN)(cite CONCERN))
(txt_token (l 18)(r 19)(cat CC)(cite AND))
(txt_token (l 19)(r 20)(cat DT)(cite A))
(txt_token (l 20)(r 21)(cat DT)(cite JAPANESE))
(txt_token (l 21)(r 22)(cat NN)(cite TRADING))
(txt_token (l 22)(r 23)(cat NN)(cite HOUSE))
...
(sentence (n 1)(p 1)(l 0)(r 33))
...
(paragraph (n 1)(l 0)(r 33))
...
(company-name (s 1)(l 0)(r 1)(metric 1.000000)(cite BRIDGESTONE))
(company-name (s 1)(l 21)(r 23)(metric 1.000000)(cite TRADING HOUSE))
(company-name (s 1)(l 0)(r 3)(metric 0.950000)(cite BRIDGESTONE SPORTS CO.))
(company-name (s 2)(l 37)(r 38)(metric 1.000000)(cite BRIDGESTONE))
(company-name (s 2)(l 51)(r 52)(metric 0.600000)(cite START))
(company-name (s 2)(l 37)(r 41)(metric 0.950000)(cite BRIDGESTONE SPORTS TAIWAN CO.))
(company-name (s 4)(l 94)(r 95)(metric 0.600000)(cite SOUTHERN))
(company-name (s 4)(l 102)(r 103)(metric 1.000000)(cite BRIDGESTONE))
(company-name (s 4)(l 108)(r 109)(metric 0.600000)(cite UNION))
(company-name (s 4)(l 109)(r 110)(metric 0.600000)(cite PRECISION))
(company-name (s 4)(l 125)(r 126)(metric 0.600000)(cite TRADING))
(company-name (s 4)(l 108)(r 112)(metric 0.950000)(cite UNION PRECISION CASTING CO.))
(company-name (s 4)(l 118)(r 120)(metric 0.950000)(cite TAGA CO.))
(company-name (s 5)(l 133)(r 134)(metric 1.000000)(cite BRIDGESTONE))
(company-name (s 5)(l 137)(r 138)(metric 0.600000)(cite FAR))
(company-name (s 5)(l 146)(r 147)(metric 0.600000)(cite UNION))
(company-name (s 5)(l 147)(r 148)(metric 0.600000)(cite PRECISION))
(company-name (s 6)(l 160)(r 161)(metric 0.600000)(cite UNIT))
```

Figure 3: Tokenization Output.

```
<TEMPLATE-0592-1> :=
    DOC NR:  0592
    DOC DATE: 241189
    DOCUMENT SOURCE: "Jiji Press Ltd."
    CONTENT: <TIE-UP-RELATIONSHIP-0592-1>
<TIE-UP-RELATIONSHIP-0592-1> :=
    TIE-UP STATUS:  EXISTING
    ENTITY: <ENTITY-0592-1>
            <ENTITY-0592-2>
            <ENTITY-0592-3>
    JOINT VENTURE CO: <ENTITY-0592-4>
<ENTITY-0592-1> :=
    NAME: UNION PRECISION CASTING CO
    NATIONALITY: Taiwan (COUNTRY)
    TYPE: COMPANY
    ENTITY RELATIONSHIP: <ENTITY-RELATIONSHIP-0592-1>
<ENTITY-0592-2> :=
    NAME: TRADING HOUSE
    TYPE: COMPANY
    ENTITY RELATIONSHIP: <ENTITY-RELATIONSHIP-0592-1>
<ENTITY-0592-3> :=
    NAME: TAGA CO
    TYPE: COMPANY
    ENTITY RELATIONSHIP: <ENTITY-RELATIONSHIP-0592-1>
<ENTITY-0592-4> :=
    NAME: BRIDGESTONE SPORTS TAIWAN CO
    ALIASES: "BRIDGESTONE"
    TYPE: COMPANY
    ENTITY RELATIONSHIP: <ENTITY-RELATIONSHIP-0592-1>
<ENTITY-RELATIONSHIP-0592-1> :=
    ENTITY1: <ENTITY-0592-1>
             <ENTITY-0592-3>
             <ENTITY-0592-2>
    ENTITY2: <ENTITY-0592-4>
    REL OF ENTITY2 TO ENTITY1: CHILD
    STATUS: CURRENT
```

Figure 4: Template generated for example text.

CONCLUSIONS

A motivating factor in the design of CBAS has been a desire to exploit simple data extraction methods to the fullest extent possible. We do not believe that we have fully exploited the capabilities of non-linguistic data extraction methods and intend to continue exploring such techniques, especially special-purpose parsers. However, at the same time, we do believe that linguistic analysis techniques will ultimately be essential in data extraction applications, and our research group is actively engaged in the development of new linguistically-based methodologies which meet the portability, reliability, accuracy, and speed requirements of large-scale systems.

Another motivating factor has been the desire to build a relatively inexpensive system which individuals with no training whatsoever in linguistics could develop and maintain. The current implementation of CBAS certainly demonstrates that we have been successful in meeting this goal: the primary implementation media, Perl and CLIPS, are available at little or no cost; and we have made successful use of rule developers with little or no experience in linguistic analysis.

Finally, the most significant factor in the design of CBAS has been a desire to exploit multiple preprocessors in the same way that multiple sensors are exploited in multisensor data fusion engines. The basic idea behind this design concept is simple: by having many different processors contributing information, the failure of any one processor will not result in a lot of information being lost. Thus, instead of having a single NLP parser from which all information regarding constituent structure is derived, multiple specialized parsers are implemented, parsers for recognizing company names, dates, names of individuals, place names, and so forth. In this type of situation, different parsers may contribute "competing information". For example, a company name parser may determine that a given substring denotes the name of a company whereas a place name parser may determine that the same substring denotes the name of a city.

We have not yet actually proven the merit of the "multisensor" approach: there is no "sensor management" capability in existing CBAS implementations to compensate for preprocessor failure, nor is there any methodology in place for managing competing processor output. We of course intend to pursue the goal of proving the utility of this approach in future evaluation efforts with more sophisticated implementations of the CBAS architecture. In future implementations we are particularly interested in the possibility that a multisensor approach will provide a natural framework for the development of *interactive* data extraction systems in which the multiple preprocessors extract "basic" objects and relations (ie, an *ontology*) from which composite structures are derived in response to user extraction queries (which are constrained by the ontology and a set of composition rules defined over it).

REFERENCES

[1] Laura Blumer Balcom and Richard M. Tong. Advanced decision systems: Description of the Codex system as used for MUC-3. In *Proceedings of the Third Message Understanding Conference*, pages 129–136, San Diego, May 1991. Morgan Kaufmann Publishers, Inc.

[2] Charles L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1), 1982.

[3] Carnegie Group. Text categorization shell. Technical brief, Carnegie Group, Five PPG Place, Pittsburgh, PA 15222, 1989.

[4] Tomek Stralkowski. TTP: a fast and robust parser for natural language. Technical report, New York University Department of Computer Science, New York, NY, 1991.

[5] Edward Waltz and James Llinas. *Multisensor Data Fusion,*. Artech House, Norwood, MA, 1990.

---

template specification formulated for the evaluation, but to only extract key features which had the most payoff in points. This was an extremely useful strategy in terms of CBAS's performance with respect to other systems participating in MUC-5.