Building up and Making Use of Corporate Knowledge Repositories

Gian Piero Zarri

Centre National de la Recherche Scientifique
54, boulevard Raspail
75270 Paris Cedex 06, France
zarri@cams.msh-paris.fr

Abstract

In this paper, we present a methodology for the construction and use of corporate knowledge repositories. We consider here only the "textual component" of corporate knowledge — i.e., all sort of economically valuable, natural language (NL) documents like news stories, telex reports, internal documentation, normative texts, intelligence messages, etc. In this case, we suggest that the construction of effectively usable corporate knowledge repositories can be achieved with the translation of the original documents into some type of conceptual format. The "metadocuments" obtained in this way can then be stored into a knowledge repository (knowledge base) and, given their role of advanced document models, all the traditional functions of information retrieval, (searching, retrieving and producing an answer) can be directly executed on them. We illustrate here the architecture of a prototypical, implemented system used to exploit a knowledge repository of metadocuments.

Introduction

Until now, the problem of dealing with the dispersed know-how that exists in a corporation ("corporate memory") has been normally dealt with as a problem of "modelling", see KADS (Breuker & Wielinga 1989), (Akkermans et al. 1993), EM (Bubenko 1993), (Kirikova & Bubenko 1994), etc. In this very sophisticated, cyclical and structured approach to KBS development, one of the main hypothesis — that which justifies the use of this sort of methodologies as practical, industrial tools — concerns the possibility of detecting, within the global, "expert" behaviour of a corporation, a set of elementary tasks, independent from a particular application domain. Once the tasks discovered and formalised, they can be used to set up libraries of basic building blocks, to be reused for the description of a large number of intellectual processes in a company. This endeavour is really very ambitious — which explains why so many studies in this domain limit themselves to a purely theoretical approach — and meets all sort of practical problems, ranging from the difficulties in defining the building blocks in a really general way to the ambiguities concerning which aspects (the model or the code) of the blocks can really be reused.

In this paper, we suggest that a more modest, but less fuzzy and immediately useful approach to the practical use of corporate memory which should not be neglected can be found in the (at least partially automated) construction and use of corporate knowledge repositories. They can be defined as on-line, computer-based storehouses of expertise, knowledge, experience and documentation about particular aspects of a given corporation. We will consider here only the "textual component" of corporate knowledge — i.e., all sort of economically valuable, natural language (NL) documents like news stories, telex reports, internal documentation (memos, policy statements, reports and minutes), normative texts, intelligence messages, etc.

In this case, we suggest that the construction of effectively usable corporate repositories can be achieved with the translation of the original documents into some type of advanced conceptual representation, e.g., semantic nets (Lehmann 1992), frames (Bobrow & Winograd 1977), conceptual graphs (Sowa 1984), etc. The "metadocuments" obtained in this way can then be stored into a knowledge repository (knowledge base) and, given their role of advanced document models, all the traditional functions of information retrieval, e.g., searching, retrieving and producing an answer (and other functions like the intelligent navigation inside the repository) can be directly executed on them. We remember here that, in information retrieval, a document model (Boolean models, vector models, probabilistic models ...) is assumed to represent, in some way, the semantic or informational content of the documents under consideration. For example, in the traditional Boolean model, the content of the document is represented simply as a set of keywords ; a "correspondence function" will try then to find a match between this set and a query expressed, in turn, by keywords grouped into some form of Boolean combination (i.e., making use of the usual and, or, not ... operators).

In the following, we describe the main features of a prototypical system, devoted to exploit a knowledge repository of metadocuments, that we have (partially) built up thanks to the aid of the European Commission (Esprit project NOMOS, P5330 and LRE project COBALT, P61011) and of the French National Centre for Scientific Research (CNRS). After having outlined the architecture of the prototype, we will introduce NKRL (Narrative Knowledge Representation Language), the conceptual language we use for the production of metadocuments. We will then examine in some detail the interrogation sub-

system of the prototype, which takes advantage of the canonical format of the metadocuments to implement some sort of advanced search and retrieval applications.

Dealing with textual corporate knowledge

Figure 1 gives a very general overview of the architecture of the system we propose. Please note that all the blocks shown in this figure already exist, at least in a prototypical form ; however, the procedures concerning the selection and the activation of the high-level inference procedures, at the bottom, realised in a CNRS context, are still not integrated with the remaining blocks, realised mainly in a NOMOS/COBALT context. All the blocks of Figure 1 have been implemented in COMMON LISP. In NOMOS/COBALT, we have used as support platform an object-oriented environment : CRL (Carnegie Representation Language) in the NOMOS version, QSL (Quinary Semantic Language) in the COBALT version.

Supposing the integrated system already exists, the original NL documents, on the left, pass through the NL processing modules which constitute the core of the acquisition sub-system. This sub-system executes, essentially, a conversion of the "meaning" of the original documents into NKRL format. The results of this translation activity are stored into the central knowledge repository (metadocument knowledge repository) ; in very large environments it should also be possible, of course, to think in terms of distributed communicating repositories. Some functions, not shown in Figure 1 for simplicity's sake, allow browsing and maintenance operations on the contents of the repository. Moreover, some possibilities of immediate display of the resulting NKRL code have been added recently in a COBALT context.

The boxes on the right-hand side of Figure 1 represent the query sub-system of the prototype. The user's queries are, firstly, translated into search patterns or sequences of search patterns — search patterns are NKRL data structures which represent the general framework of information to be searched for, by filtering or unification ("correspondence function"), within the metadocument repository. The metadocuments can i) directly unify the original query (search pattern) ; ii) unify some new search patterns obtained by inference from the original one, see below. After the unification of the patterns with one or more metadocuments, these are presented to the user. The output can be shown in two different formats, which can also coexist : i) in a concise, tabular format (immediate display); ii) as the original NL messages, if they have been stored along with the corresponding metadocuments.

A quick glimpse of NKRL

NKRL (Narrative Knowledge Representation Language) supplies a language-independent description of the semantic content (the "meaning") of non-trivial natural language documents, like the textual corporate documents introduced above.

The basic structures

NKRL is a two layer language.

The lower layer supplies a set of general tools. It consists of four integrated components

The descriptive component tools are used to set up the formal NKRL expressions (templates) describing general classes of narrative events, like "moving a generic object", "formulate a need", "be present somewhere". Templates encode, therefore, the standard features of various ordinary events and human activities. Templates are structured into a hierarchy, H_TEMP(lates), which can be equated to a taxonomy (ontology) of events, see (Zarri 1995a).

Templates' instances (called occurrences), i.e., the NKRL representations of single, specific events like "Tomorrow, I will move the wardrobe", "Lucy was looking for a taxi", "Peter lives in Paris", are in the domain of the factual component. Templates and occurrences are characterised by a tripartite format : (P_i (R_1 a_1) (R_2 a_2) ... (R_n a_n)), where the central piece is a semantic predicate P_i (like BEHAVE, EXPERIENCE, MOVE, PRODUCE etc.) which represents a named relation that exists among one or more arguments a_i introduced by means of roles R_i (like SUBJ(ect), OBJ(ect), SOURCE, etc.). From a formal point of view, a metadocument consists, therefore, of the association of several templates and/or occurrences, tied up by particular binding structures, see (Zarri 1994), that represent the logico-semantic links which can exist between them (the co-ordination and subordination links, using a metaphor from the domain of natural language).

The definitional component supplies the formal representation of all the general notions, like *physical_entity*, *taxi_*, *location_*, which can be used as arguments in the two components above. Their representations in NKRL terms are called concepts (basically, their data structures can be equated to frames), and are grouped into a hierarchy, H_CLASS(es) — H_CLASS corresponds well, therefore, to the usual concept ontologies, see (Zarri 1995a).

The instances of concepts, like lucy_, wardrobe_1, paris_ are called individuals, and pertain to the enumerative component. For more details about other important tools — e.g., those allowing the representation of modal or temporal information — see (Zarri 1992a), (Zarri 1994).

The upper layer of NKRL includes the standard conceptual structures built up making use of the general tools above, and it consists of two parts.

The first is a catalogue, where we can find a description of the formal characteristics and the modalities of use of the well-formed, "basic" templates (like "moving a generic object" mentioned above) associated with the language — presently, about 150, pertaining mainly to a (very general) socio-economico-political context where the main characters are human beings or social bodies (e.g., a company). By means of proper specialisation operations, it is then possible to obtain from the basic templates the (specific) "derived" templates that could be needed in the context of an application — e.g., "move an industrial

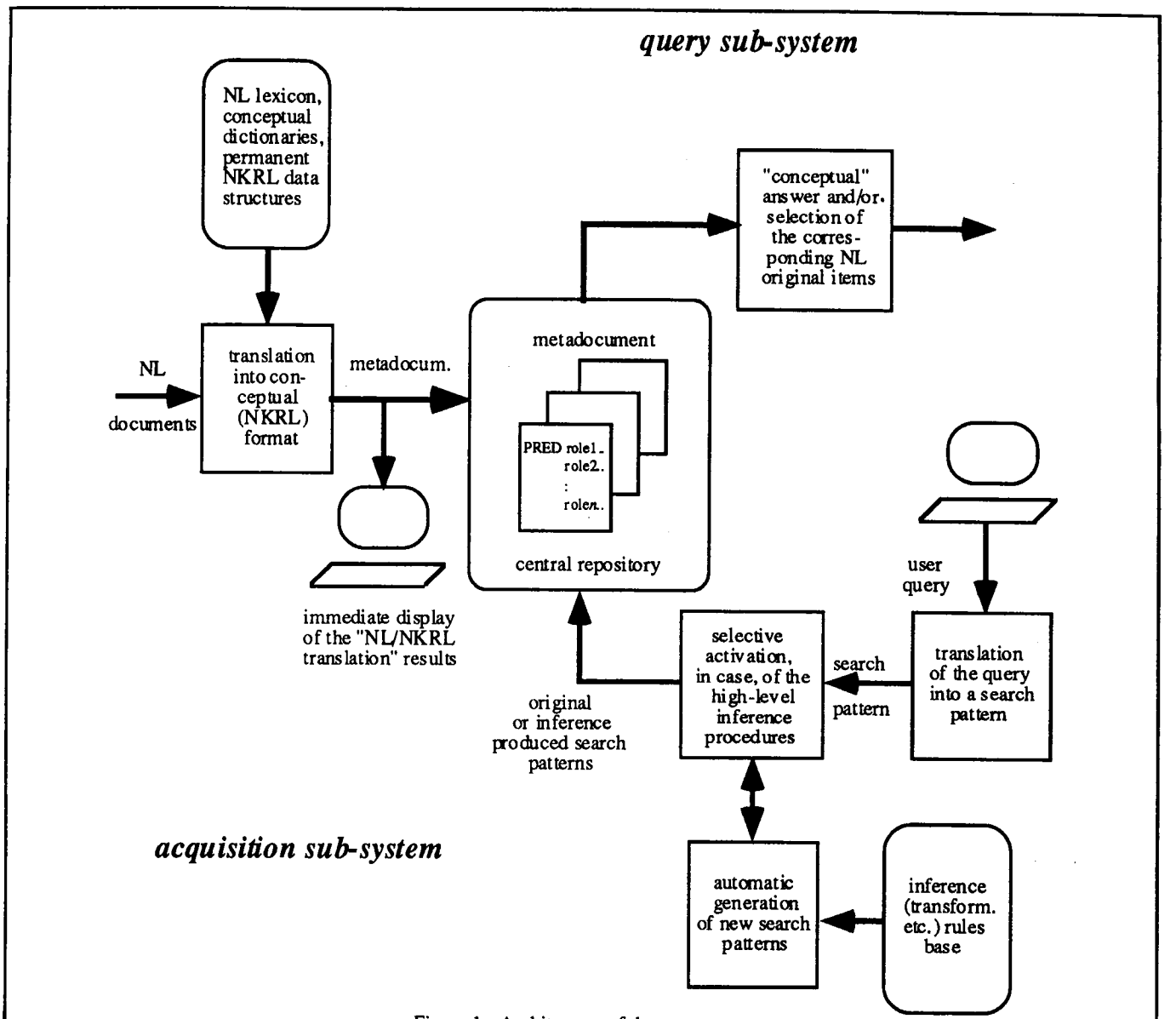

Figure 1 - Architecture of the prototype.

process" — and the corresponding occurrences. In NKRL, the set of legal, basic templates included in the catalogue can be considered as fixed. This means that : i) a system-builder does not have to create himself the knowledge needed to describe the events proper to a large class of NL documents ; ii) it becomes easier to secure the reproduction and the sharing of previous results.

The second part is made up of the general concepts which pertain to the upper levels of H_CLASS and which form a sort of standard, invariable ontology.

An example

We supply now, see Figure 2, a simple example of NKRL code (a metadocument consisting only of occurrences). It translates a small fragment of news in the COBALT's style: "Milan, October 15, 1993. The financial daily Il Sole 24 Ore reported Mediobanca had called a special board meeting concerning plans for capital increase".

In Fig. 2, c1 and c2 are the symbolic names of two occurrences, instances of NKRL basic and derived templates. MOVE and PRODUCE are predicates ; SUBJ, OBJ, TOPIC (the theme, "à propos of ...", of the event(s) or situation(s) represented in the occurrence) are roles. With respect now to the arguments, sole_24_ore, milan_, mediobanca_ (an Italian merchant bank), summoning_1, etc. are individuals ; *financial_daily*, *special_*, *cardinality_* (which pertains to the *property_* sub-tree of H_CLASS) and *several_* (belonging, like *some_*, *all_* etc., to the *logical_quantifier* sub-tree of H_CLASS) are concepts. The attributive operator, SPECIF(ication), is one of the four NKRL operators used to build up structured arguments

(or "expansions"), see below and (Zarri 1994), (Zarri & Gilardoni 1996) ; the SPECIF lists are used to represent some properties of the first element of the list. *several_* is used within a *cardinality_* SPECIF list as a standard way of representing the plural number mark, see c2.

```
c1)  MOVE      SUBJ   (SPECIF sole_24_ore
                          financial_daily): (milan_)
               OBJ    #c2
               date-1: 15_october_93
               date-2:

c2)  PRODUCE   SUBJ   mediobanca_
               OBJ    (SPECIF summoning_1 (SPECIF
                          board_meeting_1
                          mediobanca_ special_))
               TOPIC  (SPECIF plan_1 (SPECIF
                          cardinality_ several_)
                          capital_increase_1)
               date-1: circa_15_october_93
               date-2:
```

Figure 2 - An example of NKRL coding.

The arguments, and the templates/occurrences as a whole, may be characterised by the addition of determiners (attributes). In particular, the location attributes (lists) are associated with the arguments by using the colon, ":", operator, see occurrence c1. For a recent paper on the NKRL representation of the temporal determiners, date-1 and date-2, see (Zarri 1992a).

Please note that the MOVE basic template at the origin of the occurrence c1 is necessarily used to translate any event concerning the transmission of an information ("The financial daily Il Sole 24 Ore reported ..."). It makes use of what is called a "completive construction". Accordingly, the filler of the OBJ(ect) slot in the occurrences (here, c1) which instantiates the transmission template is always a symbolic label (here, #c2) which refers to another predicative occurrence, i.e., the occurrence bearing the informational content to be spread out ("...Mediobanca had called a meeting..."). For other sorts of NKRL structures (binding structures) allowing to build up second order objects from templates and occurrences, see, e.g., (Zarri 1994). See (Zarri 1995a) for some additional information about NKRL's "ontologies", ontology of events (H_TEMP) and ontology of concepts (H_CLASS)

The querying and inferencing procedures

Generalities

Each of the four components of NKRL is characterised by the association with a class of proper inference procedures.

For example, the specialised query language for dealing with the definitional and enumerative component data structures fulfils the following minimal requirements :

- it must be possible to specify a query to match a specific concept/individual ;
- it must be possible to specify a query to match a set of individuals characterised by being offsprings of a (set of) specific H_CLASS concepts (e.g., all the financial companies, or all financial companies excepting the French ones), and, if necessary, characterised by a set of attribute-slots with specific values (e.g., all companies whose "capital" slot holds a value ≥ 2,000,000).
- it must be possible to introduce variables to be bound to specific parts of the matched concept/individuals.

From the above, it is now evident that the main inference mechanism associated with the definitional and enumerative components must be the usual taxonomic inheritance mechanism which proceeds over IsA and InstanceOf relations. At least part of the basic building blocks for this mechanism are offered for free by the knowledge representation tools which support the NKRL's implementations : this is the case for both CRL, used in the NOMOS project, and QSL, used in COBALT. The subtyping relations to be used for the inheritance operations are directly encoded, in NKRL, in the H_CLASS hierarchy supporting the definitional component.

With respect now to the factual component, the key inference mechanism for this component (and the basic inference tool of NKRL) is the Filtering and Unification Module (FUM). The primary data structures handled by FUM are the search patterns that, as already stated, represent (in NKRL terms) the general properties of an information to be searched for, by filtering or unification, within a metadocument knowledge repository — a search pattern can be considered an NKRL equivalent of a natural language query, see Figure 1. For clarity's sake, we reproduce in Figure 3 the search pattern corresponding to the question : "Which was the theme of the recent board meeting called out by Mediobanca ?" that, obviously, unify with occurrence c2 in Figure 2. In this figure, we have voluntarily enlarged the scope of the question, by adding more constraints on the OBJ(ect) variable (*x*). The two dates of Figure 3 constitute the "search interval" associated with the search pattern : this interval is used to limit the search for unification to the slice of time which it is considered appropriate to explore, see (Zarri 1992a).

```
((?w    IS-OCCURRENCE
  :predicate PRODUCE
  :SUBJ  mediobanca_
  :OBJ   (SPECIF ?x (SPECIF ? y mediobanca_))
  :TOPIC ?z)
  (1_october_93, 20_october_93)
  ((?x IS-A (:OR assembly_ adjournment_ dissolution_))
  (?y  IS-A board_meeting)
  (?z  IS-A planning_activity)))
```

Figure 3 - A simple search pattern.

A generalisation of the FUM module is used for matching the templates of the descriptive component. A specific inference mechanism, based on FUM and proper to

this last component, is a join procedure (Sowa 1984) allowing for the merge of (two or more) basic or derived templates. Information on these particular merge procedures can be found, e.g., in (Zarri 1995b).

The AECS sub-language

In Figures 2 and 3, the arguments made up of a SPECIF list are examples of structured arguments (or "expansions").

In NKRL, structured arguments of templates and occurrences are built up by applying a specialised sublanguage, AECS, which includes four binding operators (see Table 1), the disjunctive operator (ALTERNative, A), the distributive operator (ENUMeration, E), the collective operator (COORDination, C), and the attributive operator (SPECIFication, S). Accordingly, structured arguments are lists of undefined length, which may include both concepts and individuals, and labelled using the AECS operators.

Operator	Mnemonic Description
ALTERN	The "disjunctive operator". It introduces a set of elements, i.e., concepts, individuals, or other expansion lists. Only one element of the set takes part in the particular relationship with the predicate defined by the role-slot to be filled ; however, this element is not known.
COORD	The "collective operator" : all the elements of the set take part (necessarily together) in the relationship with the predicate defined by the role-slot.
ENUM	The "distributive operator ": each element of the set satisfies the relationship, but they do so separately.
SPECIF	The "attributive operator". It links a series of attributes with the concept or individual that constitutes the first element of the SPECIF list. Each attribute that appears inside a SPECIF list can be recursively associated with another SPECIF list.

Table 1 - NKRL operators for structured arguments.

From a formal point of view, we note that, e.g. :

a) (SPECIF e1 a b) = (SPECIF e1 b a) ;

b) (ENUM e1 e2) = (e1 AND e2 AND ¬ (COORD e1 e2).

The first formula says that the order of the properties a, b, ... associated with en entity e1, concept or individual, is not significant. The second formula enunciates in a more formal way what already stated in Table 1 : the main characteristics of the ENUM lists is linked with the fact that the entities e_1, e_2, ... take part obligatorily in the particular relationship between the structured argument and the predicate expressed by the role which introduces the arguments, but they satisfy this relationships separately. A more complete description of the semantics of AECS can be found, e.g., in (Gilardoni 1993).

Because of their recursive nature, the AECS operators could give rise to very complex expressions, difficult to interpret and disentangle (unify). Therefore, to build up well-formed NKRL expansions, the definitions of Table 1 are used in association with the so-called "priority rule", which can be visualised by using the following expression:

(ALTERN (ENUM (COORD (SPECIF)))).

This is to be interpreted as follows : it is forbidden to use inside the scope of a list introduced by the binding operator *B*, a list labelled in terms of one of the binding operators appearing on the left of *B* in the priority expression above — e.g., it is impossible to use a list ALTERN inside the scope of a list COORD. An example of utilisation of this rule can be found, e.g., in (Zarri 1994).

A query language for structured arguments

We have seen above that the AECS language allows us to describe complex relations among concepts and individuals. While, sometimes, a query about an AECS structure must be able to exploit completely the information carried by this structure, the situation in which only part of this information is really useful is relatively frequent. For example, faced with an AECS expression like "(COORD john_ paul_)", which tells us that both John and Paul take part in a particular event in a co-ordinated manner, it is often the case that the information we want to obtain is simply if John (or Paul) takes part in the event.

Therefore, a query language operating on the AECS structures must be able to express a wide range of query modalities, and to obtain constantly the correct results. Keeping in mind that it is always possible to express the AECS structures (the AECS lists) in term of trees, the basic requirements for the AECS-query language (which is part of the FUM module) are :

1) It must be possible to specify a "perfect match", defined as a match that succeeds if and only if the query, and the target (matched) AECS expression, have the same identical structure (apart from variables), i.e., if the tree representations of the query and of the target expression are strictly identical. As an example, we can say that the query (ENUM *?x ?y*) succeeds against the target AECS expression (ENUM credit_lyonnais mediobanca_), but fails against (COORD credit_lyonnais (SPECIF mediobanca_ *merchant_bank*)) or against (ENUM credit_lyonnais mediobanca_ chase_manhattan).

2) It must be possible to specify a perfect match apart from "cardinality", i.e., a match that succeeds if and only if the query, and the target AECS expression, have the same identical structure — apart from variables and, chiefly, without taking into account the cardinality of the AECS lists. In this case, (ENUM *?x ?y*) succeeds against (ENUM credit_lyonnais mediobanca_) and against (ENUM credit_lyonnais mediobanca_ chase_manhattan), but fails, obviously, against (COORD credit_lyonnais mediobanca_).3) It must be possible to specify a "subsumed" match, i.e., a match that succeeds if and only if the query, and the target AECS expression, carry an

information which can be considered as globally congruent from a semantic point of view. For example, we admit here the presence, in the target (matched) expression, of additional SPECIF lists (additional lists of attributes). According to this paradigm, (COORD *?x ?y*) succeeds against (COORD credit_lyonnais mediobanca_), against (COORD credit_lyonnais (SPECIF mediobanca_ *merchant_bank*)), and (COORD credit_lyonnais mediobanca_ chase_manhattan).

4) It must be obviously possible to mix the above kind of queries, in such a way that, for example, *perfect match is required for the top level structure of the query and target trees but not for the underlying parts*, see also the examples of Figure 4 below. In this way, e.g., (ALTERN *?x ?y*) can match against (ALTERN chase_manhattan (COORD credit_lyonnais mediobanca_)).

We can now define a query language for AECS structures. For clarity's sake, the query language is based on the logical structures of the original AECS sub-language, augmented to allow a) the use of variables, and b) the correct specification of the kind of match required by the query. The AECS query language is therefore defined in the following way : take the AECS descriptive language as the basis, but allow :

• The use of variables (*?x*), possibly with constraints, instead of concepts or individuals.

• The use of the special construct STRICT-SUBSUMPTION, taking as argument an NKRL entity (variable, concept or individual), or a complex AECS structure. Bearing in mind the priority rule introduced in the previous sub-Section, these complex AECS structures, expressed as trees, may consist of : a simple list COORD (coord-list) ; a coord-tree, subsuming as "branches" at least two coord-lists ; an enum-tree, subsuming one or more coord-tree (or coord-list) ; an altern-tree, subsuming any of the previous tree-structures.

• The use of the special construct STRICT-CARDINALITY, taking as argument an NKRL element, or one of the AECS structures listed above.

The operational meaning of STRICT-CARDINALITY and STRICT-SUBSUMPTION is the following :

• the presence of a STRICT-SUBSUMPTION operator forces the interpretation of the argument according to a "no-subsumption" rule, thus requiring a perfect match, see point 1 before, on the type (NKRL entity, coord-branch, coord-tree, enum-tree, altern-tree)) of the argument ;

• the presence of a STRICT-CARDINALITY operator forces the interpretation of the argument according to a "fixed-cardinality" rule, thus requiring a perfect match, see point 2 before, on the cardinality of the argument ;

• the absence of any of the two special operators implies the "subsuming" rule, see point 3 before, thus producing a successful match if the semantics of the query construct *is subsumed* by the semantics of the matched construct.

The unification algorithms for the AECS-query language are described, e.g., in (Zarri & Gilardoni 1996).

To illustrate intuitively what expressed above, we give in Figure 4 some examples which are relative to different modalities of matching the target AECS structure : (ENUM chase_manhattan (COORD credit_lyonnais (SPECIF mediobanca_ *merchant_bank*) city_bank)).

• The query : (ENUM *?x* (STRICT-SUBSUMPTION (COORD credit_lyonnais mediobanca_ city_bank))) succeeds, binding *x* to chase_manhattan. Please note that the STRICT-SUBSUMPTION operator concerns only the general structure of the coord-trees, and not the structure of the single coord-branches.

• The query : (ENUM *?x* (COORD credit_lyonnais (STRICT-SUBSUMPTION mediobanca_) city_bank)) fails, given that the STRICT-SUBSUMPTION restriction prevents mediobanca_ from matching a coord-branch.

• The query : (ENUM *?x* (STRICT-CARDINALITY (COORD credit_lyonnais mediobanca_))) fails, because of the STRICT-CARDINALITY restriction.

• The query : (STRICT-SUBSUMPTION (ENUM *?x* (COORD credit_lyonnais mediobanca_))) succeeds, binding *x* to chase_manhattan. The STRICT-SUBSUMPTION restriction only concerns the top-level structure of the enum-trees.

• The query : (ENUM *?x* credit_lyonnais) succeeds, binding *x* to chase_manhattan.

• The query : (STRICT-SUBSUMPTION (ENUM *?x* (STRICT-SUBSUMPTION credit_lyonnais))) fails, see the (STRICT-SUBSUMPTION credit_lyonnais) restriction.

• The query : (STRICT-SUBSUMPTION (COORD credit_lyonnais mediobanca_)) fails.

• The query : (COORD credit_lyonnais mediobanca_) succeeds.

Figure 4 - Examples of AECS unifications

An example of high-level inference procedures

The basic querying and inference mechanisms outlined above are used as basic building blocks for implementing all sort of high level inference procedures like, e.g., the transformation rules, see (Zarri 1986), (Ogonowski 1987).

NKRL "transformations" deal, with the problem of obtaining a plausible answer from a database of factual occurrences also in the absence of the explicitly requested information, by searching semantic affinities between what is requested and what is really present in the repository. The fundamental principle employed is then to transform the original query into one or more different queries which — unlike the "transformed" queries in a database context — are not strictly "equivalent" but only "semantically close" to the original one.

To give a very simple example, suppose that, working in the context of an hypothetical metadocument database about university professors, we should want to ask a question like : ``Who has lived in the United States", even

without an explicit representation of this fact in the base. If this last contains some information about the degrees obtained by the professors, we can tell the user that, although we do not explicitly know who lived in the States, we can nevertheless look for people having an American degree. This last piece of information, obtained by transformation of the original query, would indeed normally imply that some time was spent by the professors in the country, the United States, which issued their degree.

Transformation rules are made up of a left hand side — formulation in NKRL (search pattern) format of the linguistic expression which is to be transformed — and one or more right hand sides — NKRL representations of one or more linguistic expressions that must be substituted for the given one. A transformation rule can, therefore, be expressed as : A (left hand side) $\Rightarrow$ B (right hand side). The "transformation arrow", "$\Rightarrow$", has a double meaning :

- operationally speaking, the arrow indicates the direction of the transformation : the left hand side A (the original search pattern) is removed and replaced by the right hand side B (one or more new search patterns) ;
- the standard logical meaning of the arrow is that the information obtained through B implies the information we should have obtained from A.

In reality, the "always true" implications (noted as $B \rightarrow A$, where we assume that the symbol "$\rightarrow$" represents, as usual, the implication arrow) are not very frequent. Most transformations found in real world applications represent "modalised implications". We will note them as $\Diamond(B \rightarrow A)$, which means "it is possible that B implies A". "$\Diamond$" is the usual modal operator for "possibility", which satisfies then the relation $\Diamond p = \neg \blacklozenge \neg p$ with respect to the second modal operator, "$\blacklozenge$ = necessity". An example of modalised transformation is given by the transformation t1 in Fig. 5, which allows us to deal — by using an inference engine based on the FUM module, see (Zarri 1986) for some details — with the informal example above about "university professors" ; as we can see, the antecedent and consequent of t1 are formed by search patterns, slightly simplified here for clarity's sake. Transformation t1 says : "If someone (x) receives a title from an official authority by means of an official document, then it is possible that he has been physically present at that moment in the place (k) where the authority is located". This rule, for example, is not always valid in the case of an university degree (it could be obtained in a correspondence school, etc.). Nevertheless, it is possible to see that, in this case, the semantic distance between an "always true" implication and a "modalised" one is not too important, as it is becomes possible, at least in principle, to change t1 into a true transformation by the addition of a few constraints on the variable p , for instance the constraint: $p \neq$ *<obtainable_by_correspondence_degree>*. More examples, and a complete "theory" of transformations, can be found in, e.g., (Zarri 1986).

Coming back to the general architecture of the prototype in Figure 1 above, we can see that a search pattern may be generated from outside the system when it directly represents the NKRL translation of a query issued by the user — this corresponds to the basic mode of functioning of the query sub-system of the prototype. But, in some cases, it may be also generated automatically from inside the system when executing the inference procedures.

t1) EXIST SUBJ x : [l] $\Rightarrow$ RECEIVE SUBJ x
OBJ y
SOURCE w : [l]
MODAL z

x = *<human_being>*
y = *<title_>*
w = *<authority_>*
z = *<official_document>*
l = *<location_>*

Figure 5 - A simple example of "transformation" rule.

Let us consider, in fact, what will happen to a search pattern corresponding directly to a user's NL query when, having used this search pattern in order to ask the knowledge base, we obtain no answer, or when, having recovered some information, we would like to know more. In such cases we can consider, as a first hypothesis, to be in a situation where the NKRL image of the information which could supply a plausible answer effectively exists in the central knowledge base, but it may be difficult to retrieve and recognise. Under this hypothesis, we will ask the query system to automatically transform the initial search pattern by substituting it with another "semantically equivalent" pattern, see above. Unfortunately, the problems associated with the practical utilisation of this type of (implemented) approach are not only technical (construction of appropriate inference engines), but concern mainly the way of i) discovering the common sense rules which constitute the real foundation of the transformation procedures, and ii) formalising them so that we can obtain a sufficient degree of generality. These two activities can be executed *a priori* only to some extent, because the knowledge engineers find difficulty in predicting all the possible practical situations. Concretely, the transformation rules are established often *a posteriori*, by abstracting and formalising some procedures empirically found in order to solve particular "cases". This is why we plan to introduce, in a more complete version of the prototype, the possibility of having recourse to Case-Based Reasoning techniques, see (Kolodner 1992), (Wess, Althoff & Richter 1994), etc. Given that the "rules", when they exist, are normally considered of a more economic use than "cases", we would like to use the CBR procedures not only for providing the users with a sophisticated and up-to-date problem-solving modality, but also, at the same time, for blazing a trail toward the creation of a practical set of transformation rules to be stored in the rule base, and which will subsume all the different concrete cases used empirically to set up an useful solution.

Conclusion

In this paper, we have suggested that a modest, but pragmatically useful modality of use of what we have called the "textual component" of corporate knowledge could be obtained by translating this component into conceptual format (metadocuments). In this case, it becomes possible, in fact, to make use of the textual corporate knowledge according to the specific characteristics of its proper "meaning". We have then illustrated the general architecture of a prototypical system, implemented to a large extent, for setting up and exploiting a knowledge repository of metadocuments, and we have supplied more detailed information about some fundamental building blocks of the prototype.

The different versions of the implemented components have been tested as far as possible, and their performances seem to be satisfactory. For example, in the COBALT project, we have used a corpus of about 200 candidate news items which have then been translated into NKRL format, and examined through a query system in order to i) confirm their relevance ; ii) extract their main content elements (actors, circumstances, locations, dates, amounts of shares or money, etc.). Of the candidate news, 80% have been (at least partly) successfully translated; "at least partly" means that, sometimes, the translation was incomplete due, e.g., to the difficulty of instantiating correctly some binding structures. We plan to ask the European Commission for a financial aid in order to set up a new, industrial prototype, able to be tested in the environment of a real, industrial user.

References

Akkermans, H., van Harmelen, F., Schreiber, G., and Wielinga, B. 1993. A Formalization of Knowledge-Level Models for Knowledge Acquisition. *International Journal of Intelligent Systems* **8**: 169-208.

Bobrow, D.G., and Winograd, T. 1977. An Overview of KRL, a Knowledge Representation Language. *Cognitive Science* **1**: 3-46.

Breuker, J.A., and Wielinga, B.J. 1989. Model Driven Knowledge Acquisition. In *Topics in the Design of Expert Systems*, Guida, P., and Tasso, G., eds. Amsterdam: North-Holland.

Bubenko, J.A. 1993. Extending the Scope of Information Modelling. In Proceedings of the 4th Int. Workshop on the Deductive Approach to Information Systems and Databases. Lloret: Universitat Politecnica de Catalunya.

Gilardoni, L. (1993) COBALT Deliverable 2 Addendum: Interface Between Component Parts, Report COBALT/QUI/14/93. Milano: Quinary SpA.

Kirikova, M., and Bubenko, J.A. 1994). Entreprise Modelling: Improving the Quality of Requirements Specifications. In Proceedings of the 1994 Information Systems Research Seminar in Scandinavia, IRIS-17. Oulu: Department of Information Processing Science of the University.

Kolodner, J.L. 1992. An Introduction to Case-Based Reasoning. *Artificial Intelligence Review* **6**: 3-34.

Lehmann, F., ed. 1992. *Semantic Networks in Artificial Intelligence*. Oxford: Pergamon Press.

Ogonowski, A. 1987. MENTAT: An Intelligent and Cooperative Natural Language DB Interface. In Proceedings of the 7th Avignon International Workshop on Expert Systems and Their Applications. Nanterre: EC2.

Sowa, J.F. 1984. *Conceptual Structures: Information Processing in Mind and Machine*. Reading, Mass.: Addison-Wesley.

Wess, S., Althoff, K.-D., and Richter, M.M., eds. 1994. *Topics in Case-Based Reasoning* (Lectures Notes in Artificial Intelligence 837). Berlin: Springer-Verlag.

Zarri, G.P. 1986. The Use of Inference Mechanisms to Improve the Retrieval Facilities from Large Relational Databases. In Proceedings of the Ninth International ACM Conference on Research and Development in Information Retrieval, Rabitti, F., ed. New York: ACM.

Zarri, G.P. 1992. Encoding the Temporal Characteristics of the Natural Language Descriptions of (Legal) Situations. In *Expert Systems in Law*, Martino, A., ed. Amsterdam: Elsevier Science Publishers.

Zarri, G.P. 1992. The "Descriptive" Component of a Hybrid Knowledge Representation Language. In *Semantic Networks in Artificial Intelligence*, Lehmann, F., ed. Oxford: Pergamon Press.

Zarri, G.P. 1994. A Glimpse of NKRL, the "Narrative Knowledge Representation Language". In Knowledge Representation for Natural Language Processing in Implemented Systems - Papers from the 1994 Fall Symposium, Ali, S., ed. Menlo Park, Calif: AAAI Press.

Zarri, G.P. 1995. An Overview of the Structural Principles Underpinning the Construction of "Ontologies" in NKRL. In Proceedings of the IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Ottawa: Department of Computer Science of the University.

Zarri, G.P. 1995. Knowledge Acquisition from Complex Narrative Texts Using the NKRL Technology. In Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Calgary: Department of Computer Science of the University.

Zarri, G.P., and Gilardoni, L. 1996). Structuring and Retrieval of the Complex Predicate Arguments Proper to the NKRL Conceptual Language. In Foundations of Intelligent Systems - Proceedings of 9th International Symposium on Methodologies for Intelligent Systems, ISMIS'96. Berlin: Springer-Verlag.