A GLIMPSE OF NKRL, THE "NARRATIVE KNOWLEDGE REPRESENTATION LANGUAGE"

Gian Piero Zarri
Centre National de la Recherche Scientifique
EHESS - CAMS
54, boulevard Raspail
75270 PARIS Cedex 06, France
phone : (33) (1) 49.54.20.33
fax : (33) (1) 49.54.21.09

Abstract

In this paper, I describe NKRL, a language expressly designed for representing, in a standardised way, the semantic content (the "meaning") of complex narrative texts. After having introduced the four "components" (specialised sub-languages) of NKRL, I will give some examples of its practical modalities of use. I will then describe, in a very sketchy way, the inference techniques and the natural language processing procedures associated with this language.

1. Introduction

NKRL (Narrative Knowledge Representation Language) has been created to provide a "normalised" description of the semantic content (the "meaning") of complex narrative texts : the resulting NKRL code must avoid, as much as possible, any too important loss of the original information. In an industrial context, "narrative texts" may correspond, e.g., to news stories, telex reports or intelligence messages. NKRL has been developed by the author at CNRS (the French National Centre for Scientific Research) over a long period. It is a fully implemented language ; the most recent implementations have been realised in the framework of two European projects : NOMOS ("Knowledge Acquisition for Normative Reasoning Systems"), Esprit P5330, and COBALT ("Construction of Knowledge Bases from Natural Language Documents"), LRE ("Linguistic Research and Engineering") P61011.

Because of its (chiefly) practical aims, NKRL does not intend to present itself as a comprehensive modelling framework for Natural Language (NL) Understanding — like, e.g., Episodic Logic [Schubert89], [Hwang93] — nor as a "semantically clean" formal model of some specific NL phenomenon, like, e.g., the UNO model [Iwanska92], [Iwanska93]. However, NKRL has worked out some original solutions to typical knowledge representation problems (as, e.g., representing the "wishes, desires, intention" domain) : this can make this language of some interest also from a general "representation theory" point of view.

2. The general framework of the NKRL language

In NKRL — which presents some (loose) similarities with the standard hybrid and terminological languages in the KL-ONE style, see, e.g., [Brachman91] — we make use of four neatly differentiated but interrelated components (sub-languages), each of which employs distinct, specialised representation languages which call, in turn, for their own, proper inference systems ; see, e.g., [Zarri93] for more details.

- The "descriptive component" concerns the representation of the "meaning" of NL clauses describing some general classes of real-world events. In the context of the descriptive component, the events taken into consideration must be structured events, i.e., characterised by the explicit indication of an actor, an object, an instrument, etc. Examples of such general classes may be "moving a physical object", "formulate a need", "having a negative attitude towards someone", "be present somewhere", "come in possession of new resources", etc. The formal NKRL representations of those NL expressions are called "templates". Of course, the formal realisation of each template is independent from the surface structures of the various possible NL utterances which can be used to describe the corresponding classes of events.

- The "factual component" gives the formal representation of the NL narrative expressions relating some specific events — characterised, at least implicitly, by specific spatial and temporal co-ordinates — which constitute the concrete instantiations of the general classes of the descriptive component. This component therefore handles such NL expressions as: "Tomorrow, I will move the wardrobe", "Lucy was looking for a taxi", "Mr. Smith has fired Mr. Brown", "Peter lives in Paris", "Company X, located in Geneva, has taken the control of Company Y", etc. The NKRL expressions of these narrative statements are called "predicative occurrences".

- The "definitional component" handles the formal representation of the main defining properties of all the general notions (in parts specific of an application domain) which can be used in the framework of the descriptive and factual components. The corresponding NKRL data structures are called "concepts" ; concepts will appear *italicised* throughout this paper. The definitional tools are thus used to represent, in a conceptual form, the essential properties of general entities like "*physical_object*", "*taxi_*" (the general class including all the taxis, not a specific cab), or "*resources_*".

- The "enumerative component" handles the formal representation of the instances (concrete examples) of the general notions (concepts) belonging to the definitional component. The NKRL formal representations of such instances are called "individuals". Individuals are thus created by instantiating some of the concepts properties from the definitional component. Individuals are characterised by their being countable and possessing unique conceptual labels ("smith_", "general_motors", "taxi_27"). Two individuals associated with the same NKRL description but with different labels are different individuals.

The data structures supporting the four components are highly homogeneous, given that templates, occurrences, classes and individuals are all implemented as "structured objects" identified by an "OID" (object identifier). On the one hand, the definitional and enumerative data structures which support the "concepts" and the "individuals" are built in a "frame"-like fashion, i.e., as bundles of "attribute / value" relations where neither the number nor the order of the attributes is fixed. On the other hand, the structured objects of the descriptive and factual components ("templates" and "occurrences") are built, in a "case grammar" ([Bruce75], [SparkJones87], etc.) style, around a central and unique "semantic predicate", whose "arguments" are introduced by "roles" as SUBJ(ect), OBJ(ect), SOURCE, etc. For example, the elementary occurrence (factual component) translating the NL sentence "John gives a book to Mary" includes a semantic predicate corresponding roughly to "obtaining, without any *a priori* connotation of mandatory or permanent possession" ; "Mary" will be the SUBJ of "obtaining", "book" the OBJ and "John" the SOURCE, see the next Sections for more complete examples. Please note that a (single) semantic predicate, (at least) a role, and (at least) an argument are necessary to give rise to a well-formed template or occurrence, which are, therefore, meaningless in the absence of one of these three basic components.

The descriptive and the definitional components are both implemented as hierarchies of structured objects (templates and concepts, respectively). The NKRL descriptive component is, therefore, mapped to a H_TEMP(lates) hierarchy. The hierarchy of concepts (definitional component) — called H_CLASS for historical reasons — admits multiple inheritance and is, more precisely, a DAG, Directed Acyclic Graph. For example, a concept like "*tax_*" pertains ("is-a") to two different H_CLASS sub-hierarchies, "*fiscal_law_concepts*" and "*financial_resources*". The OIDs of the concepts making up the upper levels of H_CLASS — such as "*human_being_or_social_body*", "(unstructured) *event_*", "*attribute_value*", "*property_*", "*process_*", "*physical_entity*", "*abstract_entity*", "*modality_*", "*location_*", etc. — are, as those forming the upper levels of H_TEMP (the "basic templates", see next Section), invariant. This means that the upper levels of H_CLASS do not change when another application in a different domain is taken into account. Please note that one of the sub-trees of "*attribute_value*" is the "*logical_quantifier*" sub-tree, whose branches pertain to the set {*few_*, *some_*, *several_*, *many_*, *all_*, *all_except*}. In NKRL, quantifiers are ordinary concepts defined by a frame structure, which can be parametrised according to the needs of a particular application.

3. Using NKRL in a socio-economico-political context

The NKRL structures introduced until now are very general tools having a descriptive power which can be compared with that of semantic networks or Sowa's conceptual graphs, [Sowa84], [Sowa91]. As this last class of tools, they lack, however, of the specific constructs and primitives which must be added whenever a well defined conceptual domain must be considered. This absence normally produces a proliferation of *ad hoc* solutions which are different from case to case, and gives rise to serious difficulties to secure the reproduction or the sharing of previous results.

For NKRL, if the context happens to be, as is the case in NOMOS or COBALT, a (very general) socio-economico-political one where the main characters are human beings or social bodies, experience has shown that it is possible to cover the domain, in the templates and in the occurrences of the descriptive and factual components, with a total of seven semantic predicates (see Table 1), which correspond to very general, prototypical categories of human attitudes.

Predicate	**Mnemonic Description**
BEHAVE	a character adopts a particular attitude, or acts in order to obtain a particular result.
EXIST	to be present, also metaphorically, in a certain place.
EXPERIENCE	a character is affected by some sort of good, bad or "neutral" news or events.
MOVE	the displacement of a person or a physical object, the transmission of a message ...
OWN	to have, to hold, to possess...
PRODUCE	cause to exist or occur, with respect to material or immaterial entities, like the production of a service.
RECEIVE	to acquire, to obtain, without any connotation of mandatory or permanent possession.

Table 1 - Semantic predicates in NKRL.

In such a case, thanks to the reduced number of basic semantic predicates, all the legal, descriptive and factual structures (templates and occurrences), with the practical use which is made of them, can be fully described in a "catalog" (e.g., the "Stouder's catalog" [Stouder87]). This allows these structures to be used according to a coherent strategy which can be reproduced and shared. Further NKRL descriptive and factual structures can, if necessary, be easily derived from those already described in the "catalog", even in domains which are far from the original socio-economico-political one.

3.1 An example of coding

Let us now consider a concrete, simple example of NKRL coding, see, Fig. 1, the representation of the information :"Sharp Corporation said it has shifted production of low value personal computers from Japan to companies in Taiwan and Korea (Tokyo, March 31, 1993)".

```
1) MOVE SUBJ  sharp_corp : tokyo_
        OBJ   #2
        date1:31_march_1993
        date2:

2) MOVE SUBJ  sharp_corp
        OBJ   (SPECIF production_1
                 low_value_pc_1):
                 (japan_ (taiwan_
                 korea_))
        DEST  (SPECIF company_1
                 (SPECIF cardinality_
                 several_)):
                 (taiwan_ korea_)
        date1:before_31_march_1993
        date2:
```

Figure 1 - An example of NKRL coding.

The two occurrences "1" and "2" are instances of two different, permanent "basic templates" described in the catalog, which both belong to the "MOVE" (see Table 1) sub-tree of the general H_TEMP hierarchy (descriptive component). The MOVE template at the origin of the occurrence "1" is systematically used to translate the "transmission of an information" ("Sharp Corp said ..."). It makes use of what, in NKRL's terminology, is called a "completive construction". According to this construction, the filler of the OBJ(ect) slot in the occurrences (here, "1") which instantiate the "transmission" template, is an OID (object identifier, here #2) which refers to another predicative occurrence, i.e., the occurrence which bears the informational content to be spread out ("... it has shifted ...").
Occurrence "2" is an instance of another basic template which makes use of a different "MOVE" construction to represent a specialisation of the meaning "displacement of a (generic) object". To make it clearer, we reproduce this template in Fig. 2. Optional elements are in round brackets (e.g., the role SOURCE introduces the possible "instigator" of the displacement). In the corresponding occurrences, the variables (*x*, *y*, *v*, etc.) are replaced by concepts (definitional component) or individuals (enumerative component) according to

the associated constraints ; constraints are, in turn, expressed as combinations of high-level elements (concepts) of the H_CLASS hierarchy.

```
{MOVE3.21; 'move a method or process'

IS_A : MOVE3.2; 'move a generic
  object'

  MOVE SUBJ    x :[<location_>]
       OBJ     y :[{l1, 'initial
                     location' l2,
                     'final location'}]
       (SOURCE z :[<location_>])
       (DEST   u :[<location_>])
       (MODAL  v)
       (EVENT  w)
       ({'modulators'})
       [date-1:<date_>, 'initial
                date'|'observed date']
       [date-2:<date_>, 'final date']

  x  =  <human_being_or_social_body>
  y  =  <method_> | <process_>  | ...
  z  =  <human_being_or_social_body>
  u  =  <human_being_or_social_body>
  v  =  <displacement_modality>
  w  =  <situation_framework>
  l1, l2  =  <location_> }
```

Figure 2 - An example of a template.

In the NKRL "external" notation, the "location attributes" are associated with the role fillers by using the "colon", ":", operator, see Fig. 1 and 2. Generally speaking, a location attribute is coded by using a list, see [Zarri93] for more details. We can also mention that particular attention has been paid, in a factual component context, to a detailed and algorithmically efficient NKRL representation of temporal information : a recent paper on this subject is [Zarri92b].

3.2 The AECS sub-language

In Fig. 1, the arguments made up of a "SPECIF(ication)" list are examples of NKRL "structured arguments" (or "expansions").

Structured arguments of NKRL templates and occurrences are built up in a principled way by making use of a specialised sub-language, AECS, which includes four binding operators, the "disjunctive operator" (ALTERNative = A), the "distributive operator" (ENUMeration = E), the "collective operator" (COORDination = C), and the "attributive operator" (SPECIFication = S), see, e.g., [Zarri92a], [Gilardoni93], [Zarri93]. Their meaning is given in Table 2. Accordingly, structured arguments in NKRL are lists of undefined length, which may include both concepts and individuals and which are labelled by making use of the AECS operators.

Operator	**Mnemonic Description**
ALTERN	The "disjunctive operator". It introduces a set of elements, i.e., concepts, individuals or lists labelled with different binding operators. Only one element of the set takes part in the particular relationship with the predicate defined by the role-slot to be filled with the expansion ; however, this element is not known.
COORD	The "collective operator" : all the elements of the set participate 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 is used to associate a series of attributes (properties) with the concept or individual that constitutes the first element of the SPECIF list, in order to better characterise this last element. Each attribute appearing inside a SPECIF list can be recursively associated with another SPECIF list.

Table 2 - NKRL operators for structured arguments.

The AECS sub-language is, undoubtedly, powerful and expressive ; however, because of its recursive nature, it could give rise to very complex expressions, difficult to interpret and disentangle (unify). Therefore, in order to build up well-formed NKRL expansions, the definitions of Table 2 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. According to this rule, the event represented by the narrative NL expression :"Mr. Brown attended the reception at the White House ; Mr. Smith also was there, accompanied by his daughter Jennifer (but the accompanying lady could also be his wife Lucy)", will be translated in NKRL according to the representation of Fig. 3 (please note that Mr. Brown and Mr. Smith went separately to the reception, as shown by the use of ENUM). Of course, each of the single coding elements (Mr_Brown etc.) inside the expansions could be accompanied by a list of attributes (a "SPECIF" list). EVENT is the role expressing the notion of "in the context of".

```
1) EXIST SUBJ  (ALTERN (ENUM Mr_Brown
               (COORD Mr_Smith
               Jennifer_Smith))
               (ENUM Mr_Brown
               (COORD Mr_Smith
               Lucy_Smith))) :
               white_house
         EVENT reception_15 :
               white_house
```

Figure 3 - An example illustrating the "priority rule".

The definitions of Table 2 show clearly, moreover, that the AECS operators should intervene in the NKRL representation of plural entities and expressions. For example, COORD and ENUM correspond clearly to the "collective" and "distributive" plural quantifiers normally used, among other things, to represent the "reading variants" of plural noun phrases, see, e.g., [Webber83], [Sowa91], [Franconi93]. Without entering now into too many details, and referring again to Fig. 1 above, I will say here that we make use in general of a SPECIF sub-list introduced by "*cardinality_*" — a concept pertaining to the "*property_*" sub-tree of the H_CLASS hierarchy — to represent implicitly or explicitly, in a descriptive/factual component context, the number of elements in a given set. As it appears clearly from occurrence 2 of Fig. 1, "*several_*" — a concept belonging to the "*logical_quantifier*" sub-tree of H_CLASS, see, *supra*, Section 2. — is now used inside a "*cardinality_*" list to provide a standard way of symbolising the generic "plural number" mark for templates and occurrences. To complete the NKRL representation of a general, plural NL expression ("companies" in the example of Fig. 1) we must also activate a "*has_member*" slot in the frame associated with the individual "company_1" to denote that this last coincides, in reality, with a set ; the slot will be filled, once again, with the concept "*several_*" because of the indefiniteness of the plural expression. We would insert the numeral "12" in place of "*several_*", in both the occurrence "2" of Fig. 1 and the "*has_member*" slot of "company_1", in case of a news speaking of a transfer to "twelve companies".

3.3 Additional properties of NKRL

We give, in Fig. 4, another example of NKRL coding, which concerns the representation of the narrative sentence : "We have to make orange juice". According to Hwang and Schubert [Hwang93 : 1298], this sentence exemplifies several interesting semantic phenomena.

```
1) BEHAVE  SUBJ (COORD informant_1
                (SPECIF human_being
                (SPECIF cardinality_
                several_)))
                [oblig, ment]
                date1:(observed date)
                date2:

2) PRODUCE SUBJ (COORD informant_1
                (SPECIF human_being
                (SPECIF cardinality_
                several_)))
           OBJ  (SPECIF amount_1
                orange_juice)
                date1:(observed date + η)
                date2:

3) (GOAL 1 2)
```

Figure 4 - Predicative and binding occurrences.

The representation of Fig. 4 illustrates some important properties of NKRL. The first concerns the standard way of expressing the "wishes, desires, intention" domain. To translate the idea of "acting in order to obtain a given result", we make use of a BEHAVE (see Table 1 above) occurrence to express the "acting" component — i.e., to identify the SUBJ(ect) of the action, the temporal co-ordinates, possibly the MODAL(ity) or the instigator (SOURCE), etc. — and of a second occurrence structured around one of the six residual predicates to express the "intended result" component. The two occurrences are linked by means of a "binding occurrence" (see the NKRL structure labelled with "3" in Fig. 4). "Binding" structures (templates or occurrences) are, as the "completive construction" encountered before, a way to represent the logico-semantic links which can exist between descriptive or factual structures ; GOAL is an operator pertaining to the NKRL's "taxonomy of causality", see, e.g., [Zarri92a]. If the "modulator" (see below) "ment(al)" is absent, the SUBJ(ect) of BEHAVE takes some concrete initiative (acts explicitly) in order to fulfil the result ; if "ment(al)" is present, as in Fig. 4, no concrete action is undertaken, and the "result" reflects only the wishes and desires of the SUBJ(ect).

"Modulators" are, like the "location" and "temporal" determiners (attributes), NKRL operators used to refine or modify the primary interpretation of a template or occurrence as given by the basic "predicate — roles — argument" association. "ment" pertains to the "modality" modulators ; a second modulator, "oblig(atory)" appears in the occurrence "1" of Fig. 4. "oblig" suggests that "someone is obliged to do or to endure something, e.g., by authority", and pertains, as "fac(ulty)", "interd(iction)" and "perm(ission)", to the "deontic modulators" series. Other modulators are the "temporal modulators", "begin", "end", "obs(erve)", see [Zarri93] for a complete description. In NKRL, modulators work as global operators which take as their argument the whole (predicative) template or occurrence (predicate, roles, arguments, other determiners, ...). When a list of modulators is present, as in occurrence "1" of Fig. 4, they apply successively to the template/occurrence in a polish notation style, in order to avoid any possibility of scope ambiguity.

4. Inference techniques

As already stated, each one of the four components of the language is characterised by the association with a particular class of inference procedures.

The main inference mechanism associated with the definitional and enumerative components is the usual taxonomic inheritance mechanism which proceeds over "is-a" and "instance-of" relations. The basic building blocks for this mechanism are normally offered for free by the knowledge representation tools which support the NKRL's implementations, e.g., CRL ("Carnegie Representation Language") in the context of the NOMOS project, QSL ("Quinary Semantic Language") in the COBALT project. Please note that the subtyping relations to be used for the inheritance operations — which, in the Iwanska's UNO representation system, are given by sets of type equations, see, e.g., [Iwanska92 : 360] — are directly encoded, in NKRL, in the H_CLASS hierarchy supporting the definitional component.

The key inference mechanism of the factual 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", i.e., data structures including, at least, a predicate, a predicative role with its associated (structured) argument and, possibly, the indication of the temporal interval where the unification holds. The search patterns may originate from outside the application system if, in a deductive retriever style, they represent a direct translation of queries posed by the user ; on the other hand, they may be automatically generated by the system, within an application program, during the execution of all sorts of high-level inference procedures, see, e.g., [Zarri86].

An important element of the FUM module is represented by the matching algorithms which operate on the information stored in the structured arguments. They allow for a maximum of flexibility. It is possible, e.g., to ask for i) a "perfect match", defined as a match that succeeds if and only if the argument sections of the search pattern and of the target NKRL expression are perfectly identical (excepting variables), i.e., if their tree representations coincide exactly ; ii) a perfect match apart from "cardinality", i.e., a match that succeeds if and only if the arguments of the pattern and the target NKRL expression have the same identical structure left alone variables and the cardinality of the AECS lists ; iii) a "subsumed" match, i.e., a match that succeeds if and only if the arguments of the pattern and of the target expression carry a congruent (e.g., from a "subtyping" point of view) semantic information; iv) a combination of the above possibilities. The two special operators STRICT-CARDINALITY and STRICT-SUBSUMPTION, see [Gilardoni93] can, therefore, be used within the argument sections of the search patterns. In general, checking a correspondence between the AECS lists of a search pattern and of another NKRL data structure can be executed in polynomial time thanks, mainly, to the use of the "priority rule" proper to the AECS sub-language and mentioned, *supra*, in Section 3.2. Please note, finally, that — in order to deal with the "subsumed" varieties of match — the FUM algorithm must be capable of making use of the taxonomic inheritance mechanisms linked with the definitional and enumerative components to deal directly, during the filtering/unification phases, with the generalisation vs. specialisation aspects of the

H_CLASS hierarchy (unification with inheritance, see [Aït-Kaci86]). Generally speaking, the logical organisation of the FUM module seems to be at least as powerful as that of the algorithms used to compute the "entailment relations" in [Iwanska92 : 364-366] and [Iwanska93 : 495-503].

A specialised inference mechanism, grounded on FUM and proper to the descriptive component, is a sort of "join" procedure (see [Sowa84], [Iwanska92], [Iwanska93], etc.) allowing for the merge of (two or more) basic or derived (specialised) templates. Templates can merge only if they belong to the same general H_TEMP sub-tree, i.e., if they are characterised by the same semantic predicate. Information on the characteristics of this merge procedure can be found, e.g., in [Zarri92c : 190-191].

An example of "high level inference procedures" which can be implemented by making use of the above basic mechanisms — and which are of a particular interest from a "knowledge bases management" point of view — are the "transformation rules", see [Zarri86], [Ogonowski87]. Transformations deal with the problem of obtaining a "plausible answer" from an NKRL factual database also in the absence of the explicitly requested information, by searching semantic affinities between what is requested and what is really present in the base. The fundamental principle employed is then to "transform" the original query into one or more different queries, which are "semantically close" to the original one. To give a very simple example, suppose that, working in the context of an hypothetical NKRL 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.

Without entering now in too many formal details, we can say that transformations are made up of a "left hand side" — formulation in a template format of the linguistic expression which is to be transformed — and one or more "right hand sides" — representation in the same style of one or more linguistic expressions that must be substituted for the given one. A transformation can, therefore, be expressed as : "A (left hand side) → B (right hand side)". The "transformation arrow", "→", has a double meaning :

- operationally speaking, the arrow indicates the direction of the transformation : the left hand side A is deleted and replaced by the right hand side B ;
- 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 ⇒ A", where we assume that the symbol "⇒" represents the "implication arrow") are not very frequent. Most transformations found in real world applications represent "modalised implications" (noted as "B ◊⇒ A", which means "it is possible that B implies A"). An example of this last type of transformations is given by the transformation "t1" in Fig. 5, which allows us to deal, by using the FUM module, with the informal example above about "university professors" ; as we can see, the left and hand right sides of "t1" are formed by basic templates. 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 always possible to change t1 into a "true" transformation by the addition of a few constraints on the variable *p* , for instance the "disequation" : "*p* ≠ <obtainable_by_correspondence_degree >". More examples, and a complete "theory" of transformations, can be found in, e.g., [Zarri86].

```
t1) EXIST SUBJ x:[k] → RECEIVE SUBJ x
                                OBJ    p
                                SOURCE q:[k]
                                MODAL  r

    x = <human_being>
    p = <title_>
    q = <authority_>
    k = <location_>
    r = <official_document>
```

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

5. The NL/NKRL "translation" system

The NOMOS and COBALT projects — notwithstanding some differences between the two concerning the final use of the resulting NKRL code — have both successfully dealt with the problem of computing the NKRL expressions corresponding to an input NL text. The present "NL/NKRL translation system", see [Zarri92c] for more details, is based on the use of independent "specialists" — for the morphological and syntactical analysis, and the different phases of the semantic procedures — which make use of common shared, permanent data structures.

We reserve two permanent data structures to the H_CLASS and H_TEMP hierarchies. The most important shared structure is, however, the "lexicon", which is a complex mechanism entailing that each NL lexical entry contains morpho-syntactic, semantic and pragmatic information. The lexical entries represent, mainly, "rootforms" (infinitive for the verbs, etc.) ; each entry is a structured object including nine multivalued slots ("morphology of", "morphological-features"; "syntactic-category", "semantic-features", etc.) ; the entire lexicon is a hierarchy having the object "word" as root. The three slots "ref-scenario", "ref-template", "ref-concept" may contain pointers towards another category of conceptual data structures, i.e., the "conceptual" dictionaries (which include mainly pragmatic information).

A first conceptual dictionary concerns the "scenario references" : it contains the "text grammar" rules allowing the quick focusing on a particular information to be extracted while ignoring irrelevant elements, see, e.g., [Jacobs90]. Scenario references have not been implemented in NOMOS. In COBALT, we use a commercial product, TCS (Text Categorisation System", Carnegie Group, Pittsburgh, U.S.A.) to generate a first-level broad classification of the texts ; TCS employs the standard H_CLASS hierarchy (concept hierarchy, definitional component) of NKRL instead of its own set of "concepts". The *a priori* interesting texts are then analysed in depth, using the usual NL/NKRL techniques, to produce a "second level" categorisation (content classification) of these texts. The "template references" dictionary contains the "triggering rules" evoked by the lexical entry examined, i.e., rules allowing the activation of predicative templates of the H_TEMP hierarchy (descriptive component). The triggering rules are a sort of production rules. The left hand side (antecedent part) is a "syntactic condition", expressed under the form of a tree-like structure, which must be unified with the results of the general parse tree produced by the syntactic specialist. If the unification succeeds, the right hand sides (consequent parts) are used to generate well-formed templates. The last conceptual dictionary is the "conceptual references" dictionary, where we store the control structures ("substitution rules", similar to the "triggering rules") allowing the replacement, in the original NL text, of an NP (noun phrase group, NL domain), or part of NP, with the corresponding H_CLASS concept(s) or individual(s) (NKRL domain).

With respect now to the specialists, the output of the first specialist, the pre-processing specialist, is a set of distinct sentences, to be processed separately, which have already been submitted to the morphological analysis. In NOMOS, the syntactic analysis of each sentence is performed by a robust ATN compiler (syntactic specialist) making use of a "*Grammaire Française de Surface*", GSF ; syntactic ambiguity is handled i) by using "heuristics", see [Hobbs90] ; ii) by interacting with the user through a friendly interface. In COBALT, the ATN compiler has been replaced by a sort of unification-based "categorial grammar". The specialist called "separation into morpho-syntactic contexts" has been introduced for the purpose of modularity, robustness and simplicity. It makes use of morpho-syntactic rules which are linked, e.g., with the presence of relative clauses, and which it applies, for each sentence, to the results of the syntactic analysis. For example, in a NL fragment like: "In order to determine the income tax payable by companies which are under the authority of, or which exercise a control over, companies domiciled abroad ...", a segmentation mark is introduced in correspondence of the first "which", and the fragment is split into two different morpho-syntactic contexts which are processed separately.

The substitution procedures performed by the noun-phrases specialist constitute one of the fundamental steps of the translation process. These procedures concern the "leaves" (NL terms) of the syntactic tree which belong to the context being considered and which are part of a noun phrase (NP). For each of them, we must examine the "substitution rules" stored in the corresponding entry of the concept references dictionary (see before) in order to see if these NL leaves (or associations of NL leaves) can be substituted, on the parse tree, with the corresponding concepts (or individuals) characteristic of the NKRL definitional (enumerative) component. After this phase, the formulation (in linear form) of the "Sharp Corporation" NL fragment given, *supra*, at the beginning of Section 3.1, becomes "sharp_corp **said it has shifted** production_1 **of** low_value_pc_1 **from** japan_ **to** (SPECIF company_1 (SPECIF *cardinality_ several_*)) **in** taiwan_ **and** korea_", where "sharp-corp", "production_1", "company_1", "*cardinality_*", etc. are NKRL terms (concepts or individuals).

The triggering, filling and merging procedures (specialisation procedures), successively carried out by the generation specialist, represent the core of the "translation engine". They are applied in sequence to the contexts — already processed by the noun-phrases specialist — generated by the sentences of the original NL text. For example, in the last formulation of the "Sharp Corporation" fragment (which has by now become a mixture of NL and NKRL terms), the lexical entries corresponding to the NL terms "said" and "shifted" are effectively characterised by their association with particular triggering rules stored in the "template references" conceptual dictionary, see before. The candidate triggering rules are then tested. If their syntactic conditions (left-hand side parts or antecedents of the rules) successfully match the results of the general syntactic specialist, the corresponding templates (right-hand side parts or consequents of the rules) can be activated. The values retrieved on the syntactic parse tree by the left-hand variables during the match operations can then be used in order to adapt these templates to the particular needs of the NL text.

As an example, we reproduce in Fig. 6 one of the several triggering rules to which the lexical entry "shift" contain a pointer. This rule allows us to instantiate the template MOVE3.21 of Fig. 2 ("move a method or process") that will give rise, at a later stage, to the occurrence "2" of Fig. 1, see Section 3.1. In this example, the antecedent of the triggering rule effectively unifies the result of the syntactic analysis, thus authorising the instantiation of the template mentioned in the consequent. Moreover, its "left-hand side variables", "*xn*", can "capture" on the parse tree, during the unification phase, NL or H_CLASS terms which can then be used as "specialisation terms" for the subsequent filling operations, see [Zarri92c] for more details. The generic H_TEMP template of the consequent can then be "specialised" to the specific needs of the NL text examined. Please note that the actual template is not stored in the consequent, given that the full H_TEMP hierarchy is part of the "common shared data structures" mentioned before. Only the parameters relating to the specific triggering rule are, therefore, stored in the consequent : in Fig. 6, e.g., the "+" symbol indicates the roles of the original H_TEMP template which become here "mandatory" ; the list "constr" actualises the constraints on the variables.

```
trigger "shift" (NL):

syntactic condition :
(s (subj (np (noun/pronoun x1))) (vcl
(voice active) (v t = shift)) (dir-
obj (np (noun x2) (modifiers (pp
(prep of | in | … ) (np (noun
x3))))))) (place-adv-phrase (pp (prep
from) (np (noun x4)))) (co-pp (pp
(prep to)(np (noun x5)))/(adv-loc
x5)))

parameters for the template :
(MOVE3.21 (roles (+subj x1 +obj
(specif x2 x3):(x4) +dest x5))
(constr (x2 (process_) (x3 (goods_)
(x4 (location_))))
```

Figure 6 - An example of triggering rule.

The merge of "semi-identical" templates is the last procedure executed by the "generation" specialist. It is based on a very complex "unification" processes ; see [Zarri92c] for some details. The last specialist of the NL/NKRL translation system (references specialist) deals with the anaphoric phenomena (e.g., the resolution of "it" in the Sharp Corporation fragment) and the calculation of the "inter-context" logical and semantic links.

5. Conclusion

As already stated, the NOMOS version of NKRL is built on top of CRL, and the COBALT version on top of QSL. This means that the CRL and QSL tools supply all the basic building primitives, independent from the domain, to be used in order to support the definitional and descriptive structures of NKRL.

More precisely, given that the nucleus of NKRL's definitional and enumerative components is a quasi-standard frame-like language, see, *supra*, Section 2., the functions needed in order to create, verify and update the data structures proper to these components have been directly implemented by making use of the CRL and QSL tools. On the contrary, the implementation of the equivalent functions proper to the descriptive component has required, previously, an accurate definition, in CRL/QSL terms, of the conceptual descriptive structures, mainly under the form of a hierarchy of types. This type hierarchy includes basic types (predicates, roles, arguments, modifiers ...) and constructed types (mainly structured arguments, templates and occurrences), see [Cardelli85]. Type declarations present each new type under the form of a CRL/QSL "schema" or "object", where the CRL/QSL ordinary tools (creation of slots, restrictions concerning the domain, the range, and the cardinality, demons, metaknowledge etc.) are used in order to define the characteristic behaviour of the type — e.g., a demon is used to check that the succession of the AECS operators (ALTERN etc.) in a "structured argument" (see, *supra*, Section 3.2) follows the "priority rule" proper to AECS sub-language. Types are then used in order to set up the templates of the H_TEMP hierarchy and their instances (occurrences).

References

[Aït-Kaci86] Aït-Kaci, H., and Nasr, R. (1986) "LOGIN : A Logic Programming Language with Built-in Inheritance", The Journal of Logic Programming, **3**, 185-215.

[Brachman91] Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A., and Borgida, A. (1991) "Living with CLASSIC : When and How to Use a KL-ONE-Like Language", in Principles of Semantic Networks, Sowa, J.F., ed. San Mateo (CA): Morgan Kaufmann.

[Bruce75] Bruce, B. (1975) "Case Systems for Natural Language", Artificial Intelligence, **6**, 327-360.

[Cardelli85] Cardelli, L., and Wegner, P. (1985) "On Understanding Types, Data Abstraction, and Polymorphism", ACM Computing Surveys, **17**, 471-522.

[Franconi93] Franconi, E. (1993) "A Treatment of Plurals and Plural Quantifications Based on a Theory of Collections", Minds and Machines, **3**, 453-474.

[Gilardoni93] Gilardoni, L. (1993) COBALT Deliverable 2 Addendum : Interface Between Component Parts (Technical Report COBALT/QUI/14/93). Milano: Quinary SpA.

[Hobbs90] Hobbs, J.R., and Bear, J. (1990) "Two Principles of Parse Preference", in Proceedings of the 13th International Conference on Computational Linguistics - COLING 90, Karlgren, H., ed. Helsinki: University Press.

[Hwang93] Hwang, C.H. (1993) "Meeting the Interlocking Needs of LF-Computation, Deindexing and Inference : An Organic Approach to General NLU", in Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence - IJCAI/93. San Mateo (CA): Morgan Kaufmann.

[Iwanska92] Iwanska, L. (1992) "A General Semantic Model of Negation in Natural Language : Representation and Inference", in Proceedings of the Third International Conference On Principles of Knowledge Representation and Reasoning - KR'92, Nebel, B., Rich, C., and Swartout, W., eds. San Mateo (CA): Morgan Kaufmann.

[Iwanska93] Iwanska, L. (1993) "Logical Reasoning in Natural Language : It Is All about Knowledge", Minds and Machines, **3**, 475-510.

[Jacobs90] Jacobs, P.S., and Rau, L.F.(1990) "SCISOR: Extracting Information from On-line News", Communications of the ACM, **33**(11), 88-97.

[Ogonowski87] 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, vol. 2. Nanterre: EC2.

[Schubert89] Schubert, L.K., and Hwang, C.H. (1989) "An Episodic Knowledge Representation for Narrative Texts", in Proceedings of the First International Conference On Principles of Knowledge Representation and Reasoning - KR'89, Brachman, R.J., Levesque, H.J., and Reiter, R., eds. San Mateo (CA): Morgan Kaufmann.

[Sowa84] Sowa, J.F. (1984) Conceptual Structures : Information Processing in Mind and Machine. Reading (MA): Addison-Wesley.

[Sowa91] [Sowa91] Sowa, J.F. (1991) "Toward the Expressive Power of Natural Language", in Principles of Semantic Networks, Sowa, J.F., ed. San Mateo (CA): Morgan Kaufmann.

[SparkJones87] Sparck Jones, K., and Boguraev, B. (1987) "A Note on a Study of Cases", Computational Linguistics, **13**, 65-68.

[Stouder87] Stouder, L. (1987) RESEDA, le métalangage (Conv. INALCO/CIMSA SINTRA n° 0223A). Vélizy: Division CIMSA SINTRA de Thomson-CSF.

[Webber83] Webber, B.L. (1983) "So What Can We Talk About Now?", in Computational Models of Discourse, Brady, M., and Berwick, R.C., eds. Cambridge (MA): The MIT Press.

[Zarri86] 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 (NY): ACM.

[Zarri92a] Zarri, G.P. (1992) "The 'Descriptive' Component of a Hybrid Knowledge Representation Language", Computers and Mathematics with Applications - Special Issue on Semantic Networks in Artificial Intelligence (Part 2), Lehmann, F., ed., **23**, 697-718.

[Zarri92b] 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.

[Zarri92c] Zarri, G.P. (1992) Zarri, G.P. "Semantic Modeling of the Content of (Normative) Natural Language Documents", in Avignon '92 - Proceedings of the Specialized Conference on Natural Language Processing and Its Applications. Nanterre: EC2.

[Zarri93] Zarri, G.P. (1993) NKRL : A General Survey (Technical Report COBALT/QUI/9/93). Milano: Quinary SpA.