

A Conceptual Model for Representing Narratives

G.P. Zarri¹

¹ Centre National pour la Recherche Scientifique (CNRS), 44 rue de l'Amiral Mouchez, 75014 Paris, France

1 Introduction

A considerable amount of important, 'economically relevant' information, is buried into the so-called 'narrative' documents: this is true, e.g., for most of the corporate knowledge documents (memos, policy statements, reports, minutes etc.), for the news stories, the normative and legal texts, many intelligence messages, etc., as well as for a significant fraction of the information stored on the Web. In these 'narratives', the main part of the information content consists in the natural language (NL) description of 'facts' or 'events' that relate the real or intended behaviour of some 'actors' (characters, personages, etc.): these try to attain a specific result, experience particular situations, manipulate some (concrete or abstract) materials, send or receive messages, buy, sell, deliver etc. Note that, in this sort of documents, the actors or personages are not necessarily human beings; we can have narrative documents concerning, e.g., the vicissitudes in the journey of a nuclear submarine (the 'actor', 'subject' or 'personage') or the various avatars in the life of a commercial product. Moreover, 'annotating' narrative multimedia documents with an NL description of their 'semantic content' — see the content of a photo that, verbalised, could be expressed as "Three nice girls are lying on the beach" — increases still more the scope of the narrative domain.

Being able to represent in a general, accurate, and effective way the semantic content of these narratives — i.e., its key 'meaning' — is then both conceptually relevant and economically important.

In this Chapter, we describe then the main properties of NKRL (Narrative Knowledge Representation Language), a language expressly designed for representing, in a standardised way, the semantic content of narrative documents, in particular, of those that are economically relevant. These documents can be original NL documents or, as already stated, they can result from an annotation activity concerning, in general, multimedia documents. Note that NKRL has been used as 'the' modelling knowledge representation language in European projects like Nomos (Esprit P5330), Cobalt (LRE P61011), Concerto (Esprit P29159) and in the current Euforbia (IAP, Internet Action Plan, P26505) and Parmenides (IST P39023) projects. This Chapter is organised as follows. We will supply in Section 2 the general theoretical framework of the language. Section 3 will present a first, simple example of NKRL coding; Section 4 will supply some additional information about more advanced features of the language. Section 5 will deal with a first, simple level of

inference procedures, while Section 7 will describe an example of high-level inference procedures. Section 8 will contain our final remarks.

2 General information about NKRL

NKRL makes use, in an integrated way, of many knowledge representation principles, where each of them can be considered as the best fitted tool for modelling a particular category of narrative phenomena. Traditionally, all the NKRL knowledge representation tools are structured into four connected ‘components’, the definitional, enumerative, descriptive and factual one.

The definitional component supplies the tools for representing the *concepts*, intended here as the ‘important notions’ that it is absolutely necessary to take into account in a given application domain. In NKRL, a concept is represented, substantially, as a frame-like data structure, i.e., as an n -ary association of triples ‘name-attribute-value’ that have a common ‘name’ component. This name corresponds to a symbolic label like *human_being*, *taxi_* (the general class referring to all the possible taxis, not a specific cab), *city_*, *chair_*, *gold_*, etc. NKRL concepts are inserted into a generalisation/specialisation hierarchy that, for historical reasons, is called H_CLASS(es), and which corresponds well to the usual ontologies of terms.

The minimal requirements concerning a frame-based language suitable for playing the role of definitional component in NKRL are expressed, e.g., in [1]. To simplify we can say that, in substance, any tool in the OIL-Lite style [2] could be sufficient to implement a hierarchy of concepts (H_CLASS) according to the NKRL suggestions. In the Concerto project [3], e.g., IBM Italy has created a set of definitional component tools (of H_CLASS tools) by combining the NKRL requirements expressed in [1] with a well known knowledge specification mechanism, the Open Knowledge Base Connectivity (OKBC), see [4]. This ‘flexibility’ of NKRL with respect to the practical implementation of H_CLASS derives from the fact that the main reasoning mechanisms of NKRL are associated, in reality, with the descriptive and factual components. The most frequent use of H_CLASS concerns, in fact, a simple application of its ‘subsumption’ properties (generic/specific relationships among concepts) for checking, e.g., the constraints associated with the NKRL ‘variables’, see below.

We can nevertheless note that NKRL imposes the compliance with some very strict principles when dealing with the arrangement of the ‘upper level’ of any well-formed H_CLASS hierarchy. In particular, a fundamental assumption concerns the differentiation between ‘notions (concepts) that can be instantiated directly into enumerable specimens’, like “chair” (a physical object) and ‘notions that cannot be instantiated directly into specimens’, like “gold” (a substance). The two high-level branches of H_CLASS take origin, therefore, from two concepts labelled as *sortal_concepts* and *non_sortal_concepts*. The specialisations of the former, like *chair_*, *city_* or *european_city* can have direct instances (*chair_27*, *paris_*), whereas the specialisations of the latter, like *gold_* or

colour_, can admit further specialisations, see *white_gold* or *red_*, but do not have direct instances. Figure 1 supplies a (very simplified) image of the upper level of the H_CLASS hierarchy used in the CONCERTO project; for a discussion about non-sortal concepts like *substance_* and *colour_* see again [1].

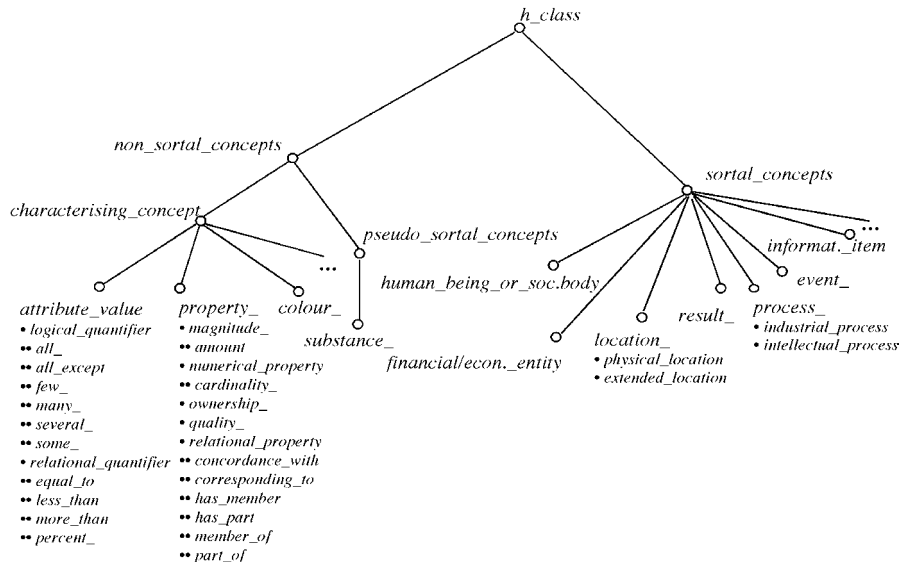


Figure 1. An abridged representation of the H_CLASS hierarchy used in the CONCERTO project.

The ‘enumerative component’ of NKRL concerns the tools used for representing formally, as (at least partially) instantiated frames, the concrete realisations (*lucy_*, *taxi_53*, *chair_27*, *paris_*) of the H_CLASS sortal concepts. In NKRL, the instances of sortal concepts take the name of *individuals*. Individuals are then countable and, like the concepts, possess unique symbolic labels (*lucy_* etc.). Throughout this paper, we will use the italic type style to represent a *concept_*, the roman style to represent an *individual_*.

Note that, notwithstanding definitional and enumerative structures are both implemented in NKRL as frame-based structures, their ontological status is totally different. Concepts (definitional component) define a generic, abstract mould for some useful notions of the domain: they are — at least in the context of a given application — permanent and universally valid. Individuals (enumerative component) represent, on the contrary, unpredictable, transient, randomly occurring entities whose existence is strictly dependent from the particular narrative documents examined. Unlike concepts, they are always associated with spatio-temporal co-ordinates, which sometimes refer to very hypothetical worlds; see, e.g., the individual *mauve_firespitting_dragon_57*. Note also that individuals do not have a proper hierarchical organisation — instances of individuals are not allowed in

NKRL, see [1]. They are however associated with the H_CLASS hierarchy, where they appear as the ‘leaves’ associated with the sortal concepts.

The definitional and enumerative tools can be assimilated, to a certain extent, with the traditional Tbox and Abox of description logics, see [5]. Passing on now to the representation of the ‘events’ proper to a given domain — i.e., to the coding of the *interactions* among the *particular concepts and individuals* that *play a role* in the contest of these events — more original tools appear, the ‘descriptive’ and ‘factual’ ones.

The descriptive component concerns the tools used to produce the formal representations, called ‘templates’, of some *general narrative classes*, like “moving a generic object”, “formulate a need”, “having a negative attitude towards someone”, “be present somewhere”, obtained by abstraction/generalisation from *sets of similar, elementary narrative events*. Note that, given the fuzziness that affects notions like those of event and of the related terms: state, situation, period, episode, history, process, action etc., we have proscribed the use of these locutions in NKRL for clarity’s sake, and we have unified them under a unique label, ‘event’. An NKRL elementary event is then the description of a particular set of interactions between concepts and/or individuals *that can be mapped to a single, specific template* and which is *temporally bounded*. This last characteristic implies that with every elementary event is then associated, at least implicitly, a proper, specific ‘duration’, which can also be ‘empty’ (point event). Therefore, in NKRL, items like “At 6.30 pm on December 15th, John had a serious automobile accident” (contracted episode), “John had several drinks during the day” (continuous action, process), “At that moment, John was intoxicated” (state), etc., are all ‘elementary events’ associated with a duration that can go from zero to several hours.

Returning now to the data structures proper to the descriptive component, i.e., to the *templates*, we can note that, in contrast to the traditional frame structures used for concepts and individuals, templates are characterised by the association of quadruples connecting together the *symbolic name* of the template, a *predicate* and the *arguments* of the predicate introduced by named relations, the *roles*. The quadruples have in common the ‘name’ and ‘predicate’ components. If we denote then with L_i the generic symbolic label identifying a given template, with P_j the predicate used in the template, with R_k the generic role and with a_k the corresponding argument, the NKRL data structures for templates have the following general format:

$$(L_i (P_j (R_1 a_1) (R_2 a_2) \dots (R_n a_n))) \quad (1)$$

see the example in Table 1, commented below. Presently, the predicates pertain to the set {BEHAVE, EXIST, EXPERIENCE, MOVE, OWN, PRODUCE, RECEIVE}, and the roles to the set {SUBJ(ect), OBJ(ect), SOURCE, BEN(e)F(iciary), MODAL(ity), TOPIC, CONTEXT}. Templates are included in an inheritance hierarchy, H_TEMP(lates), which corresponds to a new sort of ontology, an ‘ontology of events’.

The instances (called ‘predicative occurrences’) of the templates, i.e., the representation of single, specific elementary events like “Tomorrow, I will move the wardrobe” or “Lucy was looking for a taxi” are, eventually, in the domain of the

last component, the factual one. From the remarks introduced above about the temporal duration of the elementary events, it follows that each predicative occurrence should be normally associated with a temporal interval on the time axis delimited by two ‘timestamps’ $\langle t_1 t_2 \rangle$ such that $t_1 \leq t_2$ — the meaning of ‘ \leq ’ is here intuitive. A detailed description of temporal representation in NKRL can be found in [6].

3. A first example

To represent a fictitious elementary event in the Euforbia project style like “On July 5th, 2001, John Smith is appointed webmaster of the Ask Men internet site by the Ask Men management” — the Euforbia project, already mentioned in Section 1, deals with the conceptual filtering of questionable information on the Net — we must select firstly the template corresponding to ‘nominate to a post’, which is represented in the upper part of Table 1. The ‘position’ code shows the place of this ‘nomination’ template within the OWN branch (5.) of the H_TEMP hierarchy: this template is then a specialisation (see the ‘father’ code) of the particular OWN template that corresponds to ‘being in possession of a post’.

In the occurrences, the time interval $\langle t_1 t_2 \rangle$ is concretely rendered making use of two ‘temporal attributes’, `date-1` and `date-2`, that then define the time interval in which the ‘meaning’ represented by the predicative occurrence ‘holds’, see [6]. In the case of Table 1, the (mandatory) presence of a ‘temporal modulator’, ‘begin’ indicates that the only timestamp ($t_1 = \text{date-1}$) that can be associated with the predicative occurrence `c1` derived from the ‘nomination’ template corresponds to the beginning of the state of being in possession — here, to the nomination. In `c1`, the interval $\langle t_1 t_2 \rangle$ is then reduced to a point on the time axis, as indicated by the single value, ‘5-july-2001’ (the nomination date), associated with the attribute `date-1`. Attributes, like the temporal attributes (or the ‘location attributes’, see below), and modulators, like the temporal modulators, are part of a set of NKRL particular codes, the ‘determiners’. These can be associated with the arguments of the predicate, or the templates/occurrences as a whole, to supply further details with respect to the meaning supplied by the basic, well-formed descriptive/factual structure defined by the association predicate-roles-arguments.

The argument of the predicate (the a_k terms in formula (1) of the previous Section) are represented by variables with associated constraints; these last are expressed as concepts or combinations of concepts, i.e., making use of the terms of the H_CLASS hierarchy (definitional component). The ‘location attributes’, represented in the predicative occurrences as lists, are linked with the predicate arguments by using the colon operator, ‘:’. The constituents (SOURCE in Table 1) included in square brackets are optional; the symbol ‘(*)’ means: forbidden for a given template.

Table 1. Deriving a predicative occurrence from a template.

```

name: Own:NamingToPost
father: Own:BeingInPossessionOfPost
position: 5.1221
NL description: 'A Human Being is Appointed to a Post'

OWN   SUBJ      var1: [var2]
      OBJ       var3
      [SOURCE   var4: [var5]]
      (BENF)*
      [MODAL    var6]
      [TOPIC    var7]
      [CONTEXT  var8]
      { [ modulators ], begin }

      var1 = <human_being>
      var3 = <post_>
      var4 = <human_being_or_social_body>
      var6 = <action_name>
      var7 = <property_>
      var8 = <event_> | <action_name>
      var2, var5 = <physical_location>

c1)   OWN   SUBJ   john_smith
      OBJ   (SPECIF webmaster_ ask_men_internet_site)
      SOURCE (SPECIF management_team
              ask_men_internet_site)

      [begin]
      date-1:(5-july-2001)
      date-2:

```

The role fillers in the predicative occurrence *c1* represented in the lower part of Table 1 conform to the constraints of the father-template. For example, *john_smith* is an individual (enumerative component) instance of the sortal concept *individual_person* that is, in turn, a specialisation of *human_being*; *webmaster_* is a specialisation of *post_*, *management_team* is, obviously, a *social_body*, etc. — note that the filler of a SOURCE role always represents the ‘originating factor’ of the event. The ‘attributive operator’, SPECIF(ication), is one of the NKRL operators used to build up complex fillers or ‘expansions’, see next Section. The SPECIF lists, with syntax (SPECIF $e_1 p_1 \dots p_n$), are used to represent some of the properties or attributes which can be asserted about the first element e_1 , concept or individual, of the list — e.g., in *c1*, the attribute *ask_men_internet_site* (an individual) is associated with both *webmaster_* and *management_team*, to fully define these two last terms.

We can note here an important point. Unlike, e.g., canonical graphs in Sowa's conceptual graphs theory, see [7], which must be explicitly defined for each new application, the core of the H_TEMP hierarchy (about 150 templates) is fixed and fully defined. H_TEMP corresponds then to a sort of ‘catalogue’ of NKRL tem-

plates, and we can say that these templates are part and parcel of the definition of the language. An (extremely simplified) image of H_TEMP is given in Figure 2. Moreover, it is easy to ‘customise’ the existing templates, and to derive from them new templates that could be needed for a particular application. If they prove to be sufficiently general, they are then added to the ‘catalogue’. H_TEMP is then a continuously growing structure.

This approach is particularly advantageous for practical applications, and it implies, in particular, that: i) a system-builder does not have to create himself the structural knowledge needed to describe the events proper to a (sufficiently) large class of narrative documents; ii) it becomes easier to secure the reproduction or the sharing of previous results.

4. Additional properties of the NKRL language

The basic NKRL tools expounded in the previous Section are enhanced by the use of two additional classes of tools:

- the AECS ‘sub-language’, see [8], that allows the construction of complex (structured) predicate arguments, called ‘expansions’;
- the second order tools (binding structures and completive construction), see [6], used to encode the connectivity phenomena (logico-semantic links) that, in a narrative situation, can exist between single narrative fragments (corresponding to single NKRL predicative structures).

We will introduce informally the two new tools making use of additional examples of NKRL coding of narrative structures. Table 2 translates then this fragment of news story: “This morning, the spokesman said in a newspaper interview that, yesterday, his company has bought three factories abroad”. *today_* and *yesterday_* are two fictitious individuals introduced here, for simplicity’s sake, in place of real or approximate dates, see [6]. As already stated, the location attributes, represented as lists in the concrete occurrences, are linked with the arguments by using the colon operator, see *c3 (abroad_)*.

The ‘attributive operator’, *SPECIF(ication)*, already present in occurrence *c1* of Table 1 and that appears in both the occurrences *c2* and *c3*, is one of the four operators that make up the AECS sub language. AECS includes the disjunctive operator (*ALTERNative = A*), the distributive operator (*ENUMeration = E*), the collective operator (*COORDination = C*), and the attributive operator (*SPECIFICATION = S*).

The meaning of *ALTERN(ative)* is self-evident; the semantics of *SPECIF(ication)* has already been introduced informally in the previous Section.

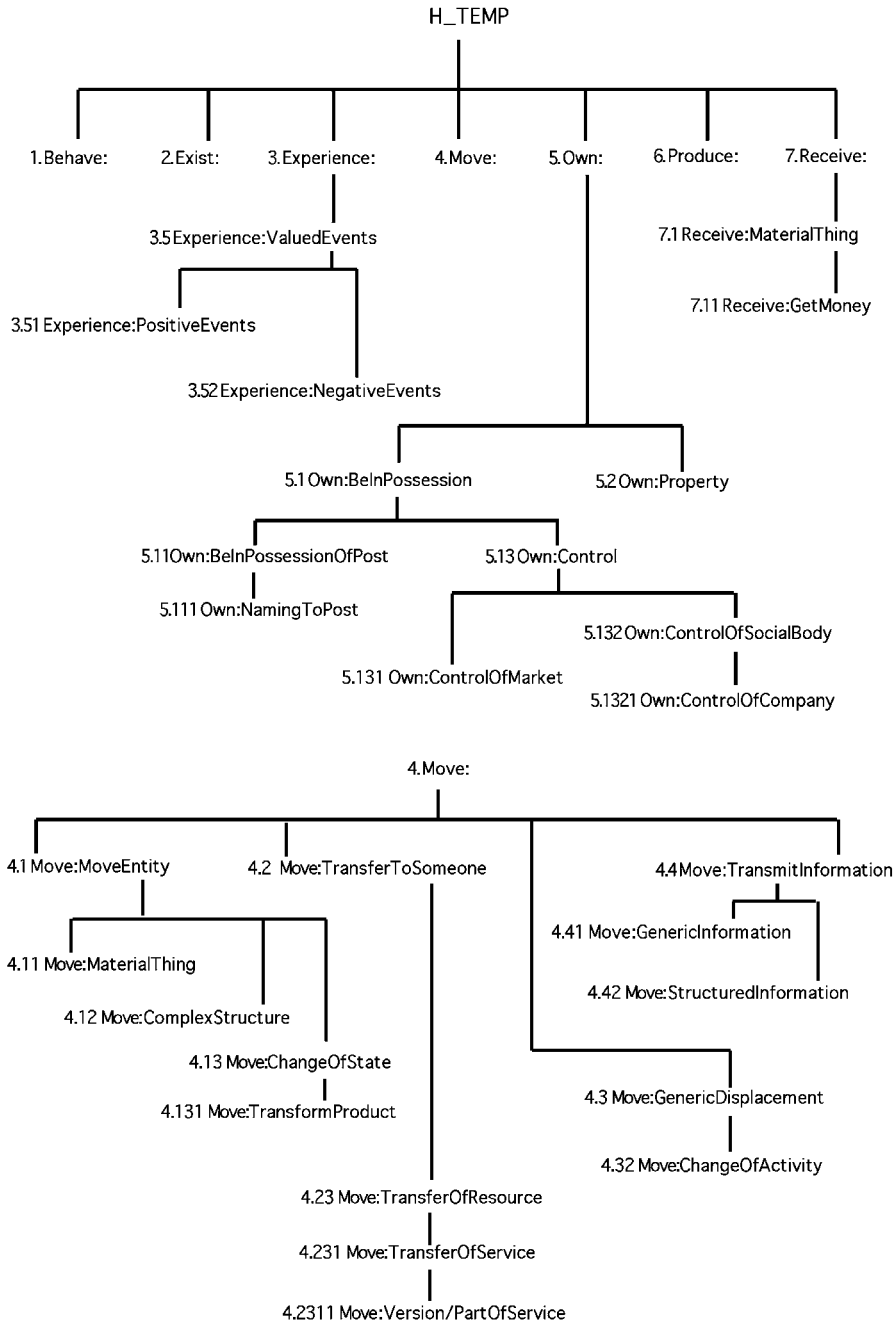


Figure 2. A (oversimplified) picture of the H_TEMP hierarchy (descriptive component).

Table 2. An example of completive construction.

c2)	MOVE	SUBJ	(SPECIF human_being_1 (SPECIF spokesman_ company_1))
	OBJ	#c3	
	BENF	newspaper_1	
	MODAL	interview_	
	date-1:	today_	
	date-2:		
c3)	PRODUCE	SUBJ	company_1
		OBJ	(SPECIF purchase_1 (SPECIF factory_99 (SPECIF cardinality_ 3))):(abroad_)
		date-1:	yesterday_
		date-2:	
[factory_99		
	InstanceOf:	factory_	
	HasMember :	3]	

In a COORD(ination) expansion list, all the elements of the expansion take part, *necessarily together*, in the particular relationship with the predicate defined by the role to be filled. This concomitance is not required for the elements of an ENUM(eration) list.

As an example, we can imagine a situation where the spokesman has communicated his information to two different newspapers, newspaper_1 and newspaper_2, the BEN(e)F(iciaries) — BENF is the role to be filled. If the two newspapers have assisted *together* to the interview, i.e., if they have received the information *together*, then the BENF slot of c2 will be filled with (COORD newspaper_1 newspaper_2). On the contrary, if the information was received *separately* — which could mean, in practice, that the newspapers have taken part in two different interviews — then the BENF filler would have been (ENUM newspaper_1 newspaper_2). ENUM can then be considered as a shortcut for reducing the size of a database of predicative occurrences by merging quite similar occurrences. For more information about the AECS operators and, in particular, about the rules for mixing them within the same expansion (‘priority rule’), see [1].

The last item of Table 2 supplies an example of enumerative data structure, explicitly associated with the individual factory_99 according to the rules for coding ‘plural situations’ in NKRL, see [1]. The non-empty HasMember slot in this structure makes it clear that the individual factory_99, as mentioned in c3, is referring in reality to several instances of factory_. Individuals like factory_99 are ‘collections’ rather than ‘sets’ (all the NKRL concepts can be interpreted as sets), given that the extensionality axiom (two sets are equal iff they have the same elements) does not hold here. In our framework, two collections, say factory_99 and factory_100, can be co-extensional, i.e., they can include exactly the same elements, without being necessarily considered as identical if created at different moments in time in the context of totally different event, see [9]. In Table 2, we have supposed that the three factories were, *a priori*, not sufficiently

important in the context of the news story to justify their explicit representation as specific individuals, e.g., `factory_1`, `factory_2`, `factory_3`. Note that, if not expressly required by the characteristics of the application, a basic NKRL principle suggests we should try to avoid any unnecessary proliferation of individuals.

In coding narrative information, one of the most difficult problems consists in being able to deal (at least partly) with the connectivity phenomena like causality, goal, indirect speech, co-ordination and subordination, etc. — in short, all those phenomena that, in a sequence of statements, cause the global meaning to go beyond the simple addition of the information conveyed by each single statement, see also the Conclusion. In NKRL, the connectivity phenomena are dealt with making a (limited) use of second order structures: these are obtained from a *reification* of the predicative occurrences that is based on the use of their symbolic labels — like `c2` and `c3` in Table 2.

A first example of second order structure is given by the so-called ‘completive construction’, which consists in accepting as filler of a role in a predicative occurrence the symbolic label of another predicative occurrence. For example, we can remark that the particular MOVE template (descriptive component) used to implement `c2` in Table 2 is systematically used to translate any sort of explicit or implicit transmission of an information (“The spokesman said...”). In this particular case of completive construction, the filler of the `OBJ(ect)` slot in the occurrence (in Table 2, `c2`) that materialises the ‘transmission’ template is a symbolic label (in Table 2, `c3`) that refers to another occurrence, i.e., the occurrence bearing the informational content to be spread out (“...the company has bought three factories abroad”).

Table 3 corresponds now to a narrative information that can be rendered in natural language as: “We notice today, 10 June 1998, that British Telecom will offer its customers a pay-as-you-go (payg) Internet service”.

To translate the general idea of ‘acting to obtain a given result’, we then use:

- A predicative occurrence (`c5` in Table 3), instance of a template pertaining to the ‘focusing on a result’ sub-tree of the BEHAVE branch of H_TEMP. This occurrence is used to express the ‘acting’ component, i.e., it allows us to identify the `SUBJ(ect)` of the action, the temporal co-ordinates, possibly the `MODAL(ity)` or the instigator (`SOURCE`), etc.
- A second predicative occurrence, `c6` in Table 3, which is used to express the ‘intended result’ component. This second occurrence, which happens ‘in the future’ with respect to the previous one (BEHAVE), is marked as hypothetical, i.e., it is always characterised by the presence of an uncertainty validity attribute, code ‘*’. Expressions like `after-10-june-1998` are concretely rendered as date ranges, see [6].
- A ‘binding occurrence’, `c4` in Table 3, linking together the previous predicative occurrences and labelled with `GOAL`, an operator pertaining to the taxonomy of causality of NKRL, see [6].

Table 3. An example of binding occurrence.

```

c4) (GOAL c5 c6)

c5)  BEHAVE SUBJ  british_telecom
      { obs }
      date1: 10-june-1998
      date2:

*c6)  MOVE   SUBJ  british_telecom
      OBJ    payg_internet_service_1
      BENF   (SPECIF customer_british_telecom)
      date1: after-10-june-1998
      date2:

```

Binding structures — i.e., lists where the elements are symbolic labels, c_5 and c_6 in Table 3 — are then another example of second-order structures used to represent the connectivity phenomena. The general schema for coding the ‘focusing on an intended result’ domain is now:

```

c $_{\alpha}$ ) (GOAL c $_{\beta}$  c $_{\gamma}$ )
c $_{\beta}$ ) BEHAVE SUBJ <human_being_or_social_body>
*c $_{\gamma}$ ) <predicative_occurrence, with any syntax>

```

In Table 3 ‘obs(erve)’ is, like ‘begin’ (see Table 1) and ‘end’, a temporal modulator, see [6]. ‘obs’ is used to assert that the event related in the occurrence ‘holds’ at the date associated with `date-1` without, at this level, giving any detailed information about the beginning or end of this event, which normally extends beyond the given date. Note that the addition of a ‘ment(al)’ modulator in the BEHAVE occurrence, c_{β} , that introduces an ‘acting to obtain a result’ construction should imply that no concrete initiative is taken by the SUBJ of BEHAVE in order to fulfil the result. In this case, the ‘result’, $*c_{\gamma}$, reflects only the wishes and desires of the SUBJ(ect).

5. ‘Search patterns’ and low-level inference procedures

A new version of NKRL, implemented in Java and XML/RDF compliant, has been realised in the context of the Concerto project. We recall here that the RDF model, see [10], is a framework for describing general-purpose metadata that is promoted by the W3C Committee and that is based on XML. In knowledge representation terms, the basic RDF data structure can be assimilated to a *triple* where two ‘resources’ are linked by a ‘property’ that behaves like a *dyadic* conceptual relation. The schema language RDFS (Resource Description Framework Schema), see [11], supplies a specialised vocabulary to model class and properties hierarchies, to define a set of core properties, to define domain and range restrictions about proper-

ties, etc. A description of some problems encountered when translating into RDF the NKRL data structures, and of the solutions adopted in this context, can be found in [12]. These problems have concerned mainly i) the correct representation of the NKRL variables and of their constraints; ii) the rendering in RDF of the *four* AECS operators making use only of *three* RDF ‘containers’, ‘Bag’, ‘Sequence’ and ‘Alternative’. Even if the solutions we have implemented work correctly, a first conclusion about RDF consists in stating that: i) we will continue to use the NKRL/RDF/XML format as an interlingua for information exchange; ii) but we will also continue to use the NKRL ‘native’ format for all sort of querying and inferencing operations. The reasons are:

- Although RDF directly provides some simple inferencing possibilities (e.g., making use of the fact that the property type **RDFS:subClassOf** is transitive), there is actually no real consensus about the possibility of implementing an RDF inference mechanism sufficiently powerful and general, able, e.g., to support an efficient implementation of the most complex NKRL inferencing tasks, see next Section.
- Moreover, it is not clear if the direct use of the RDF data structures will ever be suitable for complex inferential tasks, given the extreme ‘verbosity’ of these structures.

For these reasons, e.g., a solution to the ‘RDF inferencing problem’ adopted by the Ontobrocker group in Karlsruhe has been that of making use of a prototypical inference engine, SiLRI, to operate on the RDF structures by translating them into Frame-Logic format, that was then used for querying purposes [13]. A similar approach — translating RDF into Sowa’s Conceptual Graphs (CGs) and then using the usual CGs engines for querying — is followed by some CGs teams [14].

With respect to the second point mentioned above, we reproduce in Table 4 the native NKRL code (above) and the NKRL/RDF Concerto code (below) of an extremely simple narrative fragment: “On June 12, 2001, John and Peter were admitted (*together*) to hospital” — note that adding the indication ‘together’ forces an interpretation in a COORD style, see the previous Section. In the RDF coding, the COORD list that represents the filler of the SUBJ role in NKRL is rendered as a Bag, see [12]. It can be now more evident the interest of working directly on the NKRL data structures for any sort of operations that go beyond the simple transfer of information.

To supply now some details about the query and inferencing operations proper to NKRL, let us introduce informally the concept of ‘search pattern’.

Table 4. RDF/XML representation of a simple NKRL predicative occurrence.

```

c7)   EXIST   SUBJ (COORD john_ peter_): (hospital_1)
      { begin }
      date-1: 2-june-2001
      date-2:

<?xml version="1.0" ?>
<!DOCTYPE DOCUMENTS SYSTEM "CA_RDF.dtd">
<CONCEPTUAL_ANNOTATION>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdf="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
    xmlns:ca="http://projects.pira.co.uk/concerto#">
    <rdf:Description about="occ7">
      <rdf:type resource="ca:Occurrence"/>
      <ca:instanceOf>Template2.31</ca:instanceOf>
      <ca:predicateName>Exist</ca:predicateName>
      <ca:subject rdf:ID="Subj2.31" rdf:parseType="Resource">
        <concerto:filler>
          <rdf:Bag>
            <rdf:li rdf:resource="#john_" />
            <rdf:li rdf:resource="#peter_" />
          </rdf:Bag>
        </concerto:filler>
        <concerto:location>hospital_1</concerto:location>
      </ca:subject>
      <ca:listOfModulators>
        <rdf:Seq><rdf:li>begin</rdf:li></rdf:Seq>
      </ca:listOfModulators>
      <ca:date1>02/06/2001</ca:date1>
    </rdf:Description>
  </rdf:RDF>
</CONCEPTUAL_ANNOTATION>

```

Search patterns — which can be considered as the formal counterparts of natural language queries — are NKRL data structures that supply the general framework of information to be searched for, by filtering or unification, within an NKRL knowledge base. A simple example of search pattern, translating the query: “Was John at the hospital in July/August 2001?” (see the predicative occurrence *c7* in the upper part of Table 4) is represented in Table 5. The two timestamps associated with the pattern constitute now the ‘search interval’ that is used to limit the search for unification to the slice of time that it is considered appropriate to explore. In our example, the pattern of Table 5 can successfully unify occurrence *c7* in Table 4: in the absence of explicit, negative evidence, a given situation is assumed to persist within the immediate temporal environment of the originating event, see [6].

In the Concerto version of NKRL, a Java module called FUM (Filtering Unification Module) deals with search patterns. Unification is executed taking into account, amongst other things, the fact that a ‘generic concept’ included in the search pattern can unify one of its subsumed ‘specific concepts’ — or the instances (individuals) of a specific concept — included in the corresponding position of the occurrence. ‘Generic’ and ‘specific’ refer, obviously, to the structure of the NKRL

concept ontology, i.e., H_CLASS. Therefore, a search pattern asking which company has proposed a ‘pay-as-you-go’ (payg) Internet service to his customers will retrieve the predicative occurrence *c6* of Table 3, given that the SUBJ filler of the pattern, *company_*, subsumes in H_CLASS *telecom_company*, and the individual *british_telecom* in Table 3 is an instance of *telecom_company*.

Table 5. A search pattern that unifies the NKRL predicative occurrence of Table 4.

```
(?w IS-PRED-OCCURRENCE
      :predicate EXIST
      :SUBJ      john_
      :location of SUBJ  hospital_
      (1-july-2001, 31-august-2001))
```

From an algorithmic point of view, the main interest of FUM lies in the operations needed to unify correctly the complex fillers built up making use of the four AECS operators. These imply the decomposition of the complex fillers into tree structures labelled with the operators, and in the unification of the tree structures of the pattern with those of the occurrences following the priorities defined by the ‘priority rule’ already mentioned, see [1].

RQL [15] is a recent, declarative query language for RDF structures that is sensitive to the semantics of the RDFS primitives and that shares some analogies with the FUM implementation of the search pattern concept. RQL can, in fact, query the structure of the RDFS subclass hierarchy; moreover, it can match patterns along entire paths pertaining to RDF/RDFS graphs.

6. High-level inference procedures

The inference level supplied by FUM is only a first step towards more complex reasoning strategies.

6.1 Transformation rules

For example, the use of the ‘transformation rules’ allows us to deal with the problem of obtaining a plausible answer from a repository of predicative occurrences also in the absence of the explicitly requested information, by searching semantic affinities between what is requested and what is really present in the repository. The principle employed consists in using these rules to automatically ‘transform’ the original query (i.e., the original search pattern) into one or more different queries (search patterns) that — unlike, e.g., the ‘transformed queries’ in a database context — are not strictly ‘equivalent’ but only ‘semantically close’ to the original one.

To give a very simple example, suppose that, working in the context of an hypothetical knowledge base of NKRL occurrences about university professors, we want

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

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

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

In reality, the ‘always true’ implications (noted as $B \rightarrow A$, where we assume that the symbol ‘ \rightarrow ’ represents, as usual, the implication arrow) are not very frequent. Most transformations found in real world applications represent what we could call ‘modalised implications’. We will note them as $\diamond(B \rightarrow A)$, that means: “it is possible that B implies A ”. ‘ \diamond ’ is the usual modal operator for ‘possibility’, which satisfies then the relation $\diamond p = \neg \blacklozenge \neg p$ with respect to the second modal operator, ‘ \blacklozenge ’ = necessity’. An example of modalised transformation is given by the transformation $\tau 1$ in Table 6, which allows us to deal, by using an inference engine based on the FUM module, with the informal example above about ‘university professors’; as we can see, the antecedent and consequent of $\tau 1$ are formed by search patterns, expressed here in the traditional template format for clarity’s sake. Transformation $\tau 1$ says: “If someone ($var1$) receives a title from an authority by means of an official document, then it is possible that he has been physically present at that moment in the place ($var2$) where the authority is located”. This rule, for example, is not always valid in the case of a 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 becomes possible, at least in principle, to change $\tau 1$ into a true transformation by the addition of a few constraints on the variable $var3$, see for instance the expression: $var3 \neq \langle obtainable_by_correspondence_degree \rangle$. More examples, and a complete ‘theory’ of transformations, can be found in [16].

Table 6. A simple example of ‘transformation’ rule.

t1)	EXIST	SUBJ	$var1:(var2) \Rightarrow$	RECEIVE	SUBJ	$var1$
					OBJ	$var3$
					SOURCE	$var4:(var2)$
					MODAL	$var5$
	$var1$	=	$\langle human_being \rangle$			
	$var2$	=	$\langle location_ \rangle$			
	$var3$	=	$\langle title_ \rangle$			
	$var4$	=	$\langle authority_ \rangle$			
	$var5$	=	$\langle official_document \rangle$			

6.2 Hypothesis rules

Let us now suppose we have directly retrieved, thanks to FUM, a given information within a base of NKRL occurrences — for example, in a CONCERTO context (annotation of Internet news stories in the biotechnology domain), the occurrence corresponding to the conceptual annotation: “Pharmacoepia, an USA biotechnology company, has received 64,000,000 USA dollars from the German company Schering in connection with a R&D activity”, see occurrence *c8* in the upper part of Table 7. We will suppose, moreover, that this occurrence is not explicitly related with other occurrences in the base thanks to the presence of second order binding occurrences. Under these conditions, we can activate a specific Java module of the Concerto version of NKRL, *InferenceEngine*, asking it to test some rules of the hypothesis type to try to rely automatically the information found by FUM with other information present in the base. If this is possible, this last information will represent, in a way, a sort of ‘causal explanation’ of the information originally retrieved — i.e., in our example, an ‘explanation’ of the money paid to Pharmacoepia by Schering. A possible ‘hypothesis’ that could, in principle, fit our case is hypothesis *h1* reproduced in the lower part of Table 7. The data structures used to represent the different components of the hypothesis conform strictly to the format of the NKRL templates: they correspond in fact, to partially instantiated templates where some of the variables have been replaced by *H_CLASS* concepts. For example, the first ‘condition schema’ is derived from the template (6.42, *Produce:MutualCommitment*) that is routinely used in NKRL to represent an agreement between two or more parties.

From an algorithmic point of view, the present version of *InferenceEngine* works according to a standard backward chaining approach with chronological backtracking. The differences with respect to other well-known examples of use of this approach (*Mycin*, *PROLOG* ...) are mainly linked with the complexity of the NKRL data structures. This complexity implies the execution of difficult operations of reconstruction of the program environment whenever, after a deadlock, it is necessary to return to the previous choice point to try a new way of pursuing with the processing.

Table 7. An example of hypothesis rule.

```

c8) RECEIVE SUBJ      (SPECIF pharmacopeia_ (SPECIF
                        biotechnology_company usa_))
      OBJ              (SPECIF money_ usa_dollar (SPECIF amount_
                        64,000,000))
      SOURCE           (SPECIF schering_ (SPECIF
                        pharmaceutical_company germany_))
      TOPIC            r_and_d_activity
      date1 :
      date2 :
    
```

HYPOTHESIS h1

Premise :

```

RECEIVE      SUBJ      var1
              OBJ       money_
              SOURCE    var2

var1 = company_
var2 = human_being | company_
    
```

A company has received some money from another company or a physical person.

First condition schema :

```

PRODUCE      SUBJ      (COORD var1 var2)
              OBJ       var3
              BENF      (COORD var1 var2)
              TOPIC     (SPECIF process_ var4)

var3 = mutual_commitment | business_agreement
var4 = tool/product_
    
```

The two parties mentioned in the premise have concluded a general sort of business-oriented agreement about the creation of a new product.

Second condition schema :

```

PRODUCE      SUBJ      var1
              OBJ       var4
              MODAL     var5
              CONTEXT   var3

var5 = industrial_process | technological_process
    
```

The company that received the money has actually created the product mentioned in the first condition schema.

The screen dump of Figure 3 shows the global execution tree corresponding to the processing by InferenceEngine of the hypothesis of Table 7 making use of the information supplied by occurrence c8. The deadlocks are signalled as ‘!match’.

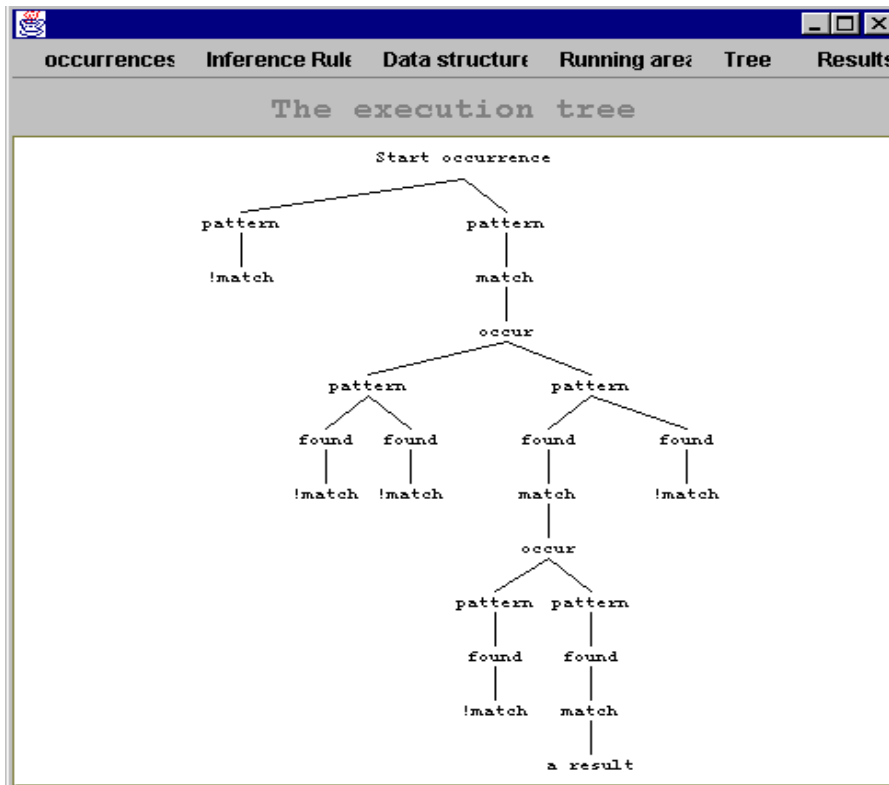


Figure 3. Global execution tree for the hypothesis h1 having as input occurrence c8 of Table 7.

The first set of operations consists in trying to unify the premise of the hypothesis and the event (the payment in our case, see occurrence c8) to be ‘explained’ — more exactly, in trying to unify the event and the different search patterns derived from the premise by systematically substituting to the variables *var1* and *var2*, see Table 7, the associated constraints. This first step allows then i) to verify again that the hypothesis tested is, *in principle*, suitable to ‘explain’ the particular event at hand and ii), mainly, to obtain from the external environment (the event, i.e., c8) some values for the premise variables *var1*, *var2*. In our case, the premise variable *var1* can only be substituted by the constraint *company_*; on the contrary, two substitutions, *var2* = *human_being* and *var2* = *company_* are possible for the variable *var2*. A first search pattern will be then constructed by substituting *human_being* for *var2*, i.e., a first unification with the event to explain

will be tried by using a search pattern corresponding to a payment done by an individual person instead of a company, as stated in occurrence *c8*. This unification obviously fails, see the first deadlock in Figure 3, reached starting from the top of the tree and traversing its left branch.

The engine then backtracks and builds up a new pattern making use now of the value *var2 = company_*, that will unify the value *schering_* in *c8*. The engine can then continue the processing of the hypothesis *h1*, see the right branch from the top of the tree of Figure 3. The two values *var1 = pharmacoepia_* and *var2 = schering_* will be then passed to the first condition schema.

This time, the search patterns derived from this first condition schema — by taking into account the values already bound to *var1* and *var2* and by replacing systematically, as usual, all the other variables with the associated constraints — will be tested (using FUM) against the general base of predicative occurrences to (try to) find possible unifications with these occurrences. As usual, many deadlocks (see again Figure 3) are generated in the course of these operations. Some are due, as in the premise case, to the chronological utilisation of the constraints. For example, when trying to make use of a pattern derived from the first condition schema where the variable *var3* has been substituted by its first constraint *mutual_commitment*, a failure will be generated. The occurrences retrieved in the base describe, in fact, some sort of *commercial* agreement between Pharmacoepia and Schering — e.g., *r_and_d_agreement* and *sale_agreement*, both specific terms of *business_agreement* (the second constraint) in the H_CLASS hierarchy — and not a *private* arrangement (*mutual_commitment*).

We will, eventually, find in the base an instantiation of the first condition schema corresponding to an event of the form: “Pharmacoepia and Schering have signed two agreements concerning the production by Pharmacoepia of a new compound”, see the second part of the screen dump of Figure 4 that illustrates the final results. The occurrence (*occur0*) displayed in the upper part of the screen corresponds to the event to ‘explain’ (“Pharmacoepia has received from Schering ...”, *c8* in Table 7). *COMPOUND_1* is the compound that is the object (TOPIC role) of the agreement.

The values associated with the variables *var3* et *var4* in the first condition schema will then be used to create the search patterns derived from the second condition schema. It will then be possible to retrieve an occurrence corresponding to the information: “In the framework of the agreement previously mentioned, Pharmacoepia has actually produced the new compound”, see again Figure 4. The global information retrieved through the execution of the hypothesis can then supply a sort of ‘plausible explanation’ of the Schering’s payment: Pharmacoepia and Schering have concluded some agreements for the production of a given compound, and this compound has been effectively produced by Pharmacoepia. We must note that, obviously, the example developed here has only a didactic purpose, and that this particular hypothesis is certainly not the only one that we could test to explain the case examined.

```

occurrences      Inference Rule      Data structure      Running area      Tree      Results

the result n∞ : 0

the start occurrence

occur0:
| RECEIVE
SUBJ(ect) : (SPECIF PHARMACOPEIA_ (SPECIF biotechnology_company USA_ )) :
OBJ(ect)  : (SPECIF money_ USA_DOLLAR (SPECIF amount_ 64,000,000 )) :
SOUR(ce)  : (SPECIF SCHERING_ (SPECIF pharmaceutical_company GERMANY_ )) :
TOPIC     : r_and_d_activity
{}
date-1    :
date-2    :
is instance of:Receive:Premisse
*****
the result for level 1

occur1:
| PRODUCE
SUBJ(ect) : (COORD PHARMACOPEIA_ SCHERING_ ) :
OBJ(ect)  : (COORD r_and_d_agreement sale_agreement ) :
BENT      : (COORD PHARMACOPEIA_ SCHERING_ ) :
TOPIC     : (SPECIF synthesis_ (SPECIF COMPOUND_1 compound_ ))
{}
date-1    :
date-2    :
is instance of:Produce:Cond1
*****
the result for level 2

occur2:
| PRODUCE
SUBJ(ect) : PHARMACOPEIA_ :
OBJ(ect)  : COMPOUND_1 :
MODAL(ity) : biotechnology_process
CONTEXT   : r_and_d_agreement
{}
date-1    :
date-2    :
is instance of:Produce:Cond2

Previous Next

```

Figure 4. Final results of the application of hypothesis h1 of Table 7.

Note that, e.g., the ‘filtering’ rules used in the Euforbia project to recognise ‘questionable’ Web sites previously annotated with ‘Euforbia labels’ (NKRL expressions translating the main content characteristics of these sites) are nothing that a downgraded version of rules in the ‘hypothesis’ style. More precisely, the Euforbia filtering rules are rules like that of Figure 7 where the ‘premise’ is missing, and the rule is then reduced to a sequence of ‘condition’ schemata that constitute together (AND) the ‘condition’ part of the rule. The implicit ‘action’ part of these rules is always of the type “access denied to the corresponding site”. A simple and intuitive example of Euforbia filtering rule is shown in Table 8.

Table 8. A simple EUFORBIA filtering rule.

RULE5:COND1)	OWN	SUBJ	var1
		OBJ	<i>property_</i>
		TOPIC	(SPECIF <i>dedicated_to</i> (SPECIF <i>internet_publishing var2</i>))
RULE5:COND2)	OWN	SUBJ	var2
		OBJ	<i>property_</i>
		TOPIC	var3

var1 = *site_* | *internet_site_section*
var2 = *information_item*
var3 = *sexuality_* | *violence_* | *racism_*

7. Conclusion

In the guise of Conclusion, we can now examine some ‘other’ possible, recent solutions that, explicitly or implicitly, have been proposed for the task of representing and processing by computer economically interesting narratives. We will limit ourselves to the ‘pure ontology’ approach; for some remarks about conceptual graphs see, e.g., [17].

For simplicity’s sake, we will assume here that the concepts of an ontology are defined as frames, i.e., represented according to two basic principles. The first is a *hierarchical* one, and it is materialised by the well-known *ISA* link: it relates the concept to be defined to *all* the other concepts of the ontology through the ‘generic’ (or ‘subsumes’), ‘specific’ (or ‘is-subsumed’) and ‘disjoint’ relationships. The second is a *relational* principle and, via the ‘attribute (property)-value’ mechanism, relates the concept to be defined to *some* of the other concepts already present in the ontology. We will not dwell here on secondary aspects of the structure of a frame like facets (used to describe properties of slots) and axioms (used to specify additional constraints).

It is now evident that an organisation in terms of frames (or an equivalent one) is largely sufficient to provide a *static* definition of the concepts — i.e., a definition *a priori* of each concept considered in itself. We can, on the contrary, wonder if this sort of organisation can be sufficient to define the *dynamic behaviour* of the concepts, i.e., to describe the mutual relationships affecting *a posteriori* the concepts and their instances when they take part in some concrete action, situation etc. (‘events’). If we want to represent a narrative fragment like “NMTV (an European media company) ... will develop a lap top computer system...”, asserting that *nmtv_* is an instance of the concept *company_* and that we must introduce an instance of a concept like *lap_top_pc* will not be sufficient. We must, in this case,

have recourse to a most complex way of structuring the concepts that, as in NKRL, includes also a ‘predicate’ and the associated ‘roles’, the temporal co-ordinates, etc.

In the literature, we find sometimes descriptions of frame-based systems trying to extend the ‘classical’ attribute-value mechanism to produce some representations of ‘events’ according to an NKRL meaning. To code, in fact, some simple sell/purchase events, it is possible to add, in the frame for, e.g., *company_*, slots in the style of *HasAcquired* or *AcquiredBy* or, better, it is possible to define a new concept like *company_acquisition* with slots like *NameOfTheCompany*, *Buyer*, *DateOfAcquisition*, *Price* etc. In this way, the instances of *company_acquisition* could be sufficient to describe in a complete way a (simple) sell/purchase event for a company. Trivial examples in this style have been still recently circulated on the Web in the context of the Standard Upper Ontology (SUO) effort, see <http://suo.ieee.org/>.

The limits of this approach are however evident. Restraining the description of sell/purchase events to the sole relationships between the buyer, the seller and the ‘object’ exchanged, with some additional information about, e.g., date of the transaction and price is, normally, only a very rough approximation of the original event, and a lot of useful information is lost. It is very likely, in fact, that the original information about a company’s sale was something in the style of: “Company X has sold its subsidiary Y to Z because the profits of Y had fallen dangerously these last years due to a lack of investments” or, returning to a previous example, “NMTV will develop a lap top computer system to put controlled circulation magazines out of business” or, to take a last example from the Concerto’s biotechnology domain, “Berlex made a milestone payment to Pharmacopeia because they decided to pursue an in vivo evaluation of the candidate compound identified by Pharmacopeia”. In Computational Linguistics terms, we are here in the domain of ‘Discourse Analysis’ that deals, in short, with the two following problems: i) determining the nature of the information that, in a sequence of statements, goes beyond the simple addition of the information conveyed by a single statement; ii) determining the influence of the context in which a statement is used on the meaning of this individual statement, or part of it. Being able to supply an, even approximate, representation of the ‘meaning’ of real events implies then to be able to deal with all sort of ‘connectivity phenomena’ like causality, goal, indirect speech, co-ordination and subordination etc., i.e., precisely the phenomena taken into account by the NKRL second order ‘binding’ structures.

It is now easy to imagine, on the contrary, the awkward proliferation of totally *ad-hoc* slots that, sticking to a strict frame paradigm, it would be necessary to introduce in order to approximate the connectivity phenomena in the above examples. This is, moreover, contradictory with the most recent tendencies of the theory of frame systems that postulate the creation *a priori* of systems of slots defined independently of any specific frame; see the Open Knowledge Base Connectivity (OKBC) protocol [4]. Trying to reduce the description of *events* to the description of *concepts* is then nothing that a further manifestation of the ‘uniqueness syndrome’ that affects some of the Artificial Intelligence (AI) and Knowledge Representation milieus — for a critical analysis, e.g., of the development of CYC, a well-known, very controversial AI system that is affected by the same sort of syndrome,

see Section 3.44 of [18]. In NKRL, we make use in an integrated way of several sorts of representational principles, and several years of successful experimentation with the most different narrative situations are there to give evidence that this seems indeed to be a quite reasonable approach.

References

1. Zarri, G.P.: NKRL, a Knowledge Representation Tool for Encoding the ‘Meaning’ of Complex Narrative Texts, *Natural Language Engineering - Special Issue on Knowledge Representation for Natural Language Processing in Implemented Systems*, 3 (1997) 231-253.
2. Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., and Klein, M.: OIL in a Nutshell. In: *Knowledge Acquisition, Modeling, and Management - Proceedings of the European Knowledge Acquisition Conference, EKAW’2000*. Springer-Verlag, Berlin, 2000.
3. Zarri, G.P., Bertino, E., Black, B., Brasher, A., Catania, B., Deavin, D., Di Pace, L., Esposito, F., Leo, P., McNaught, J., Persidis, A., Rinaldi, F., and Semeraro, G.: CONCERTO, An Environment for the ‘Intelligent’ Indexing, Querying and Retrieval of Digital Documents. In: *Foundations of Intelligent Systems - Proceedings of 11th International Symposium on Methodologies for Intelligent Systems, ISMIS’99*. Springer-Verlag, Berlin, 1999.
4. Chaudhri, A.F.V., Fikes, R., Karp, P., and Rice, J.: OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In: *Proceedings of the 1998 National Conference on Artificial Intelligence, AAAI/98*. AAAI Press/MIT Press, Cambridge (MA), 1998.
5. Buchheit, M., Donini, F.M., and Schaerf, A.: Decidable Reasoning in Terminological Knowledge Representation Systems, *Journal of Artificial Intelligence Research*, 1 (1993) 109-138.
6. Zarri, G.P., Representation of Temporal Knowledge in Events: The Formalism, and Its Potential for Legal Narratives, *Information & Communications Technology Law - Special Issue on Models of Time, Action, and Situations*, 7 (1998) 213-241.
7. Sowa, J.F.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove (CA), 1999.
8. Zarri, G.P., and Gilardoni: Structuring and Retrieval of the Complex Predicate Arguments Proper to the NKRL Conceptual Language. In: *Foundations of Intelligent Systems - Proceedings of 9th International Symposium on Methodologies for Intelligent Systems, ISMIS’96*. Springer-Verlag, Berlin, 1996.
9. Franconi, E.: A Treatment of Plurals and Plural Quantifications Based on a Theory of Collections, *Minds and Machines*, 3 (1993) 453-474.
10. Lassila, O., Swick, R.R., eds.: *Resource Description Framework (RDF) Model and Syntax Specification*. W3C, 1999 (<http://www.w3.org/TR/REC-rdf-syntax/>).

11. Brickley, D., Guha, R.V., eds.: Resource Description Framework (RDF) Schema Specification. W3C, 1999 (<http://www.w3.org/TR/WD-rdf-schema/>).
12. Zarri, G.P.: A Conceptual Model for Capturing and Reusing Knowledge in Business-Oriented Domains. In: Industrial Knowledge Management: A Micro-level Approach, Roy, R., ed. Springer-Verlag, London, 2000.
13. Decker, S., Brickley, D., Saarela, J., and Angele J.: A Query and Inference Service for RDF. In: Proceedings of QL'98 - The Query Languages Workshop. W3C, 1998 (<http://www.w3.org/TandS/QL/QL98/>).
14. Corby, O., Dieng, R., and Hébert, C.: A Conceptual Graph Model for W3C Resource Description Framework. In: Proceedings of the International Conference on Conceptual Structures, ICCS 2000. Springer-Verlag, Berlin, 2000.
15. Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., and Tolle, K.: The RDFSuite: Managing Voluminous RDF Description Bases - Technical Report. FORTH, Institute of Computer Science, Heraklion, 2000.
16. Zarri, G.P.: 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. ACM Press, New York, 1986.
17. Lukose, D., Mineau, G., Mugnier, M.-L., Möller, J.-U., Martin, P., Kremer, R., and Zarri, G.P.: Conceptual Structures for Knowledge Engineering and Knowledge Modelling. In: Supplementary Proceedings of the 3rd International Conference on Conceptual Structures: Applications, Implementation and Theory. Department of Computer and Information Sciences of the University of California, Santa Cruz (CA), 1995.
18. Bertino, E., Catania, B., and Zarri, G.P.: Intelligent Database Systems. London, Addison-Wesley and ACM Press, 2001.