CONCEPTUAL REPRESENTATION FOR KNOWLEDGE BASES AND "INTELLIGENT" INFORMATION RETRIEVAL SYSTEMS

Gian Piero ZARRI

Centre National de la Recherche Scientifique
CERTAL - INALCO
2, rue de Lille
75007 Paris, France

Abstract

This paper describes the "conceptual" Knowledge Representation Language (KRL) proper to an environment for the construction and use of large Knowledge Bases and/or "Intelligent" Information Retrieval Systems. In the KRL, we separate the treatment of the **episodic memory** (**extensional, assertional data** = "Snoopy is Charlie Brown's beagle") from the treatment of the **semantic memory** (**intensional, terminological data** = "A beagle is a sort of hound / a hound is a dog ...). A compromise between an "object-oriented approach" and a "logic-oriented approach" is proposed for implementation purposes.

1. Introduction

This paper describes the "**conceptual**" Knowledge Representation Language (KRL) which is at the core of a complete environment for the development and use of large **Knowledge Bases** (KB) and/or **Intelligent Information Retrieval Systems** (IIRS). The environment is built up in the framework of a current project carried on at the French National Center for Scientific Research (CNRS) with the support of several French public and private bodies. The main characteristic of the storing and retrieving systems which will be produced using this environment is that (almost) all the knowledge to be inserted into these systems, concerning both the **proper data** ("fact database") and the **inference procedures** ("rule base") is coded making use of this conceptual language, a high-level representation language in the Artificial Intelligence (AI) style. This, in particular, ensures a **high level of consistency** between the fact and the rule component of the knowledge bases : these components now have the same expressive and computational power.

Besides the formal conceptual language, which will be described in the next sections, the environment is endowed with several classes of tools permitting the acquisition, storing, retrieving and manipulation of knowledge represented according to this language. For example :

◊ In what concerns **knowledge acquisition**, we have assumed that the most reasonable way to express information to be entered into the **assertional** component (episodic memory) of the "**fact database**" is **natural language** : the environment must,

therefore, be provided with acquisition tools authorizing the transfer, in a higly automated style, from a description in natural language of an incoming piece of information to the internal representation mode, see [ZARR88]. For the **"rule base"** components, it seems more appropriate to have recourse to a wide spectrum of possible instruments : these can range from simple rule base **editors** and **debuggers**, to **advisory** systems (which help the expert to conceptualize the rule base and to structure and refine the rule set), to some simple realizations of **learning from example** (e.g. inducing conceptual rules, by using meta-knowledge, from the events stored in the assertional component of the fact database) and **learning from analogy** (e.g. deducing rules from other rules) techniques.

◊ Querying the knowledge bases is performed in **natural language** using an adaptation of the module permitting the translation from natural language into the internal representation ; the answers of the system will also be expressed in natural language using the generation module employed to check the quality of the translation.

◊ In what concerns the inference procedures, we distinguish between **standard inference procedures**, a necessary component of every knowledge based tool built using the proposed environment, and **optional inference procedures**, permitting the specialization of the KB/IIRS towards particular classes of intelligent utilization of the stored data. An example of the "standard" querying inferential tools are the **"transformations"** [ZARR86], which can be seen as a necessary and powerful extension of the ordinary match procedures between a formal query (a **search pattern**) and the complex data stored in the fact database .Examples of "optional" inference tools may concern the dynamic construction of conceptual networks, permitting to evidentiate, for example, **"causal"** relationships among distinct data subsets [ZARR84, ZARR85] ; **"planning"**, i.e. making use of retrieved data in order to fill the skeleton plans recalled from a store of standard situations describing how actors achieve goals, as in the **"script-based planning"** [WILE83] ; etc.

◊ Inference procedures may be executed according to several strategies, ranging from "classical" inference engines to extensions of the blackboard models, as in the CASSANDRA architecture defined by I.D. Craig at the University of Lancaster [CRAI87].

2. The knowledge representation problem : conceptual level and implementation level

In a KB/IIRS building context, the problem of knowledge representation involves two different aspects, a **conceptual aspect** and an **implementation** aspect.

◊ **At the conceptual level**, we must define a description language permetting an "optimal", formal representation of the information, in the broadest acceptance of the term, to be introduced into the system. If we assume, for simplicity's sake, that this information was originally expressed in natural language (and, in reality, it is always possible to revert to this situation), we can gauge this optimality with respect to (an adaptation of) the canonical criteria enounced by [SCHA74] :

- the representation must be unique and unambiguous ;
- the representation should not lose any of the fundamental piece of information contained in the original natural language definition ;
- the representation should emphasize the role of the different semantic components of the original information in order to facilitate, in particular, any possible inference to be performed later (for example, to discover facts implicit in the stored facts) ;

- if the original information is complex, the representation should not only focus on the description of the different fragments, but also guarantee the coherence of the whole.

◊ **At the implementation level,** this formal description language must be mapped onto the knowledge representation tools normally used in an AI context (frames, semantic networks, object-oriented representations, logic programming etc.) ; we must emphasize the fact that these tools are nothing but "**implementation tools**", which cannot guarantee *per se* the power and the completeness of the conceptual representation. In what concerns some "**static**", **passive** aspects of the KRL, such as the terminological hierarchies, see 3.2 below, a good candidate as an implementation vehicle could be a sort of "**object-oriented representation**", see [GOLD83, STEF86], given the importance of the inheritance properties. On the other hand, as our KRL stands in the field of **purely declarative languages**, a "**logic-oriented**" computational framework also presents a certain number of advantages, because of its relatively well understood semantics and its integrated inference mechanism. The two points of view are well reconciliable, as some recent work has demonstrated, see [GORD85], etc. In our case, it is fundamental that the implementation language is able to handle naturally and efficiently the cooperation of **assertional** (see 3.1) and **terminological** (see 3.2) knowledge, the two main concepts of our knowledge representation scheme. In this context, the KRL implementation will be centered around the realization of a clausal theorem-prover with a **powerful unification algorithm**, capable of dealing **directly** with terminological hierarchies, see [ZARR85] and, in a logic programming context, [AITK86].

In the following sections, I will describe the solutions retained for the definition of the conceptual description language. The discussion will be illustrated with references to the particular subset of this language which is temporarily used in the project for experimentation's sake and which is represented by the "metalanguage" of RESEDA, an IIRS working in the domain of complex "biographical" (in the broadest acceptance of the term) data [ZARR84, ZARR85].

3. An outline of the conceptual description language

For clarity's sake, I will utilize here, as a **practical** tool the classic distinction between **episodic** and **semantic memory.** In this way :

◊ I will refer to "**episodic memory**" (or **assertional component, extensional data**) as that part of the knowledge to be stored in the KB/IIRS which concerns the description of detailed, particular facts abouts individual things, characters and events. It includes such facts as : "A flight of Alia, the Jordanian national company, has been hijacked this morning at 7.10 am GMT on leaving from Beyrouth airport" or "Snoopy is Charlie Brown's beagle".

◊ The "**semantic memory**" (or **terminological component, intensional data**), on the contrary, concerns universal principles. It includes such general information as : "Hijack is to commandeer a flying airplane, for example by coercing the pilot at gunpoint" ; "a beagle is a sort of hound / a hound is a dog of any of various hunting breeds / all dogs are animals".

Semantic memory corresponds to dictionary definitions, and episodic memory to history *and biographical data (in the broadest acceptance of the terms), see [BRAC85c] about the* reason to distinguish carefully between these two components of the KRL. Their representation inside the knowledge bases requires the use of tools which are, at least partially, different ; I will, therefore, separate their treatment.

3.1 Episodic memory

Brachmann and his collegues propose, for the assertional component of KRYPTON [BRAC85c : 421-422], a solution in the form of a language structured compositionally like a **first order predicate calculus language**, where the sentence-forming operators are the usual ones : **Not, Or, ThereExists**, and so on. We think that the genericity of this approach may lead to the use of very heavy and complex constructions in order to accurately represent intricate, real situations. In this context, we prefere to stick to the results stemming from the "**conceptual analysis**" tradition, although we are well conscious of the dangers of *ad hoc* solutions which characterize this type of approach. As well known, this tradition goes back to the origins of AI, see [SCHA73, WILK75], etc., and owes much of its recent revival to the success of Sowa's book [SOWA84].

Therefore, the fundamental assumption which, in our KRL, characterizes the representation of data pertaining to the episodic memory is the choice to use some sort of "**deep coding**". This means that the representation of a particular **elementary event** is independent of the surface structure of the linguistic utterances which express this event : for example, the three clauses : "John gives a book to Mary" ; "Mary gets a book from John" ; "A book is given to Mary by John" show different surface structures but must correspond to the same deep coding.

3.11 Predicative conceptual units and predicative occurrences

The coding of an **elementary event** takes the appearence of a **particular** realization of a "**predicative conceptual unit**", i.e. is organized around a **semantic predicate** (something corresponding to "gain possession after a transfer" in the above example concerning Mary, John and the book) identifying the "basic" type of action, state, situation etc. which is described in the event. The **concrete entities** (or, more exactly, the codes denoting these entities) which are mentioned in the event (John, Mary, book) and which are, at least partly, proper to a particular application domain (e.g. human personages, public and private bodies, offices, governmental authorities, diplomatic missions, etc. in a socio-political context) fill the peculiar **roles** (slots, facets, cases) associated with the semantic predicate. In the example, "Mary" is the "subject" of "gaining possession", "book" the "object" and "John" the "source". According to a "case grammar" approach [FILL68, SOWA84, etc.] the entities are, therefore, the **arguments** of the semantic predicate which are introduced by means of the "roles" (case slots). The resulting code takes the name of "**predicative occurrence**" : a predicative occurrence is, therefore, a predicative conceptual unit adapted to the formal representation of a particular elementary event.

The (unique) "semantic predicate", the "roles" and the "arguments" are the basic elements which make up a predicative conceptual unit (a predicative occurrence) ; the predicate, the arguments and the conceptual unit (the occurrence) as a whole may be characterized by "**determiners**" (attributes) which give details about significant aspects of these elements. For example, the predicate may be accompanied by "**modulators**" [ZARR84] that, as their name suggests, are there to refine or modify the original semantic meaning of the predicate : an obvious example is the "negation" modulator ("neg") which allows the happening of an elementary event to be denied ("Mary did not get a book from John"). Determiners which are associated with the arguments are the "**location attributes**" ("Mary, in Pensacola, got a book from John, in Kansas City"). Determiners which refer to the global predicative occurrence are the "**temporal attributes**", which quantify time duration or frequency of the elementary event, or the "**identification attributes**", giving information about the original source of the event related in the predicative occurrence, the creation date of this specific piece of code ; etc. In RESEDA, a particular class of determiners which can be attached either to the single entities appearing inside the arguments of the predicate or to the whole occurrence are the "**validity attributes**", which can take values as "true", "conjectural", "uncertain", "contradictory", "false", etc., see below the example of Fig. 5 in 3.121.

A general scheme of a predicative occurrence is, therefore, shown in Fig. 1 ; I will now give some details about the main elements of this figure.

```
{ PREDICATE : [ modulators ]
              ROLE - 1   { < argument > : [ location ] }
              ROLE - 2   { < argument > : [ location ] }
              .
              :
              ROLE - n   { < argument > : [ location ] } }
              .
              :
              [ temporal attributes ]
              [ identification attributes ]
              .
              :
```

Figure 1

3.111 Semantic predicate

A discussion about the choice of a "correct" set of predicates shifts quickly to a discussion about the problem of the **"semantic primitives"**. It is well known, for example, that Schank utilizes, in the first descriptions of his **conceptual dependency** theory [SCHA73], a reduced set of eleven **primitive acts** which play the role of the "semantic predicate" in our KRL. In this way, the elementary event : "X walked to the cafeteria" is represented roughly as : "X (the "actor") PTRANS ("physical transfer", the primitive act) X (the "object") to the cafeteria (the "directive case")". The main advantage of this type of behaviour is, of course, the ability to dramatically reduce the capricious variety of the surface (natural language) denotations of the events to be represented ; the difficulty lies in the impossibility to reduce, in an unambiguous way, all the universe's complexity to eleven primitive acts. In their most recent work, Schank and his colleagues rely more and more on high-level concepts, like "DISPUTE", "PETITION", "AUTHORIZE" etc., see [SCHA79], instead of expanding everything into primitives.

I agree with [SOWA84 : 14] when he says that : "In general, a system should allow high-level concepts to be expanded in terms of lower ones, but such expansions should be optional, not obligatory". Accordingly, the predicate in Fig. 1 can be chosen in conformity to any "deep" or "surface" option. I would like to emphasize the fact that this choice must be a completely pragmatic one, depending on architectural considerations aimed to optimize a particular application (very often, in order to facilitate a particular set of useful inferences). For example, the metalanguage of the RESEDA system makes use of only five "semantic predicates" : BE-AFFECTED-BY (roughly to have, to be in possession of ...), BEHAVE (a person acts in order to obtain a particular result), BE-PRESENT, MOVE (the displacement of a person or a physical object, the transmission of a message ...), PRODUCE. This reduced set permits, in particular, the restriction of the number of basic legal **"predicative templates"** (predicative conceptual units describing classes of occurrences, see 3.22) admitted in the metalanguage to fifty (approximately), and is enough, for example, to contribute to the realization of an elegant and efficient indexing methodology for the fact database part of the system [ZARR83]. This reduction does not affect the semantic expressivity of the code obtained making use of the categories of the RESEDA's metalanguage thanks, mainly, to the following two reasons :

◊ The first one is that RESEDA makes an intensive use of the determiners of the "modulator" class ; these can also be combined, as in the association "recip(rocal), against" linked to the predicate "BEHAVE" in the occurrence of Fig. 2 to express the fact that Mr. Smith and Mr. Brown are engaged in a legal controversy. SUBJ(ect), OBJ(ect), and MODAL(ity) are roles ; the arguments introduced by the first two roles are complex ones ("expansions", see 3.112 and 3.113), realized via "COORD(ination) lists".

```
BEHAVE :   [ recip, against ]
           SUBJ      (COORD Smith Brown)
           OBJ       (COORD Smith Brown)
           MODAL     legal_action
```

Figure 2

◊ The second reason (and the most important one in the context of my argument) is that a fundamental, "architectural" choice of RESEDA has been to integrate, for efficiency's sake, the execution of the simplest inferences of the "generalization/specialization" type directly inside the unification module instead of using an inference engine, see again [ZARR85, AITK86]. These inferences, which make use of the inheritance properties, involve chiefly the elements of the "lexicon" (see 3.21) appearing in the "argument" sections of the predicative occurrences : hence the advantage of disposing of a lexicon organized under the form of a very rich conceptual taxonomy. As a consequence, whenever it was possible to translate a surface verb into a construction like that of Fig. 3, where the "action name" pertains obviously to some sub-tree of the lexicon, this type of solution has been systematically adopted. In this way, "to telephone" is translated as "PRODUCE phone_call",

---

```
PRODUCE    SUBJ   { <human_personage> | <social_body> | ... }
           OBJ    { <action_name> }
```

Figure 3

"to sail" as "PRODUCE departure_from_a_port", "to fire" as "PRODUCE dismissal", etc. ; the lexicon is considerably enriched and the number of predicates minimized without any loss of semantic content.

3.112 Roles

No truly convincing role classification has been proposed so far in the literature, see [SOWA84 : Appendix B] ; therefore, the choice of the "appropriate" set of roles is, once again, a purely pragmatic affair. RESEDA uses a total of 16 roles distributed into three classes :

◊ **"Predicative roles"**, used to link the predicate to its arguments inside a predicative conceptual unit. They are : SUBJ(ect), OBJ(ect), SOURCE, DEST(ination), MODAL(ity), TOPIC ("*à propos* of ..."), EVENT ("in the context of ...").

◊ **"Binding roles"**, ALTERN(ative), ASSOC(iation), COORD(ination), ENUM(eration), SPECIF(ication) ; all these roles are used as labels of lists of terms of undefined depth.

- ALTERN, COORD, ENUM and SPECIF can be used to construct **"structured arguments" ("expansions")** inside the predicative conceptual units. The first three correspond, roughly, to the logical connectives "exclusive or", "and", "inclusive or". More exactly, referring to the classification of [DAHL82], COORD corresponds to the "collective" relation (*all elements of a set participate in some relationship together*) ; ENUM to the "distributive" relation (each element of a set satisfies some relation, but they do so separately). ALTERN is the "disjunctive set" relation as defined by Sowa [SOWA84 : 118] : a set of elements of which one participates in a relationship, but the particular one is not known. The last role, SPECIF, is used to associate a series (a list) of attributes to a lexicon term appearing inside an argument of a predicative unit.

- ALTERN, COORD and ENUM can also be used to associate several predicative conceptual units (more exactly, the labels identifying these units) under the guise of undefined depth lists, giving rise to **"binding conceptual units"**, see 3.121, characterized by the absence of the predicate. ASSOC may equally be employed for this function ; more *precisely, it serves as a means of relating the predicative occurrences representing* the successive steps of the same complex event to the occurrence(s) **(founding event(s))** which is/are at the origin of the development. The role SPECIF cannot be used to construct binding conceptual units.

◊ **"Causal roles"**, CAUSE, CONFER, GOAL and MOTIV(ation). All these roles cannot be used in a predicative conceptual unit ; they are employed, on the contrary, to link predicative units inside the binding conceptual units. They define RESEDA's own causality taxonomy see, for example [ZARR84 : 207-208].

3.113 Arguments

The structuring of the conceptual entities proper to a particular application domain which appear inside the arguments of the predicate, see Fig. 1, pertain to the sphere of the **"semantic memory"**, and will thus be described later (3.21). At the level of the **"episodic memory"**, it would be necessary to say something about the syntax of the **"structured arguments" ("expansions")** which, in the particular case of the RESEDA's metalangage, are constructed by using the "binding roles" introduced in 3.112 in a "predicative occurrence" context. I will only say that, in order to simplify the search and unifying procedures, it may be useful to adopt the same syntactic rules for deal with the "expansions" inside a predicative occurrence (expansions permit the assembling of elementary conceptual entities from the domain) and with the "binding occurrences" which permit the assembling of predicative units. For example, in RESEDA, there is a strict symmetry between the treatment of the "SPECIF" lists (expansions) and of the "causal" lists (binding units) ; the treatment of the ALTERN, COORD, ENUM lists is, of course, the same inside and outside the predicative conceptual units.

3.114 Determiners

As said before, the "predicate", the "arguments" and the "predicative conceptual units" (predicative occurrences) as a whole can be characterized by the presence of "determiners" (attributes). These can be distributed in several classes, for example :

◊ **"Modulators"**, to be associated to the predicate. In the RESEDA's metalanguage, examples of modulators are : "nint" (nonintentional, indicating that the situation described in the occurrence is **not** the result of the precise willingness of the SUBJ(ect), as in "Mr. Smith has, involuntarily, started a fire") ; "krypt" (the activity of the SUBJ(ect) is hidden) ; "ment(al)" (the activities of the characters involved in the occurrence do not result in a

concrete manifestation in the physical domain) ; etc. Of paramount importance are the three "temporal modulators", "begin", "end" and "const" (from the French *constater* = to find, to recognize) which, associated with the "temporal attributes", see below, play a fundamental role in the coding of temporal information, see [ZARR83].

◊ **"Location attributes"**, normally associated with the entire argument section which fills one of the slots of the predicate. In the RESEDA's metalanguage, a location attribute is made up of three components :

- a complex, alpha-numerical "zip code-type" symbol, giving the best possible approximation of the spatial coordinates of the location to be represented ;
- a "typology" code, translating the "category" of the given location, as in "street", "school", "church", "farm", etc. ;
- a "toponym" code, giving the place-name of the location.

◊ **"Temporal attributes"**, normally associated with a whole conceptual unit : they are used to quantify time duration or frequency of a predicative occurrence. In the RESEDA system, the coding of temporal information is very important, given that the characteristics of this coding are used to structure accurately (**"indexing"**) the fact base part of the knowledge base, and to control the plausibility of the inference procedures of the system, see [ZARR83]. Without entering into too many details, I will only say that, fundamentally, the representation of RESEDA's temporal data is a representation in terms of "intervals" similar, in this respect, to James Allen's algebra of temporal intervals [ALLE83]. RESEDA's original concepts are the notions of **"category"** and **"perspective"**. The **"category of dating"** characterizes the association of a temporal marker to the beginning (**"posteriority"**, or **"subsequence"**), the end (**"anteriority"**, or **"precedence"**) or a particular moment (**"contemporaneity"**) of a given elementary event ; very often, this association is explicitely signalled by the presence of the "temporal modulators" mentioned before. The **"perspective of dating"** is used to define the degree of precision (for example, an incertitude expressed by a pair of dates) with which a given temporal marker is known.

3.12 Linking together the predicative conceptual units

Being able to represent the elementary events in predicative occurrences is not enough to translate the original information completely : it is, in fact, necessary to being able to represent also the logico-semantic links which exist between elementary events (the "coordination" and "subordination" links, using a metaphor from the domain of natural language).

3.121 Binding conceptual units and binding occurrences

One way to solve this problem is to use some **"binding conceptual units"**, i.e. a list - characterized using one of the **"binding roles"** introduced in 3.112 - whose elements are "labels" (addresses on the secondary memory) of predicative conceptual units. If the predicative conceptual units mentioned in the binding unit consist of predicative occurrences, the binding unit gives rise to a **"binding occurrence"**. As a very simple example, let us consider the Resedian representation of the information : "In the days following the11th of August 1408, King Charles VI sent the count of St. Pol to arrest Pierre d'Ailly at Cambrai". The coding will give rise to two predicative occurrences, "a" and "b", and to a binding occurrence, "c", see Fig. 4. Occurrences "a" and "b" have two (mandatory) temporal attributes, the two date blocks "date-1" and "date-2" which are used to register the dating elements giving the limits of the temporal interval associated with the occurrence. In the case

of the occurrence labelled as "a", only the first data block is filled because the situation described in this unit (St. Pol leaving Paris in order to go to Cambrai) may be represented as a "point" on the time axis. In the occurrence "b", the two blocks are empty because the apprehension of Pierre d'Ailly is not confirmed in the original information : the situation is interpreted as "conjectural", as signalled by the presence of the corresponding "conjectural validity attribute", code "^", i.e. presented as an "intention" of St. Pol.

---

a)	MOVE	SUBJ	Louis_de_St_Pol : [Paris]
		OBJ	Louis_de_St_Pol : [Cambrai]
		SOURCE	Charles_VI : [Paris]
		[date-1 :	after_11_august_1408]
		[date-2 :	]
b)	[*] PRODUCE	SUBJ	Louis_de_St_Pol : [Cambrai]
		OBJ	arrest
		DEST	Pierre_d'Ailly : [Cambrai]
		[date-1 :	]
		[date-2 :	]
c)	a (GOAL b)		

Figure 4

3.122 The completive construction

A second way of associating predicative conceptual units (predicative occurrences) is to insert unit labels inside the argument sections of the predicative units (**"completive construction"**). As an example, let us consider, Fig. 5, the Resedian coding of the information : "On September 3rd, 1967, the Sunday Times makes known to the western public opinion the fact that 329 Czechoslovakian intellectuals have prepared a declaration of

---

a)	MOVE	SUBJ	Sunday_Times : [London]
		OBJ	#b
		DEST	western_public_opinion
		MODAL	publication
		[date-1 :	3_september_1967]
		[date-2 :	]
b)	PRODUCE	SUBJ	intellectuals (SPECIF 329 czechoslovakia)
		OBJ	protest_manifest
		DEST	western_public_opinion
		[date-1 :	before_3_september_1967]
		[date-2 :	]

Figure 5

---

protest". In RESEDA, the particular completive construction where a "subordinate clause"(bearing the informational content to be spread out) is introduced as the OBJ(ect) of a type "MOVE" predicative occurrence is the privileged way of translating the "transmission of information".

3.2 *Semantic memory*

Semantic memory concerns that sort of general information which can be associated as a **"definition"** to :

◊ the **"entities"** (and their "generic" terms) corresponding to the **objects** and to the **concepts** of the application domain which appear in the elementary events of the episodic memory ;

◊ the **"occurrences"** (and their "generic" patterns) translating such events and the logico-semantic links which exist among them.

In both cases, the elements to be classified are inserted into a **"specialization hierarchy"** characterized by the inheritance of properties and behaviors : **H_ENT**, or **"lexicon"**, for the entities, **H_OCR**, or **"grammar"**, for the occurrences.

3.21 Lexicon

As already said, all the elementary entities which are necessary to the description of the application domain - e.g. terms like "arrest", "western_public_opinion", "intellectuals", "protest_declaration" in the units of Fig. 4 and 5 - are placed into a **hierarchical structure** (**"is_a"** hierarchy) which takes the name of **"lexicon" (H_ENT)** : conforming to the "universal" role of the "semantic memory", the lexicon is that part of a KB/IIRS which gives the general cognitive background of an application domain. According to the traditional terminology of object-oriented programming, the elements of the hierarchy can be divided into two major categories : **classes** (beagle) and **instances** (Snoopy). Classes take part directly in the inheritance process defined by the hierarchical structure ; instances participate indirectly through their classes making use of the **"instance_of"** links. For practical purposes, the "instance_of" link is often assimilated to the "is_a" link.

The elements of the lexicon - i.e. the "nodes" of the hierarchy - can be **"intensionally defined"** according to several strategies ranging over a scale of complexity, see 3.211 etc. They share, however, some general properties :

◊ All the nodes are characterized by an **"implicit type"** whose "label" is given by the name (the code) of the entity associated with the node. The node "beagle" defines the type "beagle" : all the specific terms depending from this node ("Snoopy" for example) are characterized by this type. This is, of course, a general property : if "hound" is the "generic" term which immediately precedes "beagle" in the hierarchy, the "supertype" of "beagle" is "hound" (i.e. "beagle" pertains to the type "hound") ; if "lexicon" is the node which represents the vertex of the hierarchy, all the nodes of this hierarchy pertain to the type "lexicon". Moreover, it may be useful to associate to the nodes of the lexicon an **"explicit type"** (a tag) permetting the rapid retrieval of all the nodes inserted in a particular sub-tree, e.g. sub-trees as "nominal_lexicon" or "qualifying_lexicon" : the advantages are obvious from the point of view of the simplification of coherence checks and, in particular, of being able to build a fast data type checking mechanism to speed up all the unification processes when querying the knowledge base.

◊ The use of **"multiple inheritance"** in the hierarchy increases the economy of expression. It makes it possible to avoid the duplication of information that would be necessary to introduce in a strictly hierarchical system in order to take into consideration the fact that an entity may have more than a parent-entity : see the case of "postage_stamps_collection" which is, at the same time, a "hobby" and an "investment". The lexicon tree becomes, in this case, an oriented graph with the "lexicon" node as top-level node. If multiple inheritance is used, some "precedence algorithm" must be provided to indicate the order in which the parent nodes are visited, for example : "starting from the entity under consideration, visit first the parent nodes of the left branch, and then those of the right branch, and then the join (the node at the origin of the different branches), and up from there" [STEF86 : 46-49].

◊ A problem which presents a certain degree of affinity with "multiple inheritance" is the problem of **"inheritance with exceptions"** : this problem has been studied extensively in an AI context, see the well-known examples "birds fly, penguins are birds, but penguins do not fly", or "Clyde is a royal elephant, a royal elephant is an elephant, elephants are gray, but Clyde (a royal elephant) is not gray" [BRAC85a]. This problem is particularly difficult to deal with, because the presence of exceptions introduces nonmonotonicity ; therefore, several well-known knowledge representation systems explicitly forbid the introduction of exceptions to inheritance, see KL-ONE [BRAC85b] for example. The normal way to solve this problem is to use some formal expression of an intuitive criterion (**"specialization principle"**) which states that subclasses should override superclasses. For example, [BREW87] utilizes a three-place "predicate" to express the information that an entity "x" is **"exceptional"** with respect to a "property" (see 3.212) of a (very general) parent entity if x is a specific term of a more specialized entity for which different information regarding this property is available ; see also [SAND86], etc.

3.211 Intensional definition using only the "is_a" links

Depending on the particular application, it is possible to derive the definition of some entities pertaining to the lexicon hierarchy simply by determining their position inside the oriented graph, i.e. by determining, following the "is_a" links, the ordered set of their "generic" terms. In this way, "beagle" is defined only by the fact of being a "specific" term of "hound", which is specific of "hunting_dog" etc. : all these terms pertain to the sub-graph which has "animal" as the top-level node. For these particular entities, the label associated with the corresponding node in the lexicon graph is reduced to the code which represents the "name" of the entity.

On the other side, the definition of entities which are of particular importance from the point of view of a given application cannot be given satisfactorily by this type of coding only, and must be **augmented** by a formal description of their outstanding characteristics. This description, coupled with the name of the entity, gives rise to a **"complex label"** which decorates the node corresponding to this entity : the arrangement of the lexicon under the basic form of an oriented graph connecting a set of labelled nodes by means of "is_a" links is not questioned at all. The formal description can be given under several forms.

3.212 Intensional definition by using "attributes"

The simplest way to describe the characteristics of an entity is to use some sort of **"attributes"** : the attributes represent significant aspects of the entity which can be quantified by means of simple values. Attributes are like fields in database records : for example, an entity (a class) "employee" could have the attributes "name", "age", "sex", "height", "weight" and "hobbies" (inherited from example from the parent-node, the class "person"), in addition to more specific attributes that we can define for this specific entity, as "job_title", "division" and "employee_number". Making an instance of the class "employee"

means filling in values associated with each one of these attributes. Several sorts of attributes can be used, for example :

◊ **"Descriptors"**, whose values very often consist of character strings. They include identifiers, proper names, titles and so on.

◊ **"Counters"** are used for both cardinal and ordinal quantities, and then their values consist of natural numbers. Cardinal number attributes quantify set cardinalities, while ordinal number attributes express position of members within an ordered set.

◊ **"Measure"** domains correspond to attributes quantified by couples "(value, unit of measure)". For instance, measure domains are "length", "area", and "weight". Values in measure domains are commonly expressed by real numbers.

"Attributes" correspond to **"instance variables"** in an object-oriented programming context. According to this paradigm, we can define the class "employee" by declaring, first, some **"class variable"** whose values are shared by all the instances of the class (the name of the company, for example) ; we introduce, then, the nine "instance variables" (which correspond to the nine attributes listed before), along with their default values. Particular values, e.g. (name John), (age 20), (hobbies skiing), are substituted for the default values in the particular instances of the class.

3.213 Intensional definition by using "attributes" and "procedures"

A complete characterization of an entity can be obtained if the description of the typical **behavior** of the entity (**"procedures"**) is added to the description of his properties (**"attributes"**). For the "employee" entity, typical procedures can be "hiring_procedures", "firing_procedures", "social_insurance_calculation", etc. As the attributes, procedures can be inherited from the parent-nodes : the parallelism with **"methods"** of object-oriented programming is, once again, well evident. Procedures can be implemented in a "procedural" style as, for example, COMMON LISP functions, or in a "declarative" manner. In this last case, the procedures are described by using some **"predicative templates"** (the general schemata subsuming the particular predicative occurrences, see 3.22) linked by **"binding templates"**, and present some similarities with the SCRIPT and the MOPS by Roger Schank [SCHA80].

3.22 Grammar

All the "abstract" conceptual units (**"templates"**, or patterns, schemata etc.) which describe the general properties of the "concrete" conceptual units (**"occurrences"**, see the examples in Fig. 4 and 5) introduced in 3.1, are placed in their own specialization hierarchy (**H_OCR**, or **grammar**) ; the concrete occurrences constitute the "leaves" of this hierarchy. As usual, the nodes of the hierarchy inherit some properties (predicates, roles, classes of lexicon terms, etc.) from their parent-nodes. The **instances** of the classes (templates) which occupy the nodes in the H_OCR oriented graph, i.e. the predicative and binding (concrete) occurrences, will not use any **quantified variables**, to the extent that this type of information represents assertions of specific facts or events (extensional data or episodic memory), see again the examples in Fig. 4 and 5. The conceptual units represented by the "templates" (intensional data or semantic memory), on the contrary, require variables, because of the need to describe large classes of occurrences.

For example, the predicative occurrence "a" of Fig. 4 is an instance of the template in Fig. 6, which gives the general framework of the displacement of a person or a group of persons (e.g. a delegation) ; the parent-node of this template is the general **displacement_template**. Optional elements are in parentheses. Note, in Fig. 6, the presence

of "variables" (x , y , v , etc.) : in the occurrences, these variables will be substituted by some instances of the entities pertaining to the lexicon hierarchy which are explicitely mentioned in the "**constraints**" associated with the variables (for simplicity's sake, the variables referring to the "descriptors" have been indicated only in an implicit form, and the full details of their constraints have been omitted). Therefore, the constraints specify the legal set of values (to be chosen among the instances of the lexicon hierarchy) which can be substitued for each variable and delimit, in this way, the validity domain of the template ; of course, constraints may also be used to express relationships among variables ("**semantic integrity management**"). The association of the same variable x to the two roles SUBJ(ect) and OBJ(ect) corresponds to a characteristic of the RESEDA's metalanguage : the movements of a person (or group of persons) are always expressed in the form of a SUBJ(ect) who moves himself as an OBJ(ect).

---

person_displacement_template

```
MOVE      ( : [ < temporal_modulators > ] )
          SUBJ          x  :  [ < departure_location > ]
          OBJ           x  :  [ < arrival_location > ]
          ( SOURCE      y  :  [ < location > ] )
          ( DEST        z  :  [ < location > ] )
          ( MODAL       u  )
          ( EVENT       v  )
          [ date-1 :        < departure_date > ]
          [ date-2 :        < arrival_date > ]

x = < human_being > | < person_group >
y = < human_being > | < social_body >
z = < human_being > | < social_body >
u = < displacement_modality >
v = < situation_framework >
```

Figure 6

With respect to the traditional subdivision of the knowledge base of an AI system into a "**fact database**" part and a "**rule base**" part, we can say that the conceptual units realized under the form of "**occurrences**" constitute the main elements appearing in the "fact database" part of our KB/IIRS. "**Templates**", on the contrary, are massively used in order to express the inference procedures in the "rule base" part of these systems (the "methods" mentioned in 3.123 are only a possible example of these inference procedures). "Occurrences" and "templates" are **conceptual units** constructed according to the same formal model, the main difference being the presence or the absence of variables. As already said in the "Introduction", there is, therefore, a **strong logical coherence** between the two parts of the knowledge bases, which contain information having the same expressive and computational power.

4. Conclusion

In this paper, I have described the "conceptual" Knowledge Representation Language (KRL) proper to an environment for the construction and use of large Knowledge Bases (KB) and/or

Intelligent Information Retrieval Systems (IIRS). The coding of elementary events occurring in the real world (**episodic memory, assertional data**) is organized around a **"semantic predicate"**. The **"entities"** which are mentioned in the elementary event fill the **"roles"** (slots, facets, cases) associated to the predicate ; the entities are, therefore, the**"arguments"** of the semantic predicate. Roles include "subject", "direct object", "source", "destination", "modality", etc. The conceptual predicative units constructed in this way take the name of **"predicative occurrences"** ; the predicate, the arguments and the conceptual unit as a whole can be characterized by **"determiners"**, for instance "temporal determiners". Moreover, the predicative occurrences translating the elementary events can be associated under the form of "binding conceptual units" using logical, causal etc. relationships, giving rise to complex conceptual constructions (**"binding occurrences"**). The basic entities in the application domain are classified into a **specialization hierarchy** (**"lexicon"**). Thus, every entity is represented as the specialization of one or more parent-entities from which it inherits properties (attributes) and behaviors. Analogously, occurrences are inserted in their own specialized hierarchy (**"grammar"**), by referring them to "abstract conceptual units" (**"templates"**) which describe their general characteristics. The "lexicon" and the "grammar, together, set up the **semantic memory** (**terminological data**). A compromise between an "object-oriented approach" and a "logic-oriented approach" is proposed for implementation purposes.

References

[AITK86] AIT-KACI, H., and NASR, R. (1986) "LOGIN : A Logic Programming Language with Built-In Inheritance", *The Journal of Logic Programming* , **3**, 185-215.

[ALLE83] ALLEN, J.F. (1983) "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM* , **26**, 832-843.

[BRAC85a] BRACHMAN, R.J. (1985) "I Lied About the Trees", *AI Magazine*, **6**, n° 3, 80-93.

[BRAC85b] BRACHMAN, R.J., and SCHMOLZE, J. (1985) "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science* , **9**, 171-216.

[BRAC85c] BRACHMAN, R.J., FIKES, R.E., and LEVESQUE, H.J. (1985) "KRYPTON : A Functional Approach to Knowledge Representation", in *Readings in Knowledge Representation* , Brachman, R.J., and Levesque, H.J., eds. Los Altos: Morgan Kaufmann.

[BREW87] BREWKA, G. (1987) "The Logic of Inheritance in Frame Systems", in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence - IJCAI/87.* Los Altos: Morgan Kaufmann.

[CRAI87] CRAIG, I.D. (1987) *Decentralised Control in a Blackboard System* (PH. D. Thesis). Lancaster: University of Lancaster Department of Computing.

[DAHL82] DAHL, V. (1982) "On Database Systems Development Through Logic", *ACM Transactions on Database Systems* , **7**, 102-123.

[FILL68] FILLMORE, C.J. (1968) "The Case for Case", in *Universals in Linguistic Theory* , Bach, E., and Harms, R.T., eds. New York: Holt, Rinehart and Winston.

[GOLD83] GOLDBERG, A., and ROBSON, D. (1983) *SMALLTALK 80 : The Language and Its Implementations.* Reading (Mass.): Addison-Wesley.

[GORD85] GORDON, T.F. (1985) "Object-Oriented Predicate Logic and Its Role in Representing Legal Knowledge", in *Computer Logic and Legal Reasoning* , Walter, C., ed. St. Paul: West Publishing Company.

[SAND86] SANDEWALL, E. (1986) "Non-Monotonic Inference Rules for Multiple Inheritance with Exceptions", *Proceedings of the IEEE* , **74**, 1345-1353.

[SCHA73] SCHANK, R.C. (1973) "Identification of Conceptualizations Underlying Natural Language", in *Computer Models of Thought and Language*, Schank, R.C., and Colby, K.M. San Francisco: W.H. Freeman and Co.

[SCHA74] SCHANK, R.C. (1974) *Understanding Paragraphs* (Working Papers n° 6). Castagnola (Lugano): Fondazione Dalle Molle.

[SCHA79] SCHANK, R.C., and CARBONELL, J.G. Jr. (1979) "Re : The Gettysburg Address - Representing Social and Political Acts", in *Associative Network - Representation and Use of Knowledge by Computers* , N.V. Findler, ed. New York: Academic Press.

[SCHA80] SCHANK, R.C. (1980) "Language and Memory", *Cognitive Science*, **4**, 243-284.

[SOWA84] SOWA, J.F. (1984) *Conceptual Structures : Information Processing in Mind and Machine.* Reading (Mass.): Addison-Wesley.

[STEF86] STEFIK, M.J., and BOBROW, D.G. (1986) "Object Oriented Programming : Themes and Variations", *AI Magazine* , **6**, n° 4, 40-62.

[WILE83] WILENSKY, R. (1983) *Planning and Understanding - A Computational Approach to Human Reasoning.* Reading (Mass.): Addison-Wesley.

[WILK75] WILKS, Y. (1975) "An Intelligent Analyzer and Understander of English", *Communications of the ACM* , **18**, 264-274.

[ZARR83] ZARRI, G.P. (1983) "An Outline of the Representation and Use of Temporal Data in the RESEDA System", *Information Technology : Research and Development*, **2**, 89-108.

[ZARR84] ZARRI, G.P. (1984) "Expert Systems and Information Retrieval : An Experiment in the Domain of Biographical Data Management", in *Developments in Expert Systems* , Coombs, M.J., ed. London: Academic Press.

[ZARR85] ZARRI, G.P. (1985) "Inference Techniques for Intelligent Information Retrieval", in *Computer Logic and Legal Reasoning* , Walter, C., ed. St. Paul: West Publishing Company.

[ZARR86] ZARRI, G.P. (1986) "The Use of Inference Mechanisms to Improve the Retrieval Facilities from Large Relational Databases", in *Proceedings of the ACM Conference on Research and Development in Information Retrieval* , Rabitti, F., ed. New York: ACM.

[ZARR88] ZARRI, G.P. (1988) "Creating and Structuring a Knowledge Base for Storing and Intelligently Retrieving Information Found in Natural Language Messages", in *Proceedings of the EUROINFO '88 Conference - First European Conference on Information Technology for Organisational Systems.* Amsterdam: North-Holland.