

Chapter 2

The Knowledge Representation Strategy

This chapter explains in a detailed way the main encoding principles underpinning the NKRL style of knowledge representation. It describes NKRL as a sort of *general environment* to be used to represent formally all sorts of narratives; *specific solutions* used to represent *particular kinds of narrative knowledge* (particular narrative “contents”) will be detailed in Chapter 3. After reading Chapter 2, interested people should be able (i) to understand the characteristic NKRL (external) code that has been used in the framework of all the existing NKRL applications; (ii) to encode themselves in NKRL’s terms at least in some simple example of nonfictional narrative.

Following Section 2.1, which is devoted to the description of the general, “architectural” organization of the “basic” NKRL language and to the introduction of its four “components,” Section 2.2, will give a detailed explanation of the *data structures* used for these components. Section 2.3 will deal with the “second-order” structures used in NKRL to take into account those *connectivity phenomena* that, as we have seen in Chapter 1, are so important for the correct representation of narrative information. A short conclusion, Section 2.4, will end the chapter.

2.1 Architecture of NKRL: the Four “Components”

From an “architectural” point of view, we can see the “basic” NKRL language (i.e. without considering the second-order tools at the moment) as structured into *four connected “components,”* even if the differences among these components are somewhat blurred in the implementation software. Each of them *takes into account a particular category of narrative phenomena,* making use of specific knowledge representation tools that can be considered as the *best fitted* for modeling these phenomena. The four components are *definitional, enumerative, descriptive* and *factual*; we provide a general description of their main characteristics below.

- The *definitional component* concerns the formal (binary) representation of both the *general* (like “human being,” “amount” or “artifact”) and the

specific notions (like “business person,” “taxi,” “city” or “demand for ransom”) that must be considered for taking correctly into account the narrative information proper to the different application domains. The NKRL formal representations of these notions are called “*concepts*” – denoted in general as C_i . NKRL concepts are inserted into a *generalization/specialization directed graph structure* (often, but not necessarily, reduced to a tree) that, for historical reasons, is called HClass(es), “hierarchy of classes.” The data structures of the NKRL concepts correspond relatively well to the analogous structures that can be built up using the usual environments for the creation of ontologies like Protégé [Noy et al., 2000], WebODE [Arpírez et al., 2003] or OntoEdit [Sure et al., 2002]. In the *concrete NKRL structures expressed in “external” NKRL format*, the concepts are “named” making use of (*lower case*) *symbolic labels* like `human_being`, `business_person`, `taxi_`, `city_`, `ransom_demand`, etc. To discriminate between concepts and other categories of the language also represented in practice by lower case “names,” e.g. the “modulators” (see Section 2.2.2.4), the symbolic labels denoting the concepts always include at least an “underscore” symbol.

- The “*enumerative component*” concerns the formal representation of the *instances* I_i (*specific examples*) of the notions (*concepts*) pertaining to the *definitional component* – as we will see in Chapter 3, not all the NKRL concepts can be endowed with instances. The formal representations in NKRL’s terms of such instances take the name of “*individuals*”; individuals are then created *by instantiating the properties of the concepts of the definitional component*. Individuals are characterized by the fact of *being countable* (*enumerable*) and of always being associated, often in an implicit way, with a *spatio-temporal dimension*. Within the actual NKRL structures, each individual owns a *unique conceptual label* (JOHN_SMITH, PARIS_, RANSOM_DEMAND_4): two individuals associated with the same NKRL description but having different labels will be considered as different individuals. In the “external” format of NKRL, individuals are represented in upper case. To discriminate between individuals and other categories of the language also denoted in upper case, like predicates and roles (Section 2.2.2.2), their symbolic labels always include at least an “underscore” symbol.
- The “*descriptive component*” concerns the formal representation of *general classes of elementary events* like “moving a generic object,” “formulate a need,” “be present somewhere,” “starting a company,” “committing acts of violence against someone,” etc. As we have already stated in Chapter 1, the *formal (n-ary) representations of these general classes in NKRL terms are called “templates”*, t_i . Note that the term “elementary event” is used in NKRL to denote also, in general, all the fuzzy, associated notions like state, situation, period, episode, history, process, action, etc.; see also Zarri [1998]. The general classes of elementary events (templates) of the descriptive component are *obtained by abstraction/generalization from sets of specific, elementary narrative “events” (in the general meaning evidenced before) that we can observe (or imagine) in the real world*. Within the

concrete NKRL structures, templates are denoted in “external” format by symbolic labels such as the `Produce:HumanBeingInjuring` already met in Section 1.1.2; see Chapter 3 for details. Templates are inserted into an inheritance hierarchy (in this case, a simple tree) that is called HTemp (hierarchy of templates).

- The “factual component” provides the formal representation (as instances of the templates of the descriptive component) of the different, possible elementary events that can be isolated within a narrative. As already stated, these formal representations are called “predicative occurrences”, c_i . A predicative occurrence is then the NKRL representation of an elementary narrative information like “Tomorrow, I will move the wardrobe,” “Lucy was looking for a taxi,” “Peter lives in Paris,” “Company X, located in Geneva, has taken the control of Company Y,” etc. The elementary events (the predicative occurrences) eventually concern the description of a particular set of interactions among individuals (and, in case, concepts) – see the interaction between Lucy and a particular wardrobe in the example before, or between Brutus and Caesar in the “stabbing” event of Chapter 1 – where: (i) the “semantic category” of the set of interactions is defined by the particular “deep predicate” (“stabbing”) associate with the event; (ii) in conformity with the Kimian analysis of Chapter 1, the set of interactions is delimited from a spatio/temporal point of view. This implies, among other things, that a specific “duration” (a specific temporal interval that can be “empty” in the case, for example, of a “future” or “hypothetical” event) will always be associated with a predicative occurrence. Within the concrete NKRL representations in external format, occurrences are denoted (usually) by a sort of “pointed” notation, introduced in Section 2.2.2.2.

With respect, for example, to a description logics’ perspective [Baader et al., 2002], we note that *the definitional component in NKRL corresponds roughly to Tbox and the enumerative component to Abox*. Tbox contains, in fact, *intensional knowledge that describes the general properties of concepts*; ABox contains the *corresponding extensional knowledge*. No description logics structures *correspond, on the contrary, to the descriptive or factual structure of NKRL* that constitute then, from a data structure point of view, the main innovation introduced by this last language.

We can conclude this section by emphasizing that, in NKRL, the concepts (definitional component) and their instances (individuals, enumerative component) are *kept conceptually distinct* – even if *they are represented using, in practice, the same data structures*; see the following sections. The main reason for operating this separation is linked with the *very different epistemological status of concepts* – which define a *generic, abstract mold* for some *compulsory notions* that must be taken into consideration in a given domain – *with respect to the individuals*, which represent only *transitory entities found out in the context of some concrete events* of the domain. In this respect, *concepts can be considered as necessary and permanent, at least in the context of a given application* – even if, in practice, customizing

HClass for an application requires the addition of several new, low-level concepts *online* (and, in some cases, their modification and withdrawal). Individuals represent, on the contrary, *unpredictable, randomly occurring entities* stored consecutively (and continuously) into an NKRL-based system, characterized by the absence of any *necessity attribute* and that can, therefore, be *eliminated without any consequence for the logical coherence of the system*.

Similar considerations could also be formulated about the reasons for distinguishing between descriptive (templates) and factual component (predicative occurrences) entities – even if, in this case, the *need* for such a differentiation is more concretely evident (the predicative occurrences constitute the formal basis for encoding our nonfictional narratives).

Note that the separation concepts/instances (individuals) can appear as an *obvious requirement today* but, in reality, recognizing that there is a need for this distinction is a *relatively recent notion*. Many “expert systems” environments in the 1970s and 1980s could not differentiate between the two. A well-known example in this context is that of KEE [Fikes and Kehler, 1985], one of the early and most powerful commercial environments for the development of complex frame-based systems. Followers of a uniform approach in which all the “units,” to adopt the KEE terminology, have the same status, claim that, for many applications, this distinction *is not very useful and only adds some important logical and semantic difficulties*. From a knowledge representation point of view, the formalization of the concepts/instances distinction can be traced back to the well-known Woods [1975] paper “What’s in a link: foundations for semantic networks” and to *his claim for a separation between “intension” (concepts) and “extension” (instances, e.g. the sets of entities that “materialize” the concepts)*. One of the examples used by Woods concerns the classical Frege’s (and Quine’s) example about the two different concepts of “morning star” and “evening star” (intension) that are both materialized by the same instance (extension), the planet Venus. We note that NKRL follows Woods’ conclusion of conceiving all the nodes of an ontology as *intensional entities* (i.e. as concepts), and *to add a specific predicate of existence only when necessary to introduce some instances*; see Section 2.2.1.4. A corollary of this decision concerns the fact that, in NKRL, individuals (instances) are *terminal symbols (leaves) of the HClass hierarchical structure (the NKRL “ontology” of concepts)* and that they *cannot be further specialized*, i.e. they cannot receive further instances; again, see Section 2.2.1.4 and Zarri [1997].

2.2 The Data Structures of the Four Components

The data structures proper of the four components are *systematically implemented as structured objects identified by a symbolic label*. Within this general framework, very important differences exist between the definitional/enumerative and the descriptive/factual structures.

2.2.1 *Definitional/Enumerative Data Structures*

The definitional and enumerative data structures that support concepts and individuals have been implemented in a relatively simple and straightforward way, given that *the most complex and intriguing aspects of the “narrative” phenomena are taken into account by the descriptive and factual data structures*. Therefore, in designing these structures, the aim has been simply that of implementing *a clean and simple semantics allowing very efficient and fast managing of operations*. These structures (basically “binary” – see the discussion in Chapter 1) have, then, been organized in a (traditional and well-known) *frame-like fashion*, i.e. as bundles of *properties (attributes, features, qualities,...)/value* relations where neither the number nor the order of the properties is fixed. This type of organization, then, concerns both the *definitions* associated with the concepts and the *descriptions* of the corresponding individuals.

NKRL “frames” conform to the general requirements of the Open Knowledge Base Connectivity (OKBC), [Chaudhri et al., 1998], *and are not very different from the object structures used in Protégé* [Noy et al., 2000; Gennari et al., 2002]. As is well known, Protégé is a sort of *standard* for the set up of frame-oriented ontologies: in a recent survey based on 627 answers to a questionnaire about the practices of the SW community [Cardoso, 2007], Protégé was the ontology editor most frequently mentioned with a “market share” of 68.2%. A prototype that extends the present Protégé system to support *collaborative ontology editing in a Web-based environment* is presented, for example, in Tudorache and Noy [2007]. Note that the NKRL software makes use of a *specific environment* – the *HclassEditor* – to set up its frame structures and to store them onto an ORACLE database; see also Appendix A. This environment is very similar to the *standard* (i.e. without additional plugins) Protégé environment, and the two can, at least in principle, *be indifferently used to introduce new concepts or individuals into HClass or to modify them*. Some of the reasons for building up a specific NKRL environment for the set up of frame-like structures are explained in the following sections. Information below refers, in general, to *concepts*: the specific problems proper to the *implementation of NKRL instances* (individuals) are examined in detail in Section 2.2.1.4.

2.2.1.1 General Principles of the HClass “Frames”

We will quickly recall here some well-known notions concerning the set up of ontologies of concepts under the form of “frames.”

These ontologies are structured as *inheritance hierarchies* making use of the *IsA* link (also called *AKindOf* (*Ako*), *SuperC*, etc.). A relatively unchallenged (however, see Brachman [1983]) semantic interpretation of *IsA* states that this relationship among concepts, when noted as (*IsA* C_2 C_1), means that concept C_2

is a *specialization* of the more general concept C_1 . In other terms, C_1 *subsumes* C_2 . This assertion can be expressed in logical form as

$$\forall x (C_2(x) \rightarrow C_1(x)) \quad (2.1)$$

Equation (2.1) says that, if any elephant_ (C_2) *IsA* mammal_ (C_1), and if CLYDE_ is an elephant_, then CLYDE_ is also a mammal_. When Eq. (2.1) is interpreted strictly, it also implies that a given concept C_k and all its instances *must* inherit *all* the properties and their values of *all* the concepts C_i in the hierarchy that have C_k as a specialization; we speak in this case of *strict inheritance*. Note that, even under the strict inheritance hypothesis, totally new properties can be added to C_k to specialize it with respect to its parents. The problems connected with a systematic interpretation of “inheritance” as “strict inheritance” are discussed in Bertino et al. [2001: 139–147].

A “frame” is now basically *a set of properties* (normally called “slots”) *with associated classes of admitted values* (the “fillers” of the slots) that is *associated with the nodes representing the concepts* (not necessarily the totality of the concepts) of a given ontology. Introducing a frame corresponds, then, to add to the fundamental *IsA* relationship that *necessarily concerns all the concepts* C_i of an ontology a new sort of relationship between a *specific* concept C_k to be defined and *some* of the other concepts C_1, C_2, \dots, C_n of the ontology. The relationship concerns the fact that C_1, C_2, \dots, C_n are used in the frame defining C_k to indicate the *class of fillers* that can be associated with the “slots” of this frame – the slots denoting, as already stated, the main properties (attributes, qualities, etc.) of C_k . In NKRL (HClass), no fixed number of slots exists and no particular order is imposed on them; slots can be accessed by their names.

To see how the relationships between a generic C_k and concepts C_1, C_2, \dots, C_n can be formally described, let us suppose that a specific concept C_1 (e.g. *home_address*) is endowed with a property R_1 (e.g. *HasNecessarily*) that associates it with a concept C_2 like *postal_code*. We can formalize this situation as

$$\forall x (C_1(x) \rightarrow \exists y (C_2(y) \wedge R_1(x, y))) \quad (2.2)$$

Equation (2.2) means, according to our example, that every home address is endowed with the property of having a postal code (the slot *HasNecessarily* will appear in the frame associated with the concept *postal_code*). As already stated, properties (slots) can be *systematically* inherited along an inheritance hierarchy only under the “strict inheritance” hypothesis. Instances (“individuals” in the NKRL terminology) inherit from the father concept.

As usual in the ontological domain, it is possible to allow an HClass concept to inherit its properties (slots) from two or more concepts by clicking on the button “Multi-Inheritance” in the *hClassEditor* environment. *The user can then choose the concepts in the ontology that they want to assume as “fathers” for the selected concept.*

We can remark now on a first, important difference with respect to the Protégé approach *in case of possible conflicts among inherited slots in a multiple inheritance context*, i.e. in the case of slots coming from different father concepts and having the *same name but different values*. Protégé makes use, in fact, of automatic heuristics to solve these conflicts. A *precedence list* is computed in a “mechanical” way by starting with the first leftmost concept that represents a generalization (superconcept) of the concept where the conflict has been observed; the construction of this list proceeds by visiting depth-first the nodes in the left branch, then those of the right branch, then the join, and up from there. The “conflicting” concept inherits the properties of the first element of the list. This technique depends, obviously, *on the particular arrangement adopted in the construction of the inheritance hierarchy, and can oblige one to insert a number of “dummy concepts” in order to produce a correct precedence list*. In NKRL, we have decided, on the contrary, *to have systematic recourse to the intervention of the user to solve any possible incoherence*. As stated above, the user will then be *asked to select the correct set of properties/values explicitly for the concept under examination*, i.e. to specify exactly the *superconcept from which a given conflicting property must be inherited*.

We conclude this section by noticing that, as we have seen, the use of multi-inheritance in HClass is admitted; we must now add that it is also *strongly discouraged*. We agree, in fact, with Guarino [e.g. Guarino, 1998] when he says that, in the ontological domain, *there is a tendency towards relying on multiple inheritance (“ISA overloading”) to solve all the possible “polysemy” (in the most general meaning of this term) problems*. In reality, a more in-depth examination of these (supposed) multiple-inheritance situations shows that, for example, by simply duplicating the “polysemic” concepts, or by reducing to “properties” (roles) some “dubious” concept, *the need for multiple inheritance disappears*. Moreover, in NKRL, the possibility of expressing *dynamic* concepts making use of descriptive/factual structures (i.e. by expressing some of these “concepts” as templates/predicative occurrences – see Section 2.2.1.3) can help to further reduce the “ISA overloading” phenomenon and to eliminate then the need for multi-inheritance connections.

2.2.1.2 Prototype Slots

In a “frame” context, there always has been a lot of debate about the theoretical problems linked with the *arbitrariness* in the choice of the slots (in the choice of the particular properties intended to describe in a better way the “meaning” of a concept). A “classical” paper in this framework is Wilensky [1987]. Given the evident impossibility of finding a global (and shared) solution to this problem at the theoretical level, some practical solutions have been implemented. They are all based on the use of *meta-structures intended to describe in a precise way the computational behavior of a given slot* – and, therefore to give, in a certain way, also a sort of “definition” of the slot. A well-known approach in this context consists in adding *facets* to the slots, where a facet is *an annotation describing*

some characteristics of the slots like, for example, type restrictions on the values of the slot (VALUE-TYPE facets) and specifications of the exact number of possible values that the slot may take on (CARDINALITY facets). An evolution of the facet approach concerns making use of full “slot-control schemata” – i.e. of a *structured object containing complete information about the properties of a specific slot*, not only those concerning domain, range and cardinality, but also, for example, *detailed inheritance specifications for the slot*.

To deal, at least partly, with the *arbitrariness* problem (in particular the arbitrariness of the “attribute” slots – see below), NKRL follows the “slot-control schemata” approach, giving then to the HClass slots the status of *full-fledged objects*. As in Protégé and in the OKBC requirements, slots in NKRL are then defined a priori as “*prototype slots*,” *independently then from the specifications of any particular concept*. Prototype slots are grouped in a list, and *attached to specific concepts when necessary*. According to a classical example, an “attribute” like “age” can, in fact, be used to describe the characteristics of both (among other things) the concepts *author_* and *manuscript_*. The slot *Age* will be then defined as a *slot prototype* whose minimal value is set to “0,” and which cannot be used in a situation where it is required that the slot must be capable of containing negative values. *Age* will then be attached, in case, to *author_*, *manuscript_* and to many other possible concepts. Analogously, an attribute like *Name* will be introduced as a slot prototype that can be filled only with a value of the “string of characters” type, etc.

In NKRL, the *data types* admitted for the fillers of the HClass (prototype) slots are *strictly the following*:

- Boolean;
- (HClass) concept/individual;
- double (precision);
- integer;
- string (of characters).

Moreover, the *values* corresponding to the previous data types can be submitted, as usual, to *constraints*. The *cardinality constraint* defines the number of values that can be associated with a slot; *this number must be included between the CardMin and CardMax limits*, which respectively define the minimal and maximal number of values. If the slots are of the *integer* or *double precision* types, then their fillers must be included *within a numeric interval*. If the slots are of the *concept/individual* type, then the fillers can only be chosen among *the instances*, and *all the specific terms with their instances*, of the concept defined as default value, etc.

There is, however, *an important difference between our approach to the management of the prototype slots and that of Protégé* – in Protégé, the prototype slots are called *template slots*. In this last environment, the features originally associated with a prototype (template) slot can always be *overridden* when the prototype is associated with a concept. For us, *the constraints associated with a specific prototype* (type of the value, cardinality, min/max, possible default

value, etc.) *are strictly enforced* to maintain the coherence of the global knowledge base, and they *cannot be changed* when a prototype is associated with a specific concept, becoming then a slot proper to this concept. This means that, when a prototype slot is concretely used and the user realizes that the constraints associated with this slot are not well tailored to her/his specific needs, *the user cannot intervene directly on the slot at the concept level* but must either (i) *edit the original prototype slots (change its constraints)* or (ii) *define a new prototype slot that fits exactly her/his specific needs*.

There is, in practice, only a possibility of *directly editing* the information already associated with the slots of a concept, and this concerns a *restriction of the domain of the admitted values*. More precisely:

- a modification of the cardinality is allowed only if *the new CardMin/Card-Max interval is included within the interval of the original prototype*;
- the same restriction applies to the *min/max numeric interval of the integer and double precision values*;
- if a default concept is associated with the slot, then this default can be changed only if the new default is *an instance, or a specific term with all its instances, of the original default*.

Figure 2.1 reproduces an *hClassEditor* screen dump, showing the use of prototype slots to associate properties with the *biotechnology_company* concept – a specific term of *biological_company* in *HClass*.

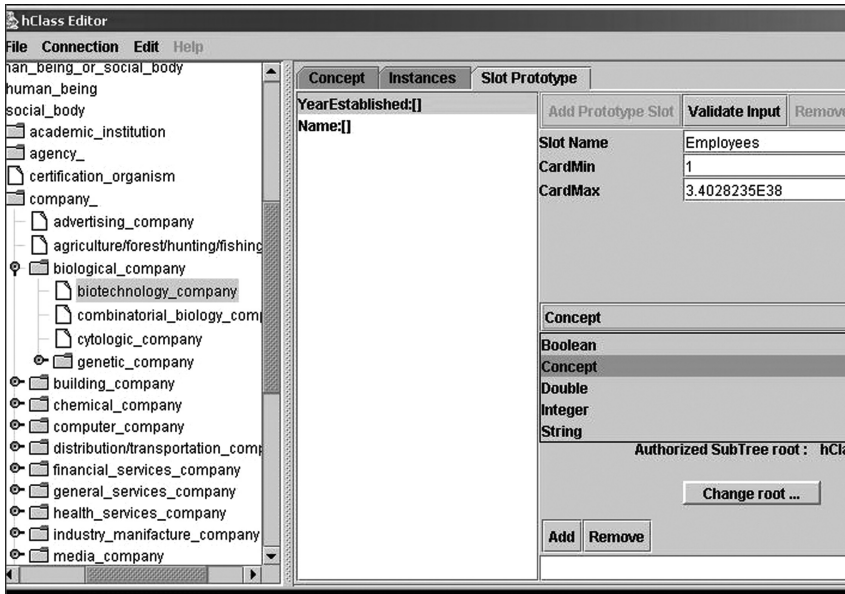


Fig. 2.1 Utilization of the prototype slots

In the situation illustrated, two slots, i.e. `YearEstablished` and `Name`, have already been associated with the concept under definition; moreover, we have selected again the “Slot Prototype” window to add a new slot, `Employees`, to `biotechnology_company`. In this window we have clicked on the “Add Prototype Slot” button, and selected `Employees` in the prototype slots list. `Employees` appears with its features, the cardinality constraints and the type, the latter being “Concept.” The *default value* associated with the prototype is `hClass`; according to the third of the rules for editing slots introduced above, *we can “specialize” the default making use of the “Change root . . .” button*. Clicking on this opens a “Select concept” window (see Fig. 2.2) where a new default for the slot will be selected, i.e. `individual_person`. Clicking on the “Validate Input” button (Fig. 2.1) will cause the `Employees` slot to be listed among the `biotechnology_company` slots.

2.2.1.3 Categories of Properties

From a *semantic* point of view, the properties of the concept C_k to be defined (i.e. all the possible types of relationships between C_k and some other concepts

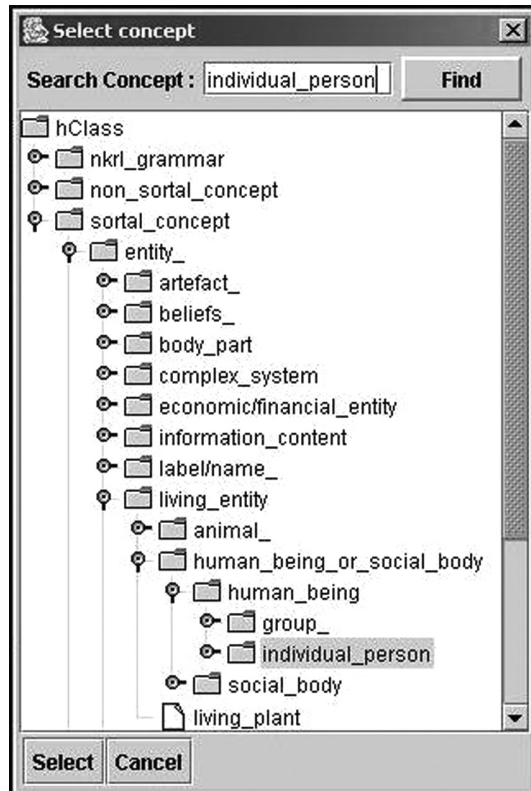


Fig. 2.2 Selecting a new default value for the `Employees` slot

Table 2.1 Types of properties in an NKRL frame

{OID	
[Relation	(IsA InstanceOf: HasSpecialization HasInstance: MemberOf HasMember: PartOf HasPart: UserDefined ₁ : ... UserDefined _n :)
Attribute	(Attribute ₁ : ... Attribute _n :)
Procedure	(Procedure ₁ : ... Procedure _n :)}

C_1, C_2, \dots, C_n of the ontology that appear in the definition of the frame associated with C_k – see Section 2.2.1.1) can be classed in three categories: *relations*, *attributes*, and *procedures*; see Table 2.1.

OID (Object Identifier) stands for the “symbolic name” of the particular concept to be defined; see, in an NKRL context, some symbolic labels like *business_person*, *postal_code* or *biotechnology_company* introduced in the previous paragraphs. The presence of the *IsA* and *HasSpecialization* properties is *mandatory for the concepts* – as is *mandatory the presence* of *InstanceOf* and *HasInstance* *for the individuals*; these properties define, in fact, *the general structure of the ontology*. From a practical point of view, we can note that the *IsA* and *HasSpecialization* properties do not require an implementation under the form of a “slot,” given that they are *implicitly defined by the structure of the HClass hierarchy* (e.g. see Fig. 2.1); the same is true for the *InstanceOf* and *HasInstance* properties. *All the residual properties are implemented as instantiations of predefined prototype slots*, independently from their semantic category.

“Relation-type” Properties

The properties of the “relation” type are used to represent *mutual kinds of relationships between a concept or individual and other concepts or individuals of the ontology*. Eight “standard” properties of the “relation” type are used in NKRL (see Table 2.1); they are: *IsA*, and the inverse *HasSpecialization*, *InstanceOf*, and the inverse *HasInstance*, *MemberOf* (*HasMember*) and *PartOf* (*HasPart*). Some of their formal attributes (under the “strict inheritance” hypothesis) are described in Table 2.2 (also see Schiel [1989]); in Table 2.2, C_i denotes a generic concept and I_i a generic instance/individual.

An important point about these properties concerns the fact that, because of the general definitions of “concept” and “instance” and because of the characteristics of *IsA*, *InstanceOf*, *PartOf* and *MemberOf* illustrated in Table 2.2,

Table 2.2 Some formal attributes of *IsA*, *InstanceOf*, *PartOf*, *MemberOf*

$(\text{IsA } C_1 C_2) \wedge (\text{IsA } C_2 C_1) \leftrightarrow C_1 \equiv C_2$
$(\text{IsA } C_1 C_2) \wedge (\text{IsA } C_2 C_3) \rightarrow (\text{IsA } C_1 C_3)$
$(\text{IsA } C_1 C_2) \wedge (\text{IsA } C_1 C_3) \rightarrow \exists C_4 (\text{IsA } C_2 C_4) \wedge (\text{IsA } C_3 C_4)$
$(\text{PartOf } C_1 C_2) \rightarrow \neg (\text{PartOf } C_2 C_1)$
$(\text{PartOf } C_1 C_2) \wedge (\text{PartOf } C_2 C_3) \rightarrow (\text{PartOf } C_1 C_3)$
$(\text{IsA } C_1 C_2) \wedge (\text{PartOf } C_2 C_3) \rightarrow (\text{PartOf } C_1 C_3)$
$(\text{IsA } C_1 C_2) \wedge (\text{PartOf } C_1 C_3) \rightarrow (\text{PartOf } C_2 C_3)$
$(\text{IsA } C_2 C_3) \wedge (\text{PartOf } C_1 C_3) \rightarrow (\text{PartOf } C_1 C_2)$
$(\text{IsA } C_1 C_2) \wedge \text{MemberOf } C_2 C_3 \rightarrow (\text{MemberOf } C_1 C_3)$
$(\text{InstanceOf } I_1 C_1) \wedge (\text{IsA } C_1 C_2) \rightarrow (\text{InstanceOf } I_1 C_2)$
$(\text{PartOf } I_1 I_2) \wedge (\text{InstanceOf } I_1 C_1) \wedge (\text{InstanceOf } I_2 C_2) \rightarrow (\text{PartOf } C_1 C_2)$
$(\text{PartOf } C_1 C_2) \wedge (\text{InstanceOf } I_2 C_2) \rightarrow \exists I_1 (\text{InstanceOf } I_1 C_1) \wedge (\text{PartOf } I_1 I_2)$

a concept or an individual (instance) cannot make use of the totality of the eight “relation properties” introduced above. More exactly:

- The relations *IsA*, and the inverse *HasSpecialization*, are reserved to concepts.
- *HasInstance* can only be associated with a *concept*, *InstanceOf* with an *individual* (i.e., the concepts and their instances, the individuals, are linked by the *InstanceOf* and *HasInstance* relations).

Moreover, *MemberOf* (*HasMember*) and *PartOf* (*HasPart*) can only be used to link concepts with concepts or instances with instances, but not concepts with instances, see also [Winston et al., 1987].

We can also remark that *only two*, i.e. *MemberOf* and *PartOf* (and their inverses), of the so-called “meronymic” relations appear in the list of the “relation properties” of NKRL. Our basic criterion for differentiating between *MemberOf* and *PartOf* is likened with the homogeneity (*MemberOf*) or not (*PartOf*) of the component parts; in this way, “Cardinal Ratzinger is *MemberOf* the Holy College” (or “This tree is *MemberOf* the forest”) and “a handle is *PartOf* a cup.” *PartOf* is also characterized by a sort of “functional” quality (see again the “a handle is part of a cup” example) that is absent in *MemberOf*.

As is well known, *six different meronymic relations* are defined on the contrary in Winston et al. [1987]: component/integral object (corresponding to our *PartOf*), member/collection (corresponding to our *MemberOf*), portion/mass, stuff/object, feature/activity, place/area. Note that Winston et al. [1987] is still the *reference paper* for people interested in the practical implications of making use of meronymic concepts; for an overview of some more theoretical (and description logics-oriented) approaches, see Artale et al. [1996]. Recent work in a SW context has apparently not introduced any significant advance in this field, e.g. see Rector and Welty [2005]. The justification of our approach is twofold:

- A first point concerns the wish of keeping the *HClass* component of the NKRL language *as simple as possible*. In this context, the only “relation

properties” that it was really necessary to introduce (in addition, of course, to *IsA*, *InstanceOf* and their inverses) were the *MemberOf* and *HasMember* relations, given that they intervene in the definition of the data structures used, for example, to represent plural situations in NKRL; see Section 2.2.2.3 and Appendix B. We have also then added, of course, the “complementary” *PartOf* and *HasPart* relation properties.

- On the other hand, dealing systematically with the examples of “non-NKRL relation properties” given by Winston and his colleagues by using only the four meronymic relations accepted in NKRL leads to results *that are not totally absurd*, even if, sometimes, some aspects of the original meaning are lost.

With respect to this last point, an example given in the Winston et al. paper is “this hunk is part of my clay,” which is interpreted as an illustration of the “portion/mass” meronymic relation. We can show that, in an NKRL context, this example could also be understood as a *MemberOf/HasMember* relation, in the style of “this tree is *MemberOf* the forest.” We can, in fact, interpret “my clay” as an individual, *GENERIC_PORTION_OF_CLAY_1*, an instance then of *generic_portion_of_clay* that can be considered as a (low-level) specialization of *physical_entity*. As we will discuss in detail in Section 3.1.2.1, we cannot, in fact, make use of a simpler individual in the *CLAY_1* style that would be a *direct instance* of *clay_*; the latter, as a *substance_*, *cannot be endowed in fact with direct instances*. It is now easy to see *GENERIC_PORTION_OF_CLAY_1* as formed (*HasMember*) by several hunks, *HUNK_1*, ..., *HUNK_n*. The latter are, in turn, instances of a concept like, for example, *hunk_of_clay*: as the trees in the forest, they are all *homogeneous* and play *no particular functional role* with respect to the whole represented by *GENERIC_PORTION_OF_CLAY_1*. Passing now to other examples: “A martini is partly alcohol” (stuff/object) can be easily rendered using, for example, an “attribute property” (see below); “an oasis is a part of a desert” (place/area) can be represented using *PartOf* (there is no particular homogeneity between “oasis” and “desert”); etc.

“Attribute-type” Properties

The *characteristic properties* of a concept/individual are specifically represented by the slots of the “attribute” type. For example, for a concept like *tax_*, possible attributes (slots) are *TypeOfFiscalSystem*, *CategoryOfTax*, *Territoriality*, *TypeOfTaxPayer*, *TaxationModalities*, etc.; all these attributes must be defined previously, of course, as *prototype slots*. These sorts of slot represent, normally, the “core” of the definition of a given concept C_k .

For these properties, the concepts C_1, C_2, \dots, C_n that appear in the slots of the frame associated with C_k must be interpreted as *constraints* on the sets of legal fillers (values) that can be associated with these properties *when the concept C_k is specialized or instantiated*. An important point concerns the fact that, in NKRL, these “constraints” can only be expressed using the data types introduced

in Section 2.2.1.2: *integer, double precision, Boolean, string of characters, HClass concept, HClass individual*. As in Protégé, additional “legal” constraints are the “cardinality” constraints; see also Fig. 2.1. This means, in particular, that *specific operators* proper to, for example, a description logics environment that could be used to build up *complex constraints* in the style of (INTERSECTION human_being (UNION doctor_ lawyer_) (NOT.ONE.OF fred_)) *are not admitted here*. This last formal expression can be interpreted as denoting a class of fillers that are men, can be doctors or lawyers, but cannot be Fred. This limitation is not at all disturbing for two main reasons:

- According to the NKRL philosophy, expressions like the above that, *introducing complex relationships about concepts and individuals*, can be considered as pertaining in reality to the “narrative” and “event-specific” domains are best described by using the “descriptive” and “factual” tools, templates and predicative occurrences. This means, in practice, that not only is it possible to represent expressions like the above correctly in NKRL, but also that NKRL is able to encode them in the most appropriate way.
- Avoiding making use of these complex constraints – and avoiding, then, all the theoretical and practical problems linked with the inheritance of role fillers built up in this way – allows one to *steer clear from the exponential complexity problems that affect terminological reasoning* and that, according to the different configurations, may be NP-hard, co-NP-hard, NP-complete, PSPACE-complete; see Bertino et al. [2001: 164–168] in this context.

“Procedure-type” Properties

The “procedure” slots are used in general to store information about the *dynamic characterization* of a concept or individual, e.g. by giving the description of its *typical behavior*, the *instructions for the use*, etc. Classically, this sort of content can be represented as *procedural pieces of code* in the “methods” or “demons” style. A characteristic of NKRL, however, concerns the possibility of describing this “procedural” information in a *declarative* style using, as fillers of the “procedure” slots, *descriptive (templates) or factual (occurrences) structures*. The data type of the “procedure” prototype slots is then always “string of characters.”

For example, *specific complex concepts* can be defined in detail making use of *templates*; an example is given by the concept `norms_for_indirect_transfer_of_revenues_abroad` in the legal domain, where templates have been used for supplying an *operational* description of these norms; see Section 2.3.2.2 (Table 2.20). As a second example we will see, in Appendix B, the use of “declarative” techniques making use of templates for *disambiguating complex “plural” expressions*. Note that the templates used in this context are *partially instantiated templates*, where *at least some of the explicit variables* have been replaced with HClass terms congruent with their constraints; we will return to this point later.

A particularly useful way of utilizing the “procedure” slots is to make use of these slots to store the so-called *HClass occurrences*. The latter take their origin

from the remark that, when examining the narrative information included in news stories, for example, we find a lot of *background information* that is (i) of a *general import*, and really needed for a complete understanding of these stories, (ii) *relatively independent* from the proper “event(s)” related in a specific story, and (iii) *highly repetitive*. If we look, for example, at the news stories inserted in the “Philippine terrorism” corpus used in a recent NKRL application carried out in the context of the PARMENIDES project [Rinaldi et al., 2003; Black et al., 2004], we find – systematically repeated for each of these stories – background information in the style of: “The Abu Sayyaf group is an Islamic separatist group in the Southern Philippines,” “The Abu Sayyaf group routinely performs ransom kidnapping in order to finance its activities,” “The town of Isabela is located on Basilan island,” “Jolo is the capital of the southern Sulu province,” etc.

It is now evident that inserting information in this style *directly into the HClass records for individuals*, like ISABELA_, IOLO_, ABU_SAYYAF_GROUP, etc., *instead of coding it again and again* for each new story, can give rise to a double benefit: (i) *increasing the logical coherence* of the knowledge stored into the system; (ii) *reducing notably the global amount of NKRL code* needed for each application. The fillers of the “procedure” roles represented under the form of *HClass occurrences* are, in this case, (sequences of) *NKRL predicative occurrences*, i.e. the formal representations of the specific “narrative events.” More details can be found in Zarri and Bernard [2004a].

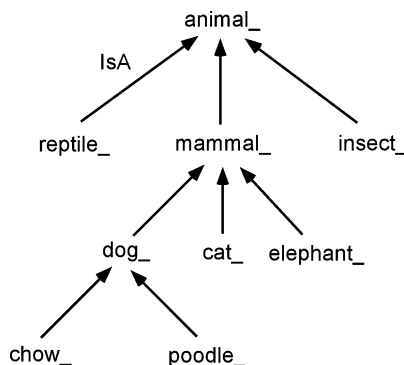
2.2.1.4 NKRL Instances (Individuals)

In Section 2.1, we have already noticed that, in NKRL, some form of *InstanceOf* link must be implemented as the *necessary complement* of *IsA* for the construction of *well-formed HClass hierarchies* of “standard” concepts.

The difference between (*IsA* C_2 C_1) and (*InstanceOf* I_1 C_1) is normally explained in terms of the difference between the two options of (i) considering C_2 as a *subclass of* C_1 in the first case, operator “ \subset ,” and (ii) considering I_1 (an instance) as a *member of* the class C_1 in the second, operator “ ϵ ” – see also the definitions in Table 2.2. Unfortunately, this is not sufficient to eliminate any ambiguity about the notion of instance, which is, eventually, *much more controversial than the notion of concept*. In this section we will discuss briefly the NKRL solutions for two of the main problems concerning the *practical implementation* of instances (“individuals,” enumerative component), namely (i) the possibility of considering *as instances in themselves all the “intermediate” nodes of an ontology* (to the exclusion then of the root), instead of admitting that the instances can only be some “leaves” of the hierarchy; (ii) even limiting the notion of instance to this last interpretation, the possibility of having several levels of instances, i.e. instances of an instance. For simplicity’s sake, we will make use for the discussion below of a fragment of the elementary, well-known ontology relating *elephant_* to *mammal_* and *animal_*; see Fig. 2.3.

With reference now to the first of the two problems stated above, if a very liberal interpretation of the notion of instance is admitted, then CLYDE_ is an

Fig. 2.3 Fragment of the elephant_/mammal_, etc. ontology



instance of `elephant_` but `elephant_` can also be considered, to a certain extent, as an instance of `mammal_`; this is accepted in many object-oriented systems, in the description logics systems, in Protégé, etc. In this latter system, for example, both individuals (instances) and classes (concepts) can be instances of classes: Protégé can then introduce *metaclasses* as classes whose instances are themselves classes. Every class (concept) has a dual identity, it is a “subclass” of another class (its “superclass”) in the normal class hierarchy defined by the `IsA` links, and it is at the same time an “instance” of another class, its “metaclass.” As a class defines a sort of “compelling mold” for its instances, *the metaclass defines a compelling mold for the associated classes*, describing, for example, which specific slots these latter can have, and the constraints for the values of these slots.

This position can be seen – along with the a priori definition of the frame slots as first class objects under the form of “template” or “prototype” slots – as a further answer to the congenital problem proper to any type of object/property-based system (including description logics systems) that concerns *the arbitrariness in the choice of the properties* (in the choice of the slots); see the discussion in Section 2.2.1.2. A solution in this style is surely *elegant*, and can contribute to assuring the logical coherence of the resulting knowledge bases, *but it is not deprived of inconveniences*.

We can note first that, by admitting that also the concepts can be considered as “instances” of other concepts, the logical and semantic properties of the “proper” instances are likely to become *strongly dependent* on the particular choice of primary concepts selected to set up a given inheritance hierarchy. Under this assumption in fact, we can infer that, according to the definition given in Eq. (2.1), the `InstanceOf` relationship should be, like `IsA`, *always transitive*. With reference then to Fig. 2.3, and assuming that the relationship (`InstanceOf FIDO_ poodle_`) holds, the (`InstanceOf FIDO_ animal_`) also holds. But if, in this same figure, we substitute the root `animal_` with the root `species_`, then we can still consider that (`InstanceOf poodle_ species_`) holds, *but it becomes very difficult to assert* (`InstanceOf FIDO_ species_`). More generally, the above

assumption introduces *a considerable amount of epistemological confusion* by equating entities that, like the “concepts,” are *atemporal* – at least in the context of a given application – and *not linked to a specific location* with others, like the “instances,” always characterized by *very precise (even if sometimes implicit) spatio-temporal coordinates*; also see Lenat and Guha [1990: 332–339].

The solutions adopted in NKRL for the introduction of the instances and their association to the HClass hierarchy consist, then, *in adding to the set-oriented definition of an instance a sort of an “extensional” definition in the Woods style* [Woods, 1975]; see Section 2.1. According to this principle, we consider that *all the nodes of a well-formed inheritance hierarchy (ontology) like that of Fig. 2.3 must be considered only as “concepts,”* i.e. general descriptions/definitions of generic *intensional notions*, like that of `poodle_`. When necessary, to each of these nodes can be added an `InstanceOf` link having the meaning of a *specific existence predicate*, e.g. we can declare that a specific, extensional incarnation of the “concept” `poodle_` is represented by the “individual” `FIDO_`. In this way, the introduction of instances becomes strictly a *local operation, to be executed explicitly, when needed, for each node (concept) of the hierarchy*. A consequence of this assumption is represented by the fact that, as already stated, concepts participate in the HClass inheritance hierarchy *directly*; instances participate *indirectly* in this hierarchy *through their parent concepts*.

We can now conclude this section by examining the second of the problems evoked before, i.e. the possibility of creating *instances of instances*. The classical example [e.g. Zarri, 1997] is given by `PARIS_`, an individual (enumerative component) that is an *instance* of the concept `city_`, but that could, at least in principle, be *further specialized through the addition of proper “instances”* (i.e. viewpoints) like “Paris of the tourists,” “Paris as a railway node,” “Paris in the *Belle Epoque*,” etc. According to what was stated above, instances in NKRL are always considered as *terminal symbols*; *this excludes, then, any possibility of implementing instances of instances using this language*. Viewpoints can, however, be easily realized in NKRL according to a solution that goes back to the seminal paper by Minsky [1975] about frames. This solution consists, then, in introducing in the inheritance hierarchy *specialized concepts* like `tourist_city`, `railway_node`, `historical_city` that all admit the individual `PARIS_` as an instance; `PARIS_` *inherits from each of them particular, “bundled” sets of attributes (slots)* like, for example, {`TaxisBaseFare`, `EconomyHotels`, `UndergroundStations`,...} from `tourist_city`, {`TypesOfMerchandise`, `DailyCommutersRate`,...} from `railway_node`, etc.

2.2.2 Descriptive/Factual Data Structures

The data structures used for the descriptive and factual components (templates and occurrences) are more worthy of note than those utilized for the concepts and individuals of the definitional and enumerative components. They are of

the n -ary type; see the discussion in Section 1.2.1. From a formal point of view they show, for example, some similarities with the data structures proper to the *Case Grammars* used in linguistics and computational linguistics [e.g. Fillmore, 1968; Bruce, 1975]. As already stated, however, Case Grammars deal with *linguistic/surface-level* entities, whilst the NKRL descriptive/factual data structures deal with *symbolic/deep-level* entities; see again Rosner and Somers [1980].

2.2.2.1 General Format of the Descriptive/Factual Structures

In opposition then to the *binary* (basically, “attribute-value”) structures used for the frames of the definitional and enumerative component, the descriptive/factual data structures – i.e. those used for *both the “templates” (descriptive component) and the “predicative occurrences” (factual component)*; see later Section 2.3 for the *second-order structures* – are “ n -ary” constructions characterized by the association of “*quadruples*.” These connect together the *symbolic name* of the whole template/occurrence, a *predicate* and the *arguments* of the predicate introduced by named relations, the *roles*; see again the discussion in Chapter 1 and Eq. (1.2), reproduced below for ease of reference. *The quadruples have in common the “name” and “predicate” components.* As already stated, we denote in Eq. (1.2) with L_i the *generic symbolic label* identifying a given template t_i or an occurrence c_i , with P_j the predicate used (like MOVE, PRODUCE, RECEIVE, etc.), with R_k the generic role (slot, case, like SUBJ(ect), OBJ(ect), SOURCE, DEST(ination)) and with a_k the corresponding argument (concepts C_i , individuals I_i , or associations of concepts or individuals):

$$(L_i(P_j(R_1 a_1)(R_2 a_2) \dots (R_n a_n))) \quad (1.2)$$

As we have seen, templates are inserted into an HTemp(lates) hierarchy, where each node represents a template object and which – at the difference of HClass that is (at least in principle) a direct acyclic graph (DAG) – *consists simply of a tree*. Predicative occurrences, i.e. the formal NKRL representations of “elementary events” represent the “leaves” of this tree. Taking into account the fact that templates are nothing else than *formal descriptions of classes of structured events*, HTemp corresponds, therefore, to a *taxonomy of events*. This enlarges the traditional interpretation of ontologies, where only “taxonomies of concepts” are usually taken into consideration.

2.2.2.2 Semantic Predicates, Roles, Templates and Occurrences

One of the main assumptions of NKRL is that, *at least for a large class of nonfictional narrative documents*, the semantic and conceptual structures that convey the narrative knowledge are *limited in number and relatively stable*, so that it is possible to denote these structures by making use of some sort of (partial) *canonical representation*. This confirms, among other things, that the NKRL representation of templates (classes of narrative

events) and occurrences (specific events) is *independent of the different NL surface utterances used to describe these classes or events*; see also the discussion in Section 1.1.2.2.

Semantic Predicates as Primitives

Returning to Eq. (1.2), its central element consists of a *deep semantic predicate* identifying the *basic type of action, state, situation, etc.* that concerns a specific event (predicative occurrence) or a class of events (template) – see again Section 1.1.2.2.

Note that, in conceptual systems that share some similarities with NKRL (like Semantic Networks [Lehmann, 1992] or Conceptual Graphs [Sowa, 1984, 1999]), the semantic predicates can be chosen, *at least in principle, according to any “deep” or “surface” option*. For example, in the “Primitives and prototype” section of his 1984 book, Sowa 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” [Sowa, 1984: 14]. In reality, in the implemented systems, *the “surface” option is largely preferred* – probably, among other things, because *choosing the “deep” option (the “primitive” option) means defining exactly the condition of use of these primitives, i.e. establishing some sort of HTemp “catalogue,” and this can be long and very annoying work*. Making use, on the contrary, of a “surface” approach *can give the impression that this “defining” work is not really necessary, given that the “predicates” used (NL verbs, in practice) have an intuitive meaning in NL*. In the surface approach, however, the other side of the coin concerns *the introduction of an excessive number of degrees of freedom in the representation of complex “entities” like the nonfictional narratives, with the negative effects, e.g. on the possibility of securing the reproduction or the sharing of the results* already mentioned in the discussion about CGs in Section 1.2.2.2.

For the predicates, NKRL has then chosen the “deep” (primitive) option. A long experience with the conceptual representation of all sorts of multimedia, nonfictional narratives has shown that it is possible to make use, in the templates and occurrences, of *only seven “deep” (canonical) semantic predicates corresponding to very general, prototypical categories of the “behavior” of all sorts of human and nonhuman “characters”* – these predicates were five in the RESEDA project; see the Preface. Their conventional labels (along with an *intuitive description* of their “meaning”) are indicated in Table 2.3: in NKRL, then, these seven predicates represent the *only “legal” entities* that can be substituted to the P_j term in Eq. (1.2).

We can note immediately – and this is true in general for all the entities mentioned in Eq. (1.2) like, for example, the “roles” – that the *actual definition of each of these predicates is strictly operational* and is supplied, in practice, by *the fact of being used within a given, exactly identified subset of the templates of the HTemp hierarchy* described in Chapter 3. This means also that (i) each of the structures derived from Eq. (1.2), i.e. the templates (and, accordingly, the

Table 2.3 Semantic predicates in NKRL

Predicate	Mnemonic description
BEHAVE	A character adopts a particular attitude, plays a particular role, or acts (concretely or intentionally) to obtain a given result
EXIST	To be present, also metaphorically, in a certain place; when associated with “temporal modulators” like <i>begin</i> and <i>end</i> , see Section 2.2.2.4, the predicative occurrences built up around this predicate can also be used to represent the “origin” or the “death” of a person, a company, etc.
EXPERIENCE	A character is affected by some sorts of good, bad or “neutral” news or events
MOVE	The displacement of a person or a physical object, the transmission of a message, a change of opinion . . .
OWN	To have, to possess (also metaphorically, e.g. a given entity “has” a particular property)
PRODUCE	Execute a task or an activity, cause to exist or occur (with reference to material or immaterial entities, like the production of a service), etc.
RECEIVE	To acquire, to obtain, also abstract entities like information or advice

predicative occurrences), must be considered *as a whole* and that (ii) the templates constitute the veritable *basic units* to be taken into consideration with respect to the *definition of the entire descriptive/factual domain*.

Note, in this respect, that the reduced set of NKRL “semantic predicates” *could evoke, in effect, the well-known, reduced set of 11 “primitive acts” utilized by Schank* in the early descriptions of his “conceptual dependency” (CD) theory; see Schank and Abelson, [1977] and the discussion in the first chapter. We recall here that, in Schank’s terms, 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’).” This model has been criticized mainly on the basis of the *evident impossibility of reducing always, and in an unambiguous way, all the universe’s complexity to 11 primitive acts*. We must add, for correctness’ sake, that, in the most recent applications, Schank and his colleagues have relied more and more on high-level concepts/predicates, like DISPUTE, PETITION, NEED-SERVICE, AUTHORIZE, LEGAL-CONSULTATION, HAVE-MEDICAL-PROBLEM, etc., instead of expanding everything into primitives [Schank and Abelson, 1977; Schank and Carbonell, 1979; Schank, 1982; Lytinen, 1992].

From an NKRL point of view, a first, very rough reply to this type of criticism (too limited a number of primitives) could be the remark that NKRL is not at all a *universal formalism* – like CD, but also, for example, Sowa’s CGs – but only a *formalism to represent correctly only the (relatively restricted) domain of nonfictional narratives*.

But a more convincing reply can be given with reference to a much more “subtle” type of criticism of the Schank approach expressed, for example, in Wilensky’s [1987] report already mentioned. In this document, Wilensky does not criticize the possibility/opportunity of the decomposition in primitive

terms, but the fact that *this decomposition can be of no utility in many practical situations*. Decomposing “walking” in terms of PTRANS does not exempt the CD users from the need of reconstructing, in some way, the *full concept of “walking” when inferencing is needed*, given that PTRANS has, in itself, a *very loose semantic link* with a very specific notion like that of “walking”; see Wilensky [1987: 10–14].

Wilensky’s remark about *the need of going beyond the restricted conceptual scope of each single Schankian primitive acts* (of each NKRL semantic predicate) complies well with NKRL’s general philosophy. As has been mentioned before, and as will appear even more clearly later when the characteristics of the NKRL’s “catalogue” of HTemp templates are discussed, the “primitive” entities of NKRL are *not at all represented by the seven semantic predicates listed in Table 2.3 – or by the “roles” of Table 2.4. They coincide in reality with (at least) the about 150 (extensible and customizable) templates listed in HTemp and derived by the controlled combinatory of predicates, roles and arguments of the predicates (HClass concepts or combinations of these concepts) expressed according to the format represented by Eq. (1.2).*

We can conclude about “primitives” by noticing that a *hardline support to a whole primitive approach* can be found in the work of Anna Wierzbicka; see her well-known early books [Wierzbicka, 1972, 1981] and the recent book with Cliff Goddard [Goddard and Wierzbicka, 2002]. Wierzbicka’s aim – carried on through her “Natural Semantic Metalanguage” (NSM) research program – is that of finding (making use of “reductive paraphrase” tools) *the smallest set of basic concepts in terms of which all other words, concepts and grammatical constructions can be explicated, and that cannot themselves be explicated in a noncircular fashion*. The (provisional) result is a list of about 60 *semantic primes* regrouped in classes like “substantives” (I, YOU, SOMEONE/PERSON, PEOPLE), “quantifiers” (ONE, TWO, SOME, ALL, MANY/MUCH), “actions and events” (DO, HAPPEN, MOVE), “existence and possession” (THERE IS/EXIST, HAVE), “time” (WHEN/TIME, NOW, BEFORE, AFTER, A LONG TIME, A SHORT TIME, FOR SOME TIME, MOMENT), “space,” “logical concepts,” etc. This work is very interesting; many of these “primes” coincide in general with some “high-level” HClass concepts, and could surely be useful for the set up of all sorts of “conceptual lexica.” Unfortunately, at least for the moment, Wierzbicka and colleagues are not completely clear about the way *to combine these primitives in order to give rise to a sort of “universal syntax/grammar” that could be retrieved in all the possible languages*, and this lessens, then, the interest of her work for NKRL’s (present and pressing) needs.

Introducing the NKRL Roles

As stated at length in Chapter 1, the use of “roles” to link the “semantic predicate” to its (simple or complex) “arguments” (see the terms R_i in Eq. (1.2)) represents one of the *main features that allows NKRL to go beyond the limited*

Table 2.4 NKRL's roles

Role	Acronym	Mnemonic description
<i>Subject</i>	SUBJ	The <i>main protagonist</i> (the “agent,” but also, in case, the “patient”...) of the elementary event (of the class of elementary events). The “filler” (argument of the predicate) of this role is often, but not necessarily, an animate entity or a group of animate entities (e.g. a social body)
<i>Object</i>	OBJ	The entity, animate or not, which is <i>acted upon</i> in the context of the event (the class of events)
<i>Source</i>	SOURCE	The animate entity (group of entities) who is <i>responsible</i> for the particular behavior, situation, state, etc. of the SUBJ of the elementary event (of the class of elementary events)
<i>Beneficiary</i>	BENF	The animate entity (group of entities) who constitutes the <i>addressee</i> (the “recipient,” etc.) of the OBJ mentioned in the event or class of events (or, more generally, the addressee of the global behavior of the SUBJ of the event or class of events)
<i>Modality</i>	MODAL	The (often inanimate) entity (or the process) that is <i>instrumental</i> in producing the situation described in the event or class of events
<i>Topic</i>	TOPIC	The <i>theme</i> (“à propos of. . .”) of the fact(s) or situation(s) that are represented in the event (in the class of events)
<i>Context</i>	CONTEXT	The <i>general context</i> (“in the context of. . .”) of the fact(s) or situation(s) that are represented in the event (in the class of events)

“binary” approach used by many present knowledge representation systems. NKRL's roles are listed in Table 2.4. We can add that:

- “Role” is a *very general notion* used, among other things, both in linguistics and in knowledge representation. Making reference to the classification illustrated, e.g., in [Van Valin, 1999] – where roles are introduced at three different levels of generality, the *verb-specific* semantic roles (like runner, killer, hearer, broken, lover, etc.), the *thematic relations* (like agent, instrument, experiencer, theme, patient), and the two *generalized semantic roles* (actor, undergoer) – and in agreement with the “deep” (conceptual) nature of the language, *NKRL roles can be associated with the “thematic relation” category*.
- However (see the discussion in Section 1.1.2.3), NKRL roles represent simply a *functional/semantic relationship* that holds between a predicate and one of its (simple or complex) arguments and that is *strictly necessary in itself for the full appraisal of the “meaning” to be represented*. This means that (at the difference, for example, of what happens with Jackendoff's [1990] “thematic roles – see again the discussion mentioned above), the decision of making use of an NKRL particular role is *totally independent from any surface structure (any syntactic) consideration*. NKRL roles, then, are eventually more similar to the “correlators” of Ceccato's operational

linguistics [Ceccato, 1961, 1967] – where to each correlator corresponds a set of *mental operations* – than to the (syntactically constrained) Jackendoff’s “thematic roles.”

As it appears from Table 2.4, there are then *seven roles* in NKRL, that are used to identify the *participants* in a given elementary event or class of events. They, like the semantic predicates, are *primitive* – with the *caveats already expressed about the primitive character of the semantic predicates*, see also below and the next Section – and, like the predicates, are identified by means of *symbolic names*. In this respect, NKRL’s position is, once again, opposed to that of Jackendoff [1990: 44–58], who refuses to attribute standard names to its thematic roles – even if, in practice, he makes use of the usual “Agent, Source, Goal . . .” terms.

An important point to emphasize is that the NL descriptions that appear in Table 2.4 are there, as well as those of Table 2.3, *only to suggest an intuitive explanation of the function of the NKRL roles, and do not constitute at all a “definition” of these roles*. Their real definition is given, once again, *operationally*: (i) by their mandatory presence/optional presence/absence in the *formal descriptions of the different templates*; (ii) by *the set of constraints* that are explicitly associated with the roles in the templates and that define the *classes of their legal fillers*. As already stated at length, the templates constitute, in a sense, the real “primitives” of the NKRL language. Ambiguities that can rise by examining the NL descriptions of Table 2.4 are then settled by examining the *formal definitions* of the templates of the HTemp hierarchy. For example (see Chapter 3), the “property” to be explicitly declared in templates like `Own:SimpleProperty` and `Own:CompoundProperty` is *necessarily associated as filler with the TOPIC role*, ruling out, then, any possible ambiguity about the use in this context of the MODAL (or CONTEXT) role.

The “fillers” of the seven roles listed in Table 2.4 – i.e. the a_i terms in Eq. (1.2), the “arguments” of the semantic predicate – are represented, in the *templates*, by *variables and constraints on the variables* (see next section) and by *HClass terms, concepts and individuals, in the predicative occurrences*. Note that these arguments can be “simple,” i.e. represented by a unique variable or HClass term, or “structured.” Structured arguments, called “expansions” or “complex fillers” in NKRL terms, *are built up by combining HClass terms or variables according to a precise syntax*; this topic will be discussed in detail in Section 2.2.2.3

The NKRL Templates and the “Catalogue”

The (single) “semantic predicate,” the seven “roles” and the “arguments” are the three *basic building blocks that make up a template t_i* – and, therefore, a predicative occurrence (instance of template) c_i . *These three blocks cannot, separately, receive an interpretation in terms of classes of meaningful events; an (at least partial) valid interpretation will only arise after their (mandatory) assembling has been carried out*. An immediate corollary of the above is that *the presence of at least a filled role in a template is a necessary (but not sufficient) condition in order that the template*

be meaningfully interpreted – note that, in NKRL, the role SUBJ(ect) must *necessarily be filled in any possible template or predicative occurrence*; see also the discussion about the “Kimian-like events” in Section 1.1.2.1.

In a template/predicative occurrence, the single arguments, simple or structured (expansions), and the template/occurrence as a whole, may be characterized by “determiners” (attributes) that introduce further details about their significant semantic aspects. Note that, in opposition to what happens with the predicates, roles, and arguments, the determiners are never strictly necessary for the interpretation of a template (of a predicative occurrence) in terms of a general description of a class of meaningful events (of an event). For example, templates and occurrences may be accompanied by “modulators” (like “nonintentional,” “social,” “possible”) that, as their name suggests, are there to refine or modify the basic interpretation of the template or occurrence; moreover, predicative occurrences are necessarily associated with the two “temporal” attributes *date-1* and *date-2*. Examples of determiners that, on the contrary, can only be associated (through the “external” operator “:.”) with the fillers (arguments of the predicate) of the SUBJ, OBJ, SOURCE and BENF roles are the “location” attributes. The determiners are described in detail in Section 2.2.2.4.

The general scheme of a template or predicative occurrence is shown in Table 2.5. In reality, the temporal attributes indicated in this table are *never associated with specific templates*, given that these latter represent *atemporal* categories of the NKRL language. On the contrary, they are *always associated with the predicative occurrences*, even if these attributes can be “empty” in some specific cases; see Chapter 3.

The format of Table 2.5 is called “external NKRL format,” in contrast with the “*internal format*” concretely used to store and process the NKRL structures, see Zarri and Bernard [2004b: appendix A]. This internal format is normally called the “*Béatrice format*” in NKRL jargon. Note that, in the examples of coding inserted in the following sections, the roles will be *omitted* in the external representations of templates and occurrences, for simplicity’s sake, *when the corresponding fillers (arguments of the predicate) are “empty.”* In reality, all seven roles are always present in the internal format that is, fundamentally, a sort of “positional” format.

Table 2.5 External format of an NKRL template/predicative occurrence

PREDICATE	SUBJ	{<argument> : [location]}
	OBJ	{<argument> : [location]}
	SOURCE	{<argument> : [location]}
	BENF	{<argument> : [location]}
	MODAL	{<argument>}
	TOPIC	{<argument>}
	CONTEXT	{<argument>}
	[modulators]	
	[temporal attributes (missing in the templates)]	

Returning now to the three basic building blocks (*predicate, roles, arguments*) that make up a template/predicative occurrence, if we combine the *seven predicates* with the *seven predicative roles*, the *several hundred upper level concepts* of the HClass hierarchy (see Chapter 3: these high-level concepts are practically *invariable* given that they are used to define the constraints on the arguments of the templates – see Tables 2.6 and 2.7), the determiners, etc., we obtain a (*very large*) *solution space where all the legal templates are in principle represented*. In reality, these possible combinations are *filtered* making use of pragmatic rules like: “the OWN templates must necessarily provide for an OBJ(ect) role and, in their Own:Property variant, cannot be endowed with a BEN(e)F(iciary) role”; “all the EXIST templates of the “origin or death” sub-hierarchy require the presence

Table 2.6 Building up predicative occurrences

(a)		
<i>name:</i> Produce:Violence		
<i>father:</i> Produce:PerformTask/Activity		
<i>position:</i> 6.35		
<i>NL description:</i> “Execution of Violent Actions on the Filler of the BEN(e)F(iciary) Role”		
PRODUCE	SUBJ	<i>var1</i> : [(<i>var2</i>)]
	OBJ	<i>var3</i>
	[SOURCE	<i>var4</i> : [(<i>var5</i>)]]
	BENF	<i>var6</i> : [(<i>var7</i>)]
	[MODAL	<i>var8</i>]
	[TOPIC	<i>var9</i>]
	[CONTEXT	<i>var10</i>]
	{[modulators],	+(abs)}
	<i>var1</i> =	human_being_or_social_body
	<i>var3</i> =	violence_
	<i>var4</i> =	human_being_or_social_body
	<i>var6</i> =	human_being_or_social_body
	<i>var8</i> =	violence_, weapon_, criminality/violence_related_tool, machine_tool, general_characterizing_property, small_portable_equipment
	<i>var9</i> =	h_class
	<i>var10</i> =	situation_, spatio/temporal_relationship, symbolic_label
	<i>var2, var5, var7</i> =	geographical_location
(b)		
mod3.c5) PRODUCE	SUBJ	(SPECIF INDIVIDUAL_PERSON_20 weapon_wearing (SPECIF cardinality_several_)): (VILLAGE_1)
	OBJ	kidnapping_
	BENF	ROBUSTINIANO_HABLO
	CONTEXT	#mod3.c6
	date-1:	1999-11-20
	date-2:	
Produce:Violence (6.49)		
<i>On November 20, 1999, in an unspecified village (VILLAGE_1), an armed group of people has kidnapped Robustiniano Hablo</i>		

Table 2.7 The syntax of the constraint expressions

$var_i = h_class_term$

The *simplest form of constraint*, e.g. see the constraints on all the variables of Table 2.6 apart from the constraints on *var8* and *var10*. We will suppose first that *h_class_term* represents a *concept*. In this case, the constraint says simply that *in all the occurrences derived from the specific template where var_i occurs*, the value bound to var_i must be a *specific term*, concept or individual (with respect to the HClass hierarchy) of the concept *h_class_term* that represents the constraint. If *h_class_term* represents an *individual*, then the value associated with var_i must be an individual *strictly identical* to the *h_class_term*

$var_i = h_class_term_1, h_class_term_2, \dots, h_class_term_n$

NKRL constraints are frequently expressed using this syntax, e.g. see the constraints on *var8* and *var10* of Table 2.6. The “comma” operator “,” represents here an “*exclusive or*”; i.e. in the occurrences, the value bound to var_i must be a *specific term*, concept or individual of *one of the listed concepts to the exclusion of all the others*. If some of the *h_class_term_i* constraints represent an *individual*, then the identity holds only if the value associated with var_i is an individual *strictly identical* to this constraint

$var_i \neq h_class_term_1, h_class_term_2, \dots, h_class_term_n$

In this case, the “comma” operator “,” represents an “*and*”; i.e. in the occurrences derived from the template where var_i occurs, the value bound to var_i *must be different from all the listed constraints $h_class_term_i$, including the specific terms, concepts and individuals, of all these constraints*

$var_i = var1, var2, \dots, var_n$

The “comma” operator “,” represents here an “*and*.” This syntax means that the values (concepts or individuals) assumed by *var1*, *var2*, ..., *var_n* in the occurrences derived from the template where var_i occurs must be *strictly identical* to the value assumed by var_i . Note that, in the concrete templates/inference rules, the variables *var1*, *var2*, ..., *var_n* are, normally, *introduced before the variable var_i comes into sight*

$var_i \neq var1, var2, \dots, var_n$

In this case, too, the “comma” operator “,” represents an “*and*”; the value (concept or individual) assumed by var_i must be *strictly different* from the values assumed by *all the variables var1, var2, ..., var_n*

$var_i = symbolic_label_1, \dots, symbolic_label_n$

This syntax (a variant of the second expression of the list) means that the value of var_i must be the *symbolic label of a predicative occurrence*; see the *completive construction* in Section 2.3.1. The operator “,” represents an “*exclusive or*”

$var_i = h_class_term_1, \dots, h_class_term_n, individual_$

This expression means that the value of var_i must be, at the same time, *an individual and a specific term, concept or individual of one of the $h_class_term_1, \dots, h_class_term_n$ constraints*. The “comma” operators “,” represent an “*exclusive or*” (see the second expression of this table), *with the exception of the last operator, which represents an “and”* (the value must be also an individual)

of a temporal modulator”; “the displacements of a person or group of persons, predicate MOVE, are always expressed in the form of a SUBJ(ect) who moves himself as an OBJ(ect),” etc. This filtering leads, eventually, to a *limited number* of templates, actually *about 150* –see Chapter 3 and the full description of the HTemp hierarchy in Zarri [2003a]. As already stated, HTemp is a *tree*; we can

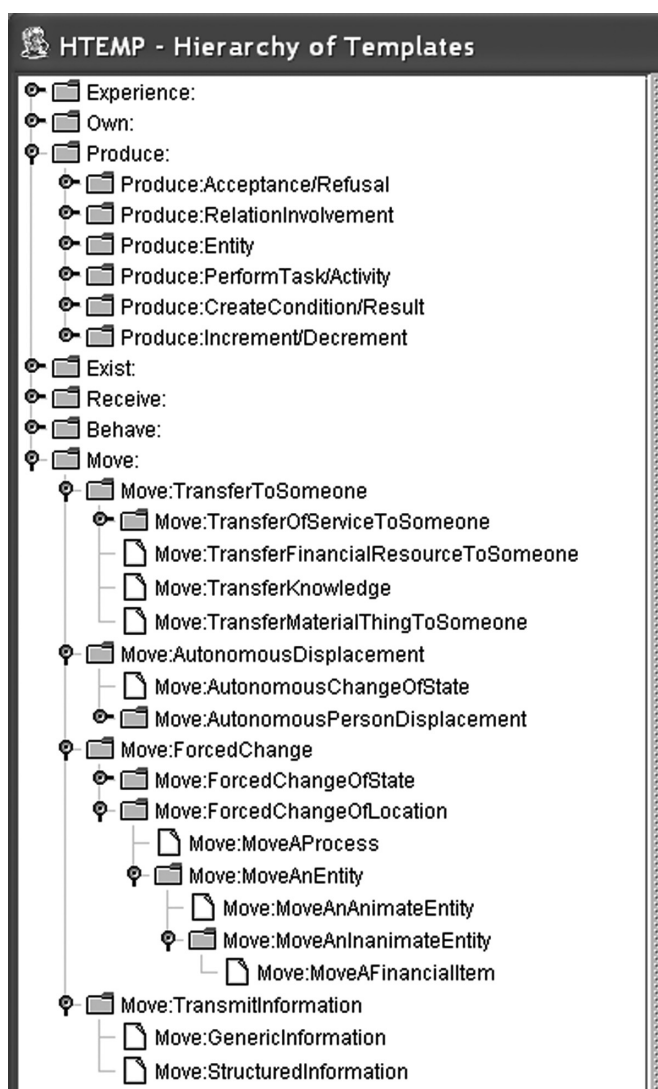


Fig. 2.4 HTemp fragment illustrating (part of) of the MOVE (and PRODUCE) branch(es)

add that it is structured into *seven branches*, where each branch corresponds to one of the *seven semantic predicates* accepted by the language and described in Table 2.3, i.e. *each branch includes only templates built up around a specific predicate*. Figure 2.4 reproduces a partial list of the templates included in the MOVE (and PRODUCE) branch(es) of Htemp; see Chapter 3. The HTemp hierarchy coincides, then, with the *catalogue of the NKRL templates*.

We must add that, when we speak of a “catalogue,” this does not mean that this last must be *immutable and definitive*. The HTemp catalogue is, on the contrary, a *living structure* where it is always possible to insert new elements. For example, *by using (mainly) specialization (customization) operations concerning the classes of possible role fillers*, we can obtain, from one of the present templates, all the (specific) “derived” templates concretely needed for the different, practical applications. From a template like Move:MoveAProcess (Fig. 2.4) [Zarri, 2003a], we can obtain, for example, a “derived” template like “move an industrial process” and the predicative occurrences – e.g. “move, in a well-defined spatio-temporal framework, this particular industrial production” – that correspond to the description of events in the style of: “Sharp Corporation has shifted production of low-value personal computers from Japan to companies in Taiwan and Korea.” When a specific NKRL application has been *completely defined and it is routinely running*, it is then possible to evaluate the “derived” templates specifically introduced for this application, choosing among them those that can be considered of a *general interest*. These latter templates will then be added to the catalogue.

From a formal point of view, since the templates represent the *basic defining entities* of the language, they can also be considered as the “axioms” of NKRL, identifying then *which classes of nonfictional narratives can be concretely dealt with and according to which modalities of use*; this explains why the “catalogue” *can be considered as part and parcel of the definition of the language*. Eventually, we can note (from a “theoretical linguistics” point of view and taking for granted that the templates are the *real “primitives” of the NKRL language*) that they can be interpreted in the context of those *verb semantic classes* that (making abstraction from minor theoretical divergences) can be found, for example, in the work of Cook [1979: 63–65], Jackendoff, [1990], Van Valin [1993: 39], and Levin [1993]. In this context – with the usual caveat about the fact that *NKRL is interested in the “deep” conceptual level of the narrative material taken into consideration and not in any theory concerning the syntax/semantics relationships* – we are particularly sympathetic with Beth Levin’s *pragmatic work*.

Deriving a Predicative Occurrence from a Template

To represent a simple narrative like “On November 20, 1999, in an unspecified village, an armed group of people has kidnapped Robustiniano Hablo,” we must first select the HTemp template corresponding to “execution of violent actions”; see Table 2.6a. The selection is realized (automatically or manually) on the basis of “deep predicate” considerations like those expounded in Section 1.1.2 (Kimian and Davidsonian analysis). In our example, the selected template is a specialization (see the “father” code in Table 2.6a) of the particular PRODUCE template corresponding to “perform some task or activity”; also see Fig. 2.4. The particular narrative to be represented is extracted from one of the (declassified) new stories dealt with, in NKRL’s terms, in the context of the PARMENIDES

project already mentioned; these stories have been supplied by the Greek Ministry of Defence (MoD), one of the PARMENIDES partners.

As appears clearly from Table 2.6a, in a template the *arguments of the predicate* (the a_k terms in Eq. (1.2)) are represented by *variables with associated constraints* – which are expressed as HClass concepts or combinations of HClass concepts. The syntax of the “constraint expressions” is explained in Table 2.7. When creating a predicative occurrence like mod3.c5 in Table 2.6b, *the role fillers in this occurrence must conform to the constraint of the father template*. In the occurrence mod3.c5, for example, ROBUSTINIANO_HABLO (the “BEN(e)F(iciary)” of the kidnapping) and INDIVIDUAL_PERSON_20 (the unknown “SUBJ(ect),” actor, initiator of this action) are both instances of individual_person – a specialization of human_being_or_social_body; see, in Table 2.6b, the constraints on variables *var1* and *var6*. kidnapping_ is a specialization of violence_ ; see *var3*, etc.

In a template, optional elements are in square parentheses. This means, for example, that filling the SOURCE, MODAL, TOPIC and CONTEXT roles of Table 2.6a *is not mandatory* in the predicative occurrences derived from the Produce:Violence template. Also, the presence of individuals/concepts that can replace the “location variables” (*var2*, *var5* and *var7* in Table 2.6a) is not compulsory in these occurrences: as already stated, the “location attributes” are linked with the predicate arguments by using the colon operator “:”; also see Section 2.2.2.4. Possible “forbidden” elements (e.g. a particular role that cannot exist for a given template) are marked as “+()”; for example, in Table 2.6a, the code {[modulators], +(abs)} means that several “modulators” (see Section 2.2.2.4) can be associated with the occurrences derived from the template, to the exception of the modulator abs(olute).

With respect now to the occurrences, a *conceptual label* like mod3.c5 in Table 2.6b represents the *symbolic name used to identify a specific predicative occurrence* c_i . The use of this sort of “pointed” notation is not, in principle, strictly mandatory; it is, however, *strongly recommended*, and it has been systematically used in all the later applications of the NKRL software. The first component of the “pointed” notation identifies the *original document* to be represented into NKRL format, in this case the third news story of the MoD corpus. The second component, c5, tells us that the specific predicative occurrence is the *fifth* within the set of predicative and binding (see Section 2.3) occurrences *that represent together the NKRL “image” of the complete original narrative document*. This formal image is called both the *metadocument* or (more often) the *conceptual annotation* associated with the original document; see again Section 2.3 and Chapter 3.

In the mod3.c5 predicative occurrence of Table 2.6b, the “filler” of the SUBJ(ect) role is a *structured argument* (expansion) that makes use of the “attributive operator,” SPECIF(ication), one of the four operators that make up the specific *AECS sub-language* – this sub-language, and the rules for building up *well-formed expansions*, is detailed in Section 2.2.2.3. This structured argument means that the kidnapping has been realized by a group,

(SPECIF cardinality_ several _), of unknown individuals that are *collectively identified* as INDIVIDUAL_PERSON_20; (SPECIF cardinality_ several _) is one of the typical “NKRL idioms” (see Table 3.2) that is used to represent the *cardinality of sets of totally undefined size*, like those corresponding to *generic plural referents* as “men” or “books.” The unknown individuals were *armed*, *weapon_wearing*. *weapon_wearing* is a specialization of the *dressing_attribute* concept of HClass; this corresponds to saying that, via the generalizations *dressing_attribute* and *physical_aspect_attribute*, it also corresponds to a specialization of *animate_entity_property*; see Section 3.1.2.1.

This particular structured argument is associated with a *location attribute*, represented by the individual VILLAGE_1 in Table 2.6b. Other attributes are the two *temporal attributes* date-1 and date-2 that materialize *the temporal interval associated with the elementary narrative event represented by the predicative occurrence*. The syntax/semantics of the attributes is explained in Section 2.2.2.4, where a detailed description of the methodology for representing temporal data in NKRL is also supplied.

2.2.2.3 Structured Arguments and the AECS Sub-language

The tools for representing structured arguments like the “filler” of SUBJ in Table 2.6b discussed above are of particular importance in NKRL, where they are used, among other things, for implementing a wide-ranging representation of *plural entities and expressions*; see Appendix B.

The AECS Sub-language

In NKRL, structured arguments are built up in a *principled way* making use of a *specialized sub-language*, AECS, which includes four *expansion operators*, the *disjunctive operator* (ALTERNative = A), the *distributive operator* (ENUMeration = E), the *collective operator* (COORDination = C), and the *attributive operator* (SPECIFication = S). Their definitions are given in Table 2.8.

From a formal semantics point of view, we can note, for example, that

$$(\text{SPECIF } e_i \ a \ b) = (\text{SPECIF } e_i \ b \ a) \quad (2.3)$$

$$(\text{ENUM } e_1 e_2) = (e_1 \wedge e_2 \wedge \neg(\text{COORD } e_1 e_2)) \quad (2.4)$$

Equation (2.3) says that, within a SPECIF list, the order of the properties *a*, *b*, ... associated with an entity *e_i*, concept or individual, is *not significant*. Equation (2.4) enunciates in a more formal way what is already stated in Table 2.8: the main characteristics of the ENUM lists are linked with the fact that the entities *e₁*, *e₂*, ... *take part necessarily* in the particular relationship between the structured argument and the predicate which is expressed by the role that introduces the arguments, but they satisfy this relationships *separately*.

Table 2.8 NKRL operators for structured arguments (expansions)

Operator	Acronym	Mnemonic description
<i>Alternative</i>	ALTERN	The <i>disjunctive operator</i> . It introduces a list of arguments, namely concepts, individuals or lists labeled with <i>different</i> expansion operators. <i>Only one</i> of the arguments of the list takes part in the particular relationship, with the predicate defined by the role-slot to be filled with the expansion; however, this <i>particular argument is not known</i>
<i>Coordination</i>	COORD	The <i>collective operator</i> . All the elements of the list (concepts, individuals or lists labeled with <i>different</i> expansion operators) take part (<i>necessarily together</i>) in the relationship with the predicate defined by the role-slot
<i>Enumeration</i>	ENUM	The <i>distributive operator</i> . Each element of the list (concepts, individuals or lists labeled with <i>different</i> expansion operators) satisfies the role–predicate relationship, but they do so <i>separately</i>
<i>Specification</i>	SPECIF	The <i>attributive operator</i> . This is used to associate a list of “attributes” (properties), under the form of concepts, individuals or other <i>SPECIF</i> lists, with the <i>concept or individual</i> that constitutes the <i>first term of the list</i> . This allows us to characterize this last element better. Note that each property appearing inside a <i>SPECIF</i> list can be <i>recursively associated</i> , in <i>turn</i> , with another <i>SPECIF</i> list

From a “pragmatic” point of view, it can be useful to conceive the *SPECIF*(ication) operator as frequently used to translate “adjectives” *that pertain to the “qualifying” and “possessive” grammatical categories*. However, as usual in NKRL – and as a confirmation of the fact that purely “linguistic” considerations are often misleading in the context of a “deep level,” “conceptual” type of representation – only *the (mandatory) syntax proper to the different templates* (and, in particular, *the restrictions associated with the different roles*) must be taken into account to solve the *possible ambiguities*. For example, “John has got a civil servant employment” – where “civil servant” could be considered as a “qualification” of John’s post – is translated in reality making use of the *Behave:Role* template (see Section 3.2.2.1) where *civil_servant* fills the *MODAL* role. In all the *Own:Property* templates (Section 3.2.2.5), the “qualifying, etc. properties” associated with a given “inanimate entity” fill systematically the *TOPIC* role, and so on.

Because of, among other things, the possibility of setting up embedded lists – see, in particular, the recursive nature of the *SPECIF* structures – *an unruly utilization of the AECS operators could give rise to very complex expressions, difficult to interpret and disentangle (unify)*. Therefore, the definitions of Table 2.8 are used in association with the so-called “*priority rule*,” which can be visualized by using the following expression:

$$(\text{ALTERN}(\text{ENUM}(\text{COORD}(\text{SPECIF})))) \quad (2.5)$$

Table 2.9 An example illustrating the “priority rule”

ex.c1)	EXIST	SUBJ	(ALTERN (ENUM MR_BROWN (COORD MR_SMITH JENNIFER_SMITH)) (ENUM MR_BROWN (COORD MR_SMITH LUCY_SMITH))): (WHITE_HOUSE)
		CONTEXT	RECEPTION_15

Exist:HumanPresentAutonomously (3.2122)

This expression is to be interpreted as follows: it is *forbidden* to use within the scope of a list introduced by the binding operator B_i a list labeled in terms of one of the binding operators *appearing on the left* of B_i in the priority expression above., e.g. it is impossible to use a list ALTERN within the scope of a list COORD.

The *strict and mandatory* application of this rule in NKRL implies that it is sometimes necessary to *duplicate* some parts of a complex expansion expression in order to comply with Eq. (2.5). For example, a narrative elementary event like: “Mr. Brown attended the reception at the White House; Mr. Smith also there, accompanied by his daughter Jennifer (but the accompanying lady could also be his wife Lucy),” will be translated in NKRL according to the representation given in Table 2.9 (note that Mr. Brown and Mr. Smith are supposed they went separately to the reception, as shown by the use of ENUM). Of course, each of the single coding elements (MR_BROWN, etc.) within the expansions could be further specified by a proper SPECIF list.

Note that an important consequence of the priority rule is that *a SPECIF list can only be linked with a specific NKRL term, concept or individual, and not with a list*. Moreover, a location associated with an expansion formed of a SPECIF list refers, obviously, to the first element of the list, i.e. to the term that is “specified.”

Sophisticated additions to the basic system of the AECS operators have been proposed in Zarri and Gilardoni [1996]: they will be discussed in Section 4.1.1.3.

2.2.2.4 Determiners (Attributes)

As already stated (see Section 2.2.2.2 and Table 2.5), *single arguments* of a template/predicative occurrence, or *templates/occurrences as a whole*, may be characterized by *determiners* (attributes) that (i) *introduce further details/precisions* about the “meaning” of these arguments or templates/occurrences, but that (ii) are *never strictly necessary for their basic semantic interpretation in NKRL terms*. We can immediately note, however, that the presence of the two *temporal determiners is mandatory* in order that we can *produce well-formed predicative occurrences* (factual component) – even if, as we will see, these temporal determiners can be “empty.” NKRL determiners can be conceived, at least partly, as the “deep-level” (conceptual) counterparts of the surface “adverbials,” e.g. see Austin et al. [2004] for recent research on this topic.

In this section, we will focus on three classes of determiners: *modulators*, *location determiners*, and *temporal attributes*). A fourth class of determiners is represented by the *validity attributes*: in the most recent NKRL applications, *only two validity attributes have been concretely used, i.e. the “contradictory validity attribute” and the “uncertainty validity attribute.”*

The *contradictory attribute*, represented in external coding by the “exclamation mark,” code “!,” is used to make known that information expressed in a given NKRL structure (in practice, in an occurrence c_k) *contradicts* information expressed in another structure (in another occurrence c_l). In this case, the symbolic labels of the two structures (occurrences) become, respectively, $!c_k(c_l)$ and $!c_l(c_k)$, i.e. *the list including the labels of all the occurrences that are in contradiction with c_l is directly associated with the symbolic label of c_l* . The *uncertainty attribute*, code “asterisk” “*,” may characterize either an occurrence c_i as a whole or single concept C_i or an individual l_i . In particular, the uncertainty attributes are *necessarily associated* with all the occurrences that describe future or possible events; see Section 2.3.2.1 and Chapter 3.

Modulators

Modulators apply to a *full template or occurrence*. This means that, to understand the meaning of a template/occurrence in the presence of modulators, we must first think about the *basic meaning* of this template/occurrence *without modulators* and then *particularize* such meaning according to the modulators used. Modulators can evoke, to a certain extent, the (*universal and standardized*) primitive “lexical functions” proper to the meaning–text theory of Igor Mel’čuk, like Magn (intensification), Loc (standard location), Bon (approval, appreciation), AntiBon (disapproval), Oper, Func, Labor, etc. and their variants [e.g. Žolkovskij and Mel’čuk, 1967; Mel’čuk, 1996]. Note once again, however, that NKRL is not a linguistic theory, but a practical tool restricted to the representation and management of nonfictional narrative information.

NKRL modulators pertain to three categories: *temporal modulators*, *deontic modulators* and *modal modulators*.

Temporal Modulators

The definitions for the temporal modulators are given in Table 2.10. They are used to represent the *start* or *end points* of given elementary events, or the *observation* that, at a given point in time, *a specific event is running*. Note, however, that their *operational meaning* is strictly associated with that of the *temporal determiners* (*temporal attributes*); see below.

Deontic Modulators

Deontic modulators were integrated in NKRL during the NOMOS project [Zarri, 1992b] to deal with subsets of the French “General Taxation Law” – namely the norms used to settle cases concerning the transfer of revenues

Table 2.10 Temporal modulators

Temporal modulator	Acronym	Mnemonic description
<i>begin</i>	begin	In the time interval associated with the elementary event described in a predicative occurrence, we distinguish a particular point, <i>the timestamp (date) identifying the beginning of the event</i> , e.g. “On April 5th, 1982, Francis Pym is appointed Foreign Secretary by the British Prime Minister”
<i>end</i>	end	The timestamp (date) marking the <i>end</i> of an elementary event is identified
<i>observe</i>	obs	We identify a <i>specific timestamp</i> where a specific event can be <i>observed</i> . <i>obs</i> will be used, for example, to represent the information “It is known that, in June 1903, Lev Trotsky still agreed with the Mensheviks about the strategy toward socialism.” This situation is attested for June 1903, but we cannot give, at this level, any information about its duration that, in reality, <i>extends in time before and after the given date</i>

abroad. They have been systematically used afterwards for representing “*legal-oriented*” narrative information; a recent paper on this topic is by Zarri [2007]. Their definitions are given in Table 2.11.

The deontic modulators of NKRL satisfy the logical relationships of Table 2.12, which allow us to reduce all of them to the modulator *perm*.

This *deontic* system is very simple, for example, with respect to the formal analysis contained in Lomuscio and Nute [2005]. Associated, however, with HClass sub-trees particularly relevant from a “legal” point of view, like *beliefs_*, *guiltiness_*, *innocence_*, *plea_*, *violence_*, etc., it can compare favorably with the metadata system described in Gangemi et al. [2003] and describe correctly the most simple legal situations.

Table 2.11 Deontic modulators

Deontic modulator	Acronym	Mnemonic description
<i>faculty</i>	fac	Power of doing or not doing <i>by autonomous choice</i>
<i>interdiction</i>	interd	To forbid in a <i>formal or authoritative manner</i>
<i>obligation</i>	oblig	Someone is <i>obliged to do or to endure something</i> , e.g. by authority of law, by moral authority, or according to an autonomous need
<i>permission</i>	perm	“ <i>To be authorized to ...</i> ,” i.e. having the right to do something where the right is granted by an external intervention

Table 2.12 Logical relationships among deontic modulator

$\text{perm}(x)$
$\text{oblig}(x) = \neg(\text{perm}(\neg(x)))$
$\text{fac}(x) = \text{perm}(x) \wedge \text{perm}(\neg(x))$
$\text{interd}(x) = \neg(\text{perm}(x))$

Modal Modulators

If the NKRL determiners correspond, in general, to surface adverbials, then modal modulators can be considered as the “deep-level” counterparts of some specific “modal adverbials” [e.g. Shaer, 2003]. The modal modulators actually accepted by the NKRL software are simply listed in Table 2.13. The semantic properties of the most important among them (e.g. *wish*, *poss*) will be examined in correspondence with *their utilization in the context of particular templates/ occurrences*; see Chapter 3.

Concurrent Utilization of Several Modulators

Several modulators *can be associated with the same occurrence*, as in this fictitious example: “Mr. Smith could renounce to go to Paris on the condition. . .” To represent correctly the first part of this narrative, two modal

Table 2.13 Modal modulators

Modulator	Acronym	Mnemonic description
<i>absolute</i>	abs	Used within EXIST occurrences to indicate the creation or disappearing (birth, death) of an individual or social body
<i>agreement, opposition</i>	for, against	E.g. “to manifest a favorable/negative attitude with respect . . .”
<i>denied event</i>	negv	The reality of a particular event is denied (“Mary did not get the book from John”)
<i>desire, intention</i>	wish	E.g. “Mary would like to be informed about. . .”
<i>firstly</i>	first	The event is the first of a series of events
<i>intentional, unintentional</i>	int, nint	E.g. “Mr. Smith has, voluntarily/involuntarily, started a fire”
<i>leadership</i>	lid	A character acts as a leader, e.g. “Mr. Smith took control of the demonstration for peace”
<i>mainly</i>	main	E.g. “the plant will mainly make high value-added products”
<i>mental</i>	ment	The activity of the SUBJ(ect) does not give rise to a concrete manifestation in the physical domain (e.g. a plan is only conceived, but not necessarily executed)
<i>multiple</i>	mult	An event takes place several times within the time period associated with the occurrence
<i>necessary</i>	necs	E.g. “required to fulfill something”; necs does not imply any coercion or obligation by force of law; see oblig in Table 2.11
<i>possibility</i>	poss	The event represented in the occurrence <i>could</i> exist, or by itself or as a consequence of another event
<i>repeat</i>	rep	A particular event already happened in the past (e.g. a politician has been re-elected)
<i>secretly</i>	krypt	The activity of the SUBJ is hidden, e.g. a clandestine meeting
<i>social</i>	soc	The activity of the SUBJ concerns her/his socio-professional duties
<i>virtuality</i>	virt	A situation, e.g. the nomination to a public function, that should be true in principle, even it is not yet realized

modulators, *negv* and *poss* (see Table 2.13), should be added to the “basic” predicative occurrence representing the displacement (MOVE) of Mr. Smith to Paris. The first one, *negv* (denied event), is needed to represent the “negation” of the event representing the journey; the second, *poss* (possibility), is needed to say that the fact of renouncing to the journey is only a possibility linked to the realization of a “condition” expressed in a second occurrence.

The possible, simultaneous presence of several modulators could introduce a problem of *scope ambiguity*, see the well-known examples in the style of “*x*{*perm*, *negv*} go to work yesterday” opposed to “*x*{*negv*, *perm*} go to work yesterday.” In NKRL, *scope ambiguity is strictly forbidden*. As already stated, when only a modulator is present, it works as a *global operator that takes as its argument the whole predicative occurrence*. When a list of modulators is present, *they apply successively to the occurrence in a prefix notation (polish notation) way*. Therefore, the presence of a list {*perm*, *negv*} in the previous example means that *x* has been permitted not to go to work yesterday (*negv* is applied first, and then *perm*), while the presence of the list {*negv*, *perm*} means that *x* has not been permitted to go to work yesterday. In the “Mr. Smith” example at the beginning of this Section, the correct sequence for the two modulators is then {*poss*, *negv*}. Note that:

- NKRL *good coding practices* recommend to make use, whenever possible, of a *single modulator for each predicative occurrence* – it is normally better to duplicate a predicative occurrence than to stack up several modal modulators. However, *particular combinations of modulators, in a prearranged order, are directly imposed as mandatory by the syntax of a few templates*; see Chapter 3. As an example, we can mention here the use of EXIST templates to represent the creation or dissolution of a social body (e.g. a company), where the association {*abs*, *begin/end*} (in this order) must be *necessarily used*. Note also that the temporal modulators can be *freely associated* with the modal/deontic ones, normally as the “more external ones”; see {*begin*, *wish*} or {*end*, *soc*} – it is important to realize that *the temporal modulators are mutually exclusive*.
- As appears clearly from the above example, many of these “concurrency” problems *are linked with the presence of the modulator negv* – for some troubles and possible solutions that concern dealing with “negation” in a conceptual representation context, see, among many others, the recent paper by Mugnier and Leclère [2007]. From an NKRL point of view, we can note that *negv* is a “historical heritage” of the RESEDA project and that, with the growth of the number of conceptual entities present in both the HTemp and HClass hierarchies, it is used less and less, *even outside a strict “multiple modulators” context*. For example, “Bill is too busy, and he doesn’t have time to exercise” will be “translated” using an Experience: NegativeHuman/Social template (see Section 3.2.2.3), where BILL_, as a SUBJ, EXPERIENCE(s) an impediment_ (OBJ) about (SPECIF finding-time gym_exercising), which represents the TOPIC; the CONTEXT is given

by (SPECIF commitment_ (SPECIF cardinality_ many_)). “Mary doesn’t want people watching her undress” will be represented making use of the template Behave:HumanProperty (Section 3.2.2.1), where MARY_, as a SUBJ, BEHAVE(s) MODAL unwilling_ about the TOPIC of being scrutinized, etc.; “Catia is unable to make use successfully of. . .” will make use again of the template Experience:NegativeHuman/Social, where CATIA_, the SUBJ, EXPERIENCE(s) OBJ failure_ about (TOPIC), etc. For some minor limitations on the query/inference operations imposed by the presence of negv, see Section 4.2.2.

Locations

Location determiners can only be associated, through the operator “colon” “:” (in external format), with the first *four* “main” roles of Table 2.4: SUBJ(ect), OBJ(ect), SOURCE, BEN(e)F(iciary) – see also Table 2.5. The “locations” of the fillers of the residual three roles, MODAL(ity), TOPIC and CONTEXT, can be deduced, in case, from those associated with the “main” four roles.

The *format* of the location determiners follows these two general laws:

- A location determiner is *always represented by a list of elements, concepts or individuals*. If this determiner is associated with an argument of the predicate (i.e. with a filler of a role like SUBJ, OBJ, . . .) consisting of *only one term*, then the elements of the list represent the different locations where this term is *simultaneously situated*, as in the example: “Sharp Corporation produces its personal computers in Taiwan and Korea.” In this case, the two elements (individuals) of the list (TAIWAN_ KOREA_) that make up the location determiner must be interpreted as the locations where the filler of the OBJ role, *personal_computer*, is simultaneously produced. If the argument of the predicate (i.e. the role filler) is, in turn, composed of a list, *a biunivocal correspondence must exist among the terms of the two lists* (“Sharp Corporation makes notebooks in Taiwan and desktops in Korea”).
- Particular rules are followed in the different MOVE constructions; see Section 3.2.2.4. For example, in the MOVE templates that are specializations of Move:MoveAnEntity, the location list linked with the OBJ(ect) filler is always *a two-term list where the first term represents the initial location and the second the final location*. In the example: “Sharp Corporation has shifted production from Japan to Taiwan and Korea,” the location determiner linked with the OBJ filler will be the embedded list (JAPAN_ (TAIWAN_ KOREA_)). If some *stop points* along the way ought to be represented, then it would be necessary to use a list of three terms, where the (list constituting the) *middle term represents the stop(s)*. In the MOVE templates that concern the *generic displacement of a character or a social body* and that derive from Move:AutonomousDisplacement, we systematically represent this situation by indicating that the character or social body, as a SUBJ(ect), *moves himself as an OBJ(ect)*. From a practical

point of view, this means that, in an occurrence concerning a trip of John, JOHN_ will be the filler of both the SUBJ and OBJ slots. In this case, the location determiner (possibly a list) associated with the SUBJ argument represents *the initial location(s) of John*, and the location determiner linked with the OBJ argument *his final location(s)*; see Section 3.2.2.4. If the stop points must be represented, then the OBJ location attribute is a list where *the first term – again a list in case of multiple stop points – corresponds to these points*.

We can note that, in some specific NKRL applications where a very precise characterization of the locations was required, the location determiners of the type “individual” had a *specific internal representation* made up of *two components*:

- A complex, alpha-numerical “zip-code-type” symbol, giving the *best possible approximation of the spatial coordinates of the location to be represented*.
- The *proper individual* (enumerative component), labeled as usual by using the place-name of the location (if this was known); see examples like KOREA_, CHAMPS_ELYSEES, SCHOOL_342, CHURCH_18, 10_DOWNING_STREET, FARM_71, etc.

Note also that, independently from the internal representation, individuals like those above (KOREA_, CHAMPS_ELYSEES, SCHOOL_342, etc.) are instances of the particular HClass concepts denoting the *category* of the given location, e.g. country_, street_, school_, church_, official_building_, farm_, etc.

Temporal Determiners (Attributes)

The problem of finding a complete and computationally efficient system of knowledge representation for temporal data has been intensely discussed in recent years, thanks mainly to the debate aroused by the publication by James Allen, at the beginning of the 1980s, of his proposals of an *Interval Algebra*; see Allen [1981, 1983, 1984]. An “interval” is a finite length of time that starts and ends at definite points; it can be visually represented as a horizontal line with time going from left to right. According to Allen, a “time specialist” dealing automatically with temporal information does not have to consider *either absolute time or the duration of intervals, but merely the relations between intervals*, i.e. it can leave unspecified the exact temporal relationship between intervals. Seven primitive relationships (predicates) between the temporal intervals i_1 and i_2 (and their inverses) are then defined in the Interval Algebra: i_1 before i_2 , i_1 equal i_2 , i_1 meets i_2 (i_1 is before i_2 but there is no interval between them, i.e. i_1 ends when i_2 starts), i_1 overlaps i_2 (i_1 starts before i_2 , and they overlap), i_1 during i_2 , i_1 starts i_2 (i_1 and i_2 share the same beginning, but i_1 ends before i_2), i_1 finishes i_2 (i_1 and i_2 share the same end, but i_1 begins before i_2). A set

of *transitive axioms* [Allen, 1983] defines the behavior of the above predicates; two examples are

$$\text{before}(i_1, i_2) \wedge \text{before}(i_2, i_3) \Rightarrow \text{before}(i_1, i_3) \quad (2.6)$$

$$\begin{aligned} \text{meets}(i_1, i_2) \wedge \text{during}(i_2, i_3) \Rightarrow & (\text{overlaps}(i_1, i_3) \vee \text{during}(i_1, i_3) \\ & \vee \text{meets}(i_1, i_3)) \end{aligned} \quad (2.7)$$

Work that extends Allen’s proposals is described, for example, by Ladkin [1986; Ladkin and Maddux, 1987]; work that investigates the relationships between the “Interval” and “Point” algebras can be found, for example, in Vilain and Kautz [1986] and Tsang [1987]. A temporal logic system introduced independently from the work of Allen, where the concept of “persistence of a situation” is introduced (see also Section 4.1.2) is given by McDermott [1982]. Kowalski and Sergot’s [1986] “Event Calculus” is an attempt to set up a general system for reasoning about time and event in a logic programming framework; in a way, this system can also be considered as an extension of Allen’s proposals. A “spatial” system related to Allen’s interval algebra is the RCC8 Calculus [Randell et al., 1992], which consists of eight topological base relations for extended spatial regions.

In recent years, as a reaction to a pure “interval-based” system in Allen’s style, *we can remark a renewed interest for the “point-based” approaches*: many recent systems described in the literature can handle both *metric (quantitative)* and *qualitative (interval-based)* temporal information. See, in this context, TimeGraph [Miller and Schubert, 1990], MATS (Metric/Allen Time System) [Kautz and Ladkin, 1991], TimeGraph-II [Gerevini and Schubert, 1995], LATER (LAYered TEMPoral Reasoner) [Brusoni et al., 1997], etc.

Three recent projects can be considered as representative of the “state of the art” in the temporal representation domain – at least with respect to the possibility of implementing “non-toy,” concrete applications in this specific domain.

The first is the “Advanced Research and Development Activity (ARDA) Challenge Project on Event Taxonomy” [Bolles and Nevatia, 2004], managed by the ARDA Northwest Regional Research Center and having SRI International (SRI) and the University of Southern California (USC) as principal partners. Centered on the automatic recognition of events in video records, the project has created, among other things, a Video Event Representation Language (VERL) that includes a temporal representation component. *Time can be specified as instants and intervals*. Representation of intervals conforms to Allen’s interval algebra; three relationships, *begins*, *inside*, and *ends*, can hold between an instant t and an interval T ; see, in NKRL, the three temporal modulators defined in Table 2.10. Note also the existence of the three “predicates” *change*, *cause*, and *enable*, where the last two have a *binding function*

similar to that of the “binding occurrences” that, in NKRL, deal with the “connectivity phenomena”; see Section 2.2.3.

The second project, linked with W3C/SW activities, is a “work on progress” about the definition of an *ontology of temporal concepts*, OWL-Time, for describing the temporal content of Web pages and the temporal properties of Web Services; see Hobbs and Pan [2006]. OWL-Time is the successor of the DAML-Time project [Hobbs and Pan, 2004]. The ontology defines two subclasses of TemporalEntity (i.e. Instant and Interval), three relations (i.e. begins, ends and inside) for describing the relationships between instants and intervals (see begins, inside, and ends in VERL and the NKRL’s temporal modulators), a relation (i.e. before) on temporal entities, and a series of “interval relations” (i.e. intervalEquals, intervalMeets, intervalOverlaps, etc. and their reverse relations) to reproduce Allen’s system. All these “relations” are defined as ObjectProperty in OWL and have as range either Instant or Interval. The ontology also includes a set of specialized concepts, predicates and relations to allow the representation of durations (as properties of intervals), dates and other temporal entities: e.g. January is a subclass of DateTimeDescription, with the restrictions that the property unitType takes AllValuesFrom the class UnitMonth and the property month has a value of 1. Clumsy examples of use of these definitions in a Web Services context are described by Hobbs and Pan [2006].

The third project – the best known, and probably also the most interesting from an NKRL point of view, in spite of the big differences both with respect to the practical aims and to the general approach – concerns the Specification Language TimeML; see Pustejovsky et al., [2005]. TimeML has been developed in a *computational linguistics context* with the aim of *annotating*, from a temporal point of view, an NL text by *identifying and extracting events from this text* and by *establishing their temporal anchoring*. To annotate the texts, TimeML makes use of four types of *tags* that make use of a syntax in the XML style (with attributes and values): EVENT, TIMEX3, SIGNAL and LINK.

With respect to the first tag, EVENT, the modalities of identification of events in TimeML do not appear as especially original and seem to be of the “neo-Davidsonian” type (see the discussion in Section 1.1.2.2). They are then linked with the retrieval of tensed verbs, untensed verbs (“...called the first minister *to thank* him...”), nominalizations (“...a possible *attack*...”), adjectives (“...a *dormant* volcano...”), predicative clauses (“...there is no reason why they would not *be prepared*...”) or prepositional phrases (“*on board*”). Attributes for the tag EVENT are of two types, i.e. EventID (the identification of the event, automatically assigned) and Class, with values like OCCURRENCE (die, crash, build, merge, sell), STATE (on board, kidnapped, love), PERCEPTION (see, hear, watch, feel), etc. The annotation of an event is completed making use of a “secondary” tag, MAKEINSTANCE, used to introduce further information about the event, like indications on the tense (PAST, PRESENT, FUTURE, etc.), the aspect (PROGRESSIVE, PERFECTIVE, ...), the modality, etc.

The TIMEX3 tag is employed to mark up *explicit temporal expressions*; its use is modeled on both the use of TIMEX2 – a set of annotations guidelines for creating normalized representations of temporal expressions (“temporal annotations”) in free text originally developed under the DARPA TIDES (Translingual Information, Detection, Extraction and Summarization) program (see Ferro et al. [2005] and <http://timex2.mitre.org>) – and on that of Andrea Setzer’s TIMEX tag [Setzer and Gaizauskas, 2000; Setzer, 2001]. Note the existence of an automatic time tagger, TEMPEX, to generate TIMEX2 tags in text documents automatically [Mani and Wilson, 2000]. The TIMEX3 tag can be associated with several types of *attributes*: some of them are listed below.

- TimexID (automatically assigned).
- *type*, with values like DATE, to annotate *fully specified* time expressions like September 3rd, 2007; TIME, to annotate (as in TIMEX2) *underspecified or context-dependent* expressions like “The hostages were released *that afternoon*”; DURATION, for annotating expressions like “three months” or “two years”; SET, for annotating expressions like “twice a month”, “daily”; etc.
- *value*, where this attribute is used to introduce an instance of the ISO 8601 normalized date format [ISO, 2004], like 2007-10-26T09:15:00 that denotes “October 26, 2007, 09/15.”
- *mod*, used to capture specific “temporal points” like BEFORE, AFTER, ON_OR_BEFORE (“no less than a year ago”), “temporal comparisons” like LESS_THAN, MORE_THAN, EQUAL_OR_THAN, “points and durations” like START, MID (“the middle of the month”) or APPROX (“about three years ago”).
- *temporalFunction*, which introduces a binary value to indicate whether all the temporal information needed is provided (“twelve o’clock September 2, 2007”), or if this is not true (e.g. “yesterday” or “next year”) and it is then necessary to make use of a *temporal function*.
- *FunctionInDocument* (an optional attribute), with values like CREATION_TIME, MODIFICATION_TIME, PUBLICATION_TIME, etc.

For example, the CREATION_TIME of a document can be used in the “temporal functions” mentioned above to identify a “temporal anchor” (AnchorTimeID attribute) from which to start to calculate an actual (ISO) value for “underspecified expressions” like “last week”; the anchor can then be used to locate the week of creation of the document, allowing then to find the week that precedes immediately this “creation week.”

The SIGNAL tag is used to annotate sections of the text, usually function words, which can be used to show *how temporal objects are related together*. They denote then “temporal prepositions and conjunctions” (from, before, after, during, while, when, ...), “temporal modifiers” (twice, every, three times, ...), “subordinators” like “if,” etc.

However, the most important innovation introduced in TimeML (following Setzer’s [2001] thesis) concerns certainly the set of LINK tags (TLINK, ALINK and SLINK) that, as already signaled in Section 1.1.1.2, are particularly interesting because they can be considered as TimeML’s solution to the problems posed by “connectivity phenomena.” TLINK, the “temporal link,” is used to represent *the temporal relationship that holds between events or between events and a specific time*. The `relType` attribute of this link can be used to denote simultaneous events (SIMULTANEOUS, as in the example “Jane was watching TV while Mary was cooking”), before/after events (BEFORE/AFTER), during states or events (DURING, “Mary cooked for 40 minutes on Tuesday”), etc. ALINK, the “aspectual link,” is used to represent *the relationships between an “aspectual event” and its “argument event”*; in this case, the attribute `relType` can denote an event that INITIATES, as in “Mary started (aspectual event) to cook (argument event),” CULMINATES (“Jane finished assembling the table”), TERMINATES or CONTINUES – see, on the contrary, the NKRL use of *temporal modulators* in this context in Table 2.10. Note also that, in TimeML, the “aspectual” predicates like “start” and “finish” in the above examples *are treated as “separate events,”* independent from the specific modified event like “cook.” Eventually, SLINK, the “subordination link,” is used to denote *“contexts” and relationships between two events* – dealing, then, once again, with the omnipresent “connectivity phenomena” – see values for the `relType` attribute like FACTIVE (“Mary managed to leave the party”) COUNTER_FACTIVE (“John forgot to buy some bread”) or MODAL (“Mary wanted John to buy some bread”).

In spite of its unquestionable richness and interest, TimeML sometimes also appears particularly “heavy” and convoluted – we have far from mentioned all the intricacies of its syntax – and not immune from certain dangers of ambiguity and underspecification.

NKRL, Timestamps and Intervals

The association of temporal attributes with temporal modulators is at the heart of the NKRL representation of the specific temporal data to be taken into account in a “narrative information” context; note, however, that a complete apprehension of this representation system cannot be obtained without examining how it is used in a querying and inferencing context; see Section 4.1.2. The two temporal attributes date-1 and date-2 define, in fact, the global time interval or a specific point on the time axis (when associated, in particular, with the modulators begin, end and obs; see Table 2.10) where a predicative occurrence c_k (i.e. the corresponding elementary event) “holds.” Among other things, this implies that, for each predicative occurrence – and in contrast to what happens for the modulators and the location determiners that, with few, very precise exceptions, are in general not mandatory – the presence of the attributes date-1 and date-2 is absolutely mandatory in order

to be able to assign a correct interpretation to this occurrence; e.g. see occurrence mod3.c5 in Table 2.6b.

More precisely, any predicative occurrence c_k is *necessarily associated* with the formal representation of the time interval i in which $\text{holds}(c_k, i)$ is true; i is in turn defined by *the timestamps associated as values* with the two temporal attributes date-1 and date-2. Timestamps are composed of sequences of integers like $\langle \text{year-month-day-hour-minute} \dots \rangle$, where the left and right boundaries of the sequence represent, respectively, the maximum (e.g. years) and minimum (e.g. nanoseconds) *temporal grain* chosen for a given application. In internal representation, each sequence is converted into a *single real*; the equivalence “timestamp \equiv real” preserves the order of the original timestamps on the time axis. From now onward, the term “timestamp” will denote *the real number corresponding to the original sequence*. For clarity’s sake, however, we will continue to use in the following the intuitive “*external*” *NKRL symbolic notation for timestamps*, e.g. in the style of “2006-10-26” – which corresponds, therefore, to the sequence $\langle \text{year-month-day} \rangle$ and which is compatible with the ISO 8601 standard [ISO, 2004]. Pragmatic solutions have been adopted to deal with the problem of “lacunary dates,” given that, in the historical “narrative” applications for example, not all the elements of the original sequence corresponding to a timestamp can be known simultaneously. Assuming, for example, that “maximum grain = years,” *sequences with the year unknown are not permitted*. In this case, a set of possible years must be specified, automatically or by the domain specialist: a copy of the associated occurrence is created for each possible year and the set of occurrences is added to the knowledge base.

Given the above assimilation of timestamps with reals corresponding to points of the time axis, it is now possible to make use of the standard arithmetical properties to establish the relationships between timestamps, and to calculate, when necessary, the duration of the intervals (in the James Allen style). Expressing the generic timestamp as t_k , we can make use, for example, of the simple axioms described in Table 2.14; see also Miller and Schubert [1990].

To calculate concretely the “metric duration,” dt , of an interval bounded by the timestamps t_1 and t_2 , it is necessary to reconvert t_1 and t_2 into the original sequences of integers *to express these two timestamps in terms of “significant multiples” of the minimum grain*. “Significant” means that, e.g., for two timestamps t_1 and t_2 including the years 2006 and 2007 in their

Table 2.14 Relationships among timestamps

$((t_1 \leq t_2) \wedge (t_2 \leq t_3)) \Rightarrow (t_1 \leq t_3)$
$(t_1 \leq t_2) \Leftrightarrow (t_1 < t_2) \vee (t_1 = t_2)$
$(t_1 \leq t_2) \Leftrightarrow (t_2 \geq t_1)$
$(t_1 < t_2) \Rightarrow \neg (t_2 \leq t_1)$
$(t_1 \geq t_2) \Rightarrow \neg (t_2 < t_1)$
$((t_1 \leq t_2) \wedge (t_2 \leq t_1)) \Rightarrow (t_1 = t_2), \text{ etc.}$

original sequences, and assuming that “years” and “days” are, respectively, the maximum and the minimum grain, *only one year period will be added to t_2 when converting the whole sequences into days*. Denoting with $g(t_k)$ the significant multiple of the minimum grain corresponding to the generic time stamp t_k , we will then simply have

$$dt(t_1, t_2) = g(t_2) - g(t_1) \text{ and if } t_1 \leq t_2 \leq t_3, dt(t_1, t_3) = dt(t_1, t_2) + dt(t_2, t_3) \quad (2.8)$$

Categories and Perspectives

If we examine the *qualitative relationships* between the *duration* of an occurrence c_k (the duration of the corresponding event) and the *temporal information* carried by the temporal attributes *date-1* and *date-2*, we can notice several configurations.

In the simplest case, the duration is *fully defined*, as in the representation of the elementary event “Between July 15 and September 5, 2006, John was hospitalized”; in the corresponding predicative occurrence, we do actually have two timestamps t_1 and t_2 , $t_1 \equiv 2006-07-15$ and $t_2 \equiv 2006-09-05$, which allow us to *localize exactly the boundaries of the event*. From the point of view of temporal information, this situation can thus be schematized as in Fig. 2.5: the two timestamps t_1 and t_2 represent the values associated with the temporal attributes *date-1* and *date-2* respectively.

We can now introduce a first fundamental concept of the NKRL system of temporal representation, that of *category of dating*. The first temporal attribute, *date-1*, is said to be represented *in subsequence* – the event begins to be true at the timestamp t_1 (generalized date) that corresponds to the value associated with this attribute – and the second *date-2*, which denotes the upper temporal limit of the event, is said to be represented *in precedence*. The *values* of the two attributes, and the *category of dating* (subsequence or precedence) associated with these attributes, permit us to reconstruct the *temporal interval* fully (in Allen’s meaning) that corresponds to the event we are taking into consideration.

It is often necessary to deal with an event in which *only one* of the two boundaries, t_1 or t_2 , is to be considered – for example, when it is necessary to supply special information about the circumstances at the *beginning* or *end* of the event. Another possibility is that only an *intermediate* timestamp t_3 , between t_1 and t_2 , is known. In all these cases, NKRL requires that we make use only of

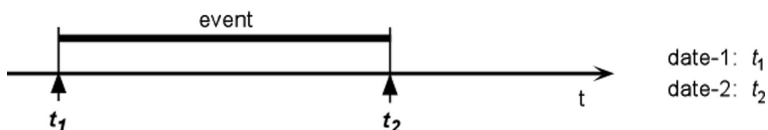


Fig. 2.5 The duration of the event is fully defined

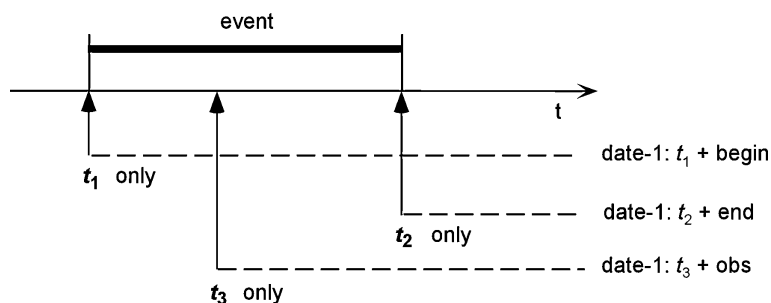


Fig. 2.6 The duration of the event is unknown

the *first* temporal attribute, **date-1**, i.e. *the single timestamp available is systematically associated as value with date-1, the second attribute, date-2, being “empty.”* The three cases are differentiated by using one of the “temporal modulators” of Table 2.10, to be linked to the global coded event. The *beginning* of an elementary event (timestamp t_1) will be represented by making use of the modulator **begin**, and the “filled” temporal attribute **date-1** is then represented *in subsequence*. To indicate the end of an event (timestamp t_2), the modulator **end** is used, and the temporal attribute, **date-1**, must be represented *in precedence*. If, finally, a *particular moment* within an event is to be indicated – for example, to express the information that *it appears* that British Airway’s indebtedness was low on July 2, 1993, or that *it is known* that, in June 1903, Lev Trotsky still agreed with the Mensheviks about the strategy toward socialism, but *without giving any information about the beginning or end of these particular states that extends beyond the given dates* – the modulator “**obs(erve)**” is used. In this last case, the nonempty temporal attribute, **date-1**, is now said to be represented *in coincidence*. The three cases illustrated in this paragraph are summarized in Fig. 2.6.

Eventually, a last case corresponds to the *point events*, i.e. events that have a metric duration dt less than or equal to the minimum temporal grain considered. If, for example, we look at the narrative fragment “Brussels, July 2, 1993. British Airways Plc President Colin Marshall said in a Belgian newspaper interview the company’s indebtedness was low . . .,” it is likely that Colin Marshall’s speech *has occupied only a relatively small slice of July 2, 1993*. This sort of event *appears, then, as concentrated in a particular “point” of the time axis*; see Fig. 2.7. The corresponding occurrences are characterized by the following format: (i) the timestamp t_i representing the date of the point event *is associated as a value with the temporal attribute date-1*; (ii) the temporal attribute **date-2** is *empty*; (iii) *no temporal modulator is to be associated with the predicative occurrence*. The category of **date-1** is the “coincidence.”

Whatever the timestamp to be considered and the associated category, an important source of fuzziness is associated with the accuracy, or rather the *lack of accuracy*, with which this timestamp can be located on the time axis. The

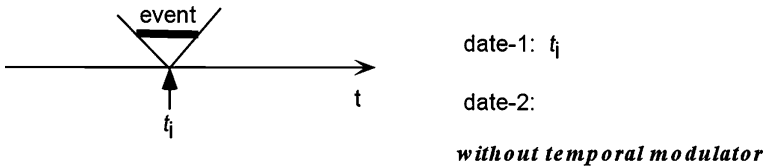


Fig. 2.7 Point event

solutions proposed here (*perspectives*) generalize some remarks that go back to Kahn and Gorry [1977]; see also Miller and Schubert [1990: 110]. Perspectives represent a *simple and elegant way for dealing with some of the “context-dependent expressions”* we have evoked, before, in the context of TimeML (e.g. see certain attributes of the TIMEX3 tag), and compare favorably with the solutions proposed in this project.

In NKRL, “perspectives” correspond to *different ways of “capturing” a timestamp*. We have defined five different perspectives:

- *direct perspective* (no fuzz, as in “July 2, 2006, at noon”);
- *inclusion fork*, or simply *fork* (“between April 7 and September 2, 2006”);
- *limit from which* (“after January 1, 2006”: a way of *indirect dating*, as in the case of a letter A that does not bear a date but mentions having received a letter B, which is dated “January 1, 2006,” this last date is thus a “limit from which” for letter A);
- *limit to which* (the symmetric case, “before December 2, 2005”: letter A does not bear a date, but is mentioned in a letter C, which is dated);
- *circa perspective* (no lower or upper bounds, as in the case of a letter A which does not bear a date, but mentions the celebration of a feast day the date of which is known, without saying if the feast has passed or is to come: letter A is situated around that date, “about December 25, 2005”).

We will stress here that the *type of perspective* that affects the temporal information associated as a value with a temporal attribute is *completely independent* of the *category* according to which the attribute itself is represented. In a piece of information of the type: “John has been hospitalized in 2006, over a period whose first limit is probably between June 10 and June 30, 2006, and the second between September 1 and September 15 of the same year,” the corresponding predicative occurrence would have both the temporal attributes **date-1** and **date-2** “filled,” the first “in subsequence” and the second “in precedence”; but, for both, *the perspectives associated with the timestamps “filling” these temporal attributes would be of the “fork” type. The “category” concerns the temporal attributes; the “perspective,” the values (timestamps) to be associated with such attributes.*

Thus, if we exclude direct dating, these perspectives *all specify a range of possible values for the timestamp to be recorded*. The only element that distinguishes them from each other is the way in which this *range* is indicated in the

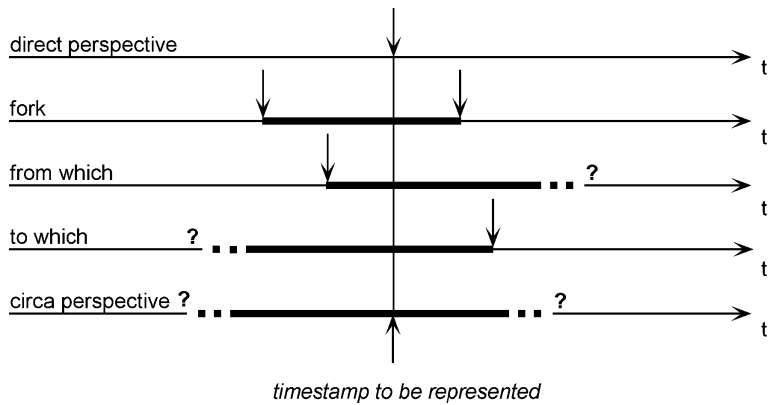


Fig. 2.8 Timestamps and perspectives

original information sources (Fig. 2.8); we see that it can always be reduced to a “fork,” where both limits can be specified, one only (“from which” and “to which”), or none of them (“circa”).

When both the limits are not provided, the missing limit(s) must be restored: a reconstructed date is noted conventionally in round brackets in the external NKRL format. Such a reconstruction could be executed (i) manually by the domain expert at the moment of building up the knowledge base of occurrences, making use of the context of the event to be coded and of its personal knowledge, or (ii) automatically by inference, at the processing time, using some algorithm designed to calculate “so many days” or “so many months,” etc. before and/or after the effective timestamp (date) known and some sort of conceptual representation of the context. See in this context, for example, the “methods” by Kahn and Gorry [1977: 95–98] and, more recently, the “temporal functions” in TimeML mentioned before.

We will conclude this section by noting that the values associated with the temporal attributes `date-1` and `date-2` are represented, in reality, by a vector of two elements (two timestamps). Each time we have to deal with *nondirect perspectives* (fork, from which, to which, circa; see Fig. 2.8 again), the two elements of the vector are both explicitly expressed, giving the limits of a fork inside which is situated the (unknown) “correct” timestamp to be associated with the attribute. On the other hand, the vector expressing a value to be represented in *direct perspective* only contains one explicitly expressed timestamp, e.g. see the value 1999-11-20 associated with `date-1` in occurrence `mod3.c5` in Table 2.6b. The temporal attributes associated with the occurrence translating the previous example, “John has been hospitalized in 2006 . . .,” will then be coded as

date-1: 2006-06-10, 2006-06-30
date-2: 2006-09-01, 2006-09-15

If the information to be represented was, on the contrary, “John has been hospitalized in 2006, over a period whose first limit is probably between June 10 and June 30, 2006, and the second surely before September 15 of the same year,” then the value (category: precedence) to be associated with the attribute `date-2` would be characterized by a “to which” perspective, and the first element of the corresponding vector would then be *reconstructed*; see:

`date-1`: 2006-06-10, 2006-06-30

`date-2`: (2006-09-01), 2006-09-15

Categories and perspectives represent, among other things, *the basic building blocks of the advanced system used for indexing the ORACLE knowledge bases of NKRL occurrences* – this system will be described in detail in Section 4.1.2. This indexing schema allows us to reach a twofold result: (i) when posing a query to the knowledge base, most of the (very complex) operations concerning temporal information *are directly take into account at the index level*; (ii) the real (and, once again, very complex) operations of filtering/unification between the query and the contents of the knowledge base *are executed on a very reduced subset of the knowledge base of predicative occurrences*.

2.3 Second-order Structures

In Chapter 1, we have already discussed the importance, in a general “narrative” context, of being able to deal with those *connectivity phenomena* – expressed, in NL, through syntactic/semantic constructions like causality, goal, indirect speech, coordination and subordination, etc. – that, in a narrative, cause its *global meaning to go beyond the simple addition of the information conveyed by each constitutive elementary event*. These NL connectives constitute, then, the *surface evidence* of those *deep semantic mechanisms* that assure the *logical coherence among the different components of the “stream” identifying a specific narrative*.

For concrete solutions suggested for dealing with the connectivity phenomena from a CS/AI point of view, we can evoke here, first, some “old,” well-known proposals of Schankian inspirations evoking all sorts of scripts, scenarios, thematic abstraction units (TAUs), memory organization packets (MOPs), etc. [e.g. Schank and Abelson, 1977; Schank, 1980; Dyer, 1983; Kolodner, 1984] – the SnePS (Semantic Network Processing System) of Shapiro [1979; Maida and Shapiro, 1982] pertains roughly to the same period and allows us, for example, to represent “narrative” situations like “Sue thinks that Bob believes that a dog is eating a bone.” Among the recent suggestions we can evoke the (already mentioned) CG mechanisms for dealing with “contexts”; see Sowa [1991] and Section 1.2.2.1. In a generic “linguistics/computational linguistics” framework, some solutions have been put forward by DRT [Kamp, 1981; Kamp and Reyle, 1993]; as already stated, DRT is a semantic theory developed for representing and computing trans-sentential anaphora and other forms of text

cohesion. In the same context, see also the TimeML proposals analyzed in the previous section.

In NKRL, the connectivity phenomena are dealt with by making a (limited) use of *second-order structures*: these are obtained from the *reification of predicative occurrences* (of templates in some special cases – see Section 2.3.2.2) *based on the use of their symbolic labels* (see the L_i terms in Eq. (1.2) and, as a concrete example, `mod3.c5` in Table 2.6b). “Reification” is intended here in general (as usual in a object-oriented and AI contexts) as the possibility of *making objects out of already existing complex conceptual patterns* (in our case, templates and predicative occurrences) and to “*say something*” about them *without making reference to the original pattern*.

2.3.1 The Compleitive Construction

A first example of second-order structure is given by the so-called “compleitive construction,” which consists of using as *filler of a role in a predicative occurrence* c_k the *symbolic label (symbolic name) of another occurrence* c_l . *Only the OBJ, MODAL, TOPIC and CONTEXT roles of c_k can accept as filler a symbolic label c_l .*

Note immediately that the symbolic labels c_l used as fillers can denote *not only predicative occurrences*, as in the examples of this section, but also examples of those *binding occurrences* we will introduce in the next section. Additional constraints are:

- *only one* among the OBJ, MODAL, TOPIC and CONTEXT roles associated in the case with an occurrence c_k can be filled with a symbolic label c_l ;
- c_l must correspond *strictly* to a *simple filler* (it must be, then, a *single* symbolic label), i.e. complex fillers (expansions) cannot be used in the framework of a compleitive construction reference.

For implementation reasons this unique label is prefixed, *in external format*, by a “sharp,” “#,” code. The general format of a “*compleitive construction filler*” actually corresponds, then, to `#symbolic_label`; see the tables below. Note that `symbolic_label` is a *regular concept of HClass* (see Section 3.1.2) that has as instances all the *concrete labels* used to denote *both predicative or binding occurrences* in the different NKRL applications.

As an example of compleitive construction, Table 2.15 gives the NKRL representation of a fragment of Reuters’ news already mentioned in previous sections, like “Brussels, July 2, 1993. British Airways Plc President Colin Marshall said in a Belgian newspaper interview the company’s indebtedness was low...” For simplicity’s sake, in this example, as in the majority of the other examples of the book, the frame structures corresponding to the individuals – Colin Marshall, Brussels, etc. in this case – are not reproduced in the table.

Table 2.15 A complete construction of the “transmission of information” type

conc5.c1)	MOVE	SUBJ	(SPECIF COLIN_MARSHALL (SPECIF chairman_ BRITISH_AIRWAYS)): (BRUSSELS_)
		OBJ	#conc5.c3
		BENF	(SPECIF newspaper_ BELGIUM_)
		MODAL	interview_
		date-1:	1993-07-02
		date-2:	
		Move:StructuredInformation (4.42)	
conc5.c3)	EXPERIENCE	SUBJ	BRITISH_AIRWAYS
		OBJ	(SPECIF indebtedness_ (SPECIF amount_ small_))
		{obs}	
		date-1:	1993-07-02
		date-2:	
Experience:NegativeHuman/Social (3.222)			

In Table 2.15, the SPECIF sub-list (SPECIF amount_ small_) in the occurrence conc5.c3 is one of the special SPECIF sub-lists *labeled in a normalized way* (“NKRL idioms”) that will be examined in detail in Section 3.1.2.1; see also this last section for a discussion about an HClass `quantifying_property` like `small_`. As already stated (see Table 2.10), the presence of the temporal modulator `obs(erve)` leads to an interpretation of the occurrence conc5.c3 as the description of a situation that, at this particular date, *is observed to exist*.

The *completive link* is realized here by introducing the symbolic label, conc5.c3, of a “subordinate clause” (bearing the informational content to be spread out, the decrease of British Airways’ indebtedness in our example) as the OBJ(ect) filler (see OBJ #conc5.c3 in occurrence conc.c1) of a MOVE predicative occurrence. In NKRL, this particular type of completive construction is largely used given that it *is the mandatory way of translating any sort of “transmission of information”*; see the discussion about the `Move:TransmitInformation` template and its specializations in Section 3.2.2.4. However, *the use of the completive construction is not limited to the representation of information transmission phenomena*; as already stated, fillers in the form of #symbolic_label can also be used in association with the MODAL, TOPIC and CONTEXT roles.

As a further example of completive construction, let us consider the code represented in Table 2.16, which translates this sort of complex situation (the example concerns a corporate application of NKRL for managing complex dialogue structures in the “Beauty Care” domain):

- *Occurrence skin7.c1*. On January 1st, 2002, Sarah121 sends a message to the Beauty Net Community; the content of this message is detailed in occurrence skin7.c2.

Table 2.16 Further examples of completive constructions

skin7.c1)	MOVE	SUBJ	SARAH_121
		OBJ	#skin7.c2
		BENF	BEAUTY_NET_COMMUNITY
		date-1:	2002-01-01
		date-2:	
Move:StructuredInformation (4.42)			
skin7.c2)	RECEIVE	SUBJ	SARAH_121
		OBJ	advice_
		SOURCE	BEAUTY_NET_COMMUNITY
		TOPIC	#skin7.c3
		CONTEXT	(SPECIF eyeshadowing_ SARAH_121)
		{wish}	
		date-1:	2002-01-01
		date-2:	
Receive:DesiredAdvice (7.21)			
skin7.c3)	OWN	SUBJ	(SPECIF use_ eyeshadow_powder)
		OBJ	property_
		TOPIC	(SPECIF more_effective_than
			(SPECIF use_ eyeshadow_cream))
		CONTEXT	eyeshadowing_
		{poss}	
		date-1:	2002-01-01
		date-2:	
Own:CompoundProperty (5.12)			

- *Occurrence skin7.c2*. In this message, she says she would like (modulator wish) to obtain their opinion, in the context of some eye shadowing operations, about (TOPIC role) what is related in occurrence skin7.c3.
- *Occurrence skin7.c3*. In particular, she would like to know whether, in the context of these eye shadowing operations, the utilization of an eye shadow powder is more effective than that of an eye powder cream.

In these predicative occurrences, the first one, skin7.c1, is an instance of the template Move:StructuredInformation, skin7.c2 is an instance of Receive:Desired Advice and skin7.c3 is an instance of the specific Own:CompoundProperty template; the Own:Property templates are largely used in NKRL to describe the properties of both entities and processes (see Section 3.2.5.5). *advice_* is a specific term of *information_content* in HClass via its subsuming terms *generic_word_content* and *word_content*; *more_effective_than* is a *relational_property*, i.e. a *non_sortal_concept*; see Section 3.1.2.1. *eyeshadowing_* is a specific term of *beauty_care_process* (a specialization of *process_*) via its subsuming term *eye_related_care_process*; *eyeshadow_powder* and *eyeshadow_cream* are both specific terms of *artefact_* through their subsuming terms *eye_related_product/tool* and *beauty_care_product* terms, etc. – these last terms pertain to the specific branches of HClass added to deal with the particular terminology of the Beauty Care domain.

Table 2.17 The Move:StructuredInformation template

name: Move:StructuredInformation

father: Move:TransmitInformation

position: 4.42

NL description: “Transmit a Structured Information”

MOVE	SUBJ	<i>var1</i> : [(<i>var2</i>)]
	OBJ	<i>var3</i>
	[SOURCE	<i>var4</i> : [(<i>var5</i>)]]
	[BENF	<i>var6</i> : [(<i>var7</i>)]]
	[MODAL	<i>var8</i>]
	[TOPIC	<i>var9</i>]
	[CONTEXT	<i>var10</i>]
	{[modulators], \neq abs}	
<i>var1</i> =	human_being_or_social_body	
<i>var3</i> =	symbolic_label	
<i>var4</i> =	human_being_or_social_body	
<i>var6</i> =	human_being_or_social_body	
<i>var8</i> =	electronic/media_product, information_support, service_, services_agency, transmission_medium, temporal_attribute	
<i>var9</i> =	sortal_concept	
<i>var10</i> =	situation_, symbolic_label	
<i>var2</i> , <i>var5</i> , <i>var7</i> =	location_	

To give an example of how the *mandatory use* of completive constructions for predicative occurrences is denoted in a “template” context, Table 2.17 reproduces the template Move:StructuredInformation – which is at the origin of the occurrences conc5.c1 of Table 2.15 and skin7.c1 of Table 2.16 (see also Section 3.2.2.4). The constraint on the OBJ filler shows clearly that this role *must* be necessarily filled by an instance of the symbolic_label concept.

For clarity’s sake, we will now reproduce below *the totality of the syntactic rules* that govern the set up of *well-formed completive expressions*:

- The creation of an *associative relationship* of the “completive construction” type consists in introducing the symbolic label c_l of an NKRL occurrence, *predicative or binding* (see next section), as “filler” of a role of a *predicative occurrence* c_k .
- *Only the roles OBJ, MODAL, TOPIC and CONTEXT* of c_k can be “filled” with the symbolic label c_l .
- In c_k , *only one of the above four roles can be filled with c_l* , i.e. it is forbidden to make use of more than a *single* completive construction reference within the same predicative occurrence.
- The filler c_l must be a “simple” filler, i.e. expansions (structured argument) implying the use of the AECS operators *cannot be used* to set up relationships among occurrences in the completive construction style.
- In the predicative occurrence c_k , the actual format of the filler c_l is #symbolic_label, where symbolic_label is a regular HClass concept.

- In the templates, the mandatory use of a completive construction filler in the derived predicative occurrences is denoted through the use of a symbolic_label constraint on one of the variables var_i of the template; see Table 2.17.

2.3.2 Binding Occurrences

A second, *more general* way of linking together NKRL structures to take into account the “connectivity phenomena” consists of making use of “*binding occurrences*,” i.e. *binary structures under the form of lists labeled with specific “binding operators,” whose arguments are represented (reification) by symbolic labels of (predicative or binding) occurrences*. Extensions to binding structures having “partially instantiated templates” as arguments will be introduced in Section 2.3.2.2.

Unlike templates and predicative occurrences, binding occurrences are then characterized by *the absence of any predicate or role*: they present *strong syntactic similarities* with the “expansions” (structured arguments) introduced in Section 2.2.2.3, the most important difference being that “*binding occurrences*” are *full-fledged*, “*independent*” second-order objects endowed with a proper symbolic “*name*.”

2.3.2.1 The Binding Operators

Like the expansions, the binding occurrence lists are then characterized by the presence of an *operator* (a “*binding operator*” here) as the *first element* of the list. The binding operators are listed in Table 2.18; see also the expansion operators of Table 2.8. To enforce the syntactic coherence of the global NKRL code, the binding occurrences *must necessarily conform* to the following *mandatory* restrictions to be considered as *well formed*:

- The terms (arguments) c_i that, in a generic binding list, are associated with one of the operators of Table 2.18, *denote necessarily (single) symbolic labels of (predicative or binding) occurrences*. Therefore (in contrast to what happens when dealing with the expansion structures), these *arguments cannot denote (canned) lists labeled in turn with binding operators*. Note also that each element c_i of the binding list refers to a *different* binding or predicative occurrence.
- In the binding occurrence of the ALTERN, COORD and ENUM type, *no restriction is imposed on the cardinality of the list*, i.e. on the possible number of terms (arguments) c_i .
- In the binding occurrences labeled with CAUSE, REFER, GOAL, MOTIV and COND, in contrast, *only two arguments, c_k and c_l , are admitted*. The binding occurrence labeled with the above five binding operators are then simply of the type: (OPERATOR $c_k c_l$). In these lists, the arguments (symbolic

Table 2.18 Binding operators of NKRL

Operator	Acronym	Mnemonic description
<i>Alternative</i>	ALTERN	The <i>disjunctive</i> operator. <i>Only a term</i> of the associated list of labels of predicative/binding occurrences must be considered, but this term is not known a priori
<i>Coordination</i>	COORD	The <i>collective</i> operator. <i>All the terms</i> of the list must, obligatorily, be considered <i>together</i> to give rise to a valid binding relationship
<i>Enumeration</i>	ENUM	The <i>distributive</i> operator. <i>Each term</i> of the list must be considered to produce a valid binding relationship, but they satisfy this relationship <i>separately</i>
<i>Cause</i>	CAUSE	The <i>strict causality</i> operator, introducing a <i>necessary and sufficient causal relationship between the first and the second arguments of the list</i> , the latter explaining the former. Only two terms can appear in a CAUSE binding occurrence; see also Section 2.3.2.3
<i>Reference</i>	REFER	The <i>weak causality</i> operator, introducing a <i>necessary but not sufficient</i> causal relationship between the first and the second arguments of the list
<i>Goal</i>	GOAL	The <i>strict intentionality</i> operator; the first argument is <i>necessary</i> to realize the second, and the second is <i>sufficient</i> to explain the first. The predicative occurrence(s) corresponding to the second argument is/are <i>necessarily marked</i> as “uncertain,” operator “*” (see Section 2.2.2.4)
<i>Motivation</i>	MOTIV	The <i>weak intentionality</i> operator; the first argument is <i>not necessary</i> to realize the second, but the second is <i>sufficient</i> to explain the first. The predicative occurrence(s) corresponding to the second argument is/are <i>necessarily marked</i> as “uncertain,” operator “*”
<i>Condition</i>	COND	The <i>predicative occurrence</i> corresponding to the first argument represents an event that <i>could happen</i> if the <i>predicative/binding occurrence</i> (i.e. event or complex situation) corresponding to the second argument <i>could be realized</i> . The first argument (predicative occurrence) is <i>necessarily associated</i> with a modulator <i>poss</i> (see Table 2.13); the predicative occurrence(s) corresponding to the second argument is/are <i>necessarily marked</i> as “uncertain,” operator “*”

labels) can denote, in general, both a predicative and a binding occurrence; an exception is represented by the COND binding occurrences, where the first argument c_k *must correspond necessarily to a predicative occurrence*; see Table 2.18.

A first example will show, among other things, how “completive constructions” and “binding occurrences” can coexist within a fragment of *conceptual annotation* (or *metadocument*); a conceptual annotation (see Zarri [2003b] and the examples of Chapter 3) is a *structured association of binding and predicative occurrences* intended to supply a *detailed representation of the “meaning” of a*

Table 2.19 Mixing complete constructions and binding occurrences

conc5.c1)	MOVE	SUBJ	(SPECIF COLIN_MARSHALL (SPECIF chairman_ BRITISH_AIRWAYS)): (BRUSSELS_)
		OBJ	#conc5.c2
		BENF	(SPECIF newspaper_ BELGIUM_)
		MODAL	interview_
		date-1:	1993-07-02
		date-2:	
Move:StructuredInformation (4.42)			
conc5.c2) (CAUSE conc5.c3 conc5.c4)			
conc5.c3)	EXPERIENCE	SUBJ	BRITISH_AIRWAYS
		OBJ	(SPECIF indebtedness_ (SPECIF amount_ small_))
		{obs}	
		date-1:	1993-07-02
		date-2:	
Experience:NegativeHuman/Social (3.222)			
conc5.c4)	PRODUCE	SUBJ	BRITISH_AIRWAYS
		OBJ	(SPECIF capital_increase BRITISH_AIRWAYS)
		date-1:	(1993-01-01), 1993-07-02
		date-2:	
Produce:EconomicInterestActivity (6.48)			

complex narrative document. Table 2.19 provides, then, the *complete NKRL image* of the example already used in Table 2.15: “Brussels, July 2, 1993. British Airways Plc President Colin Marshall said in a Belgian newspaper interview the company’s indebtedness was low *following a capital increase.*”

In Table 2.19, *the content of the message is now represented by the binding occurrence conc5.c2.* This occurrence – which is labeled using the CAUSE operator defined in Table 2.18 (see also Section 2.3.2.3) and that follows the “only two arguments” rule – means that event conc5.c3, the main event, *has been caused by* event conc5.c4. The new predicative occurrence conc5.c4 translates an elementary event that has been interpreted as a *point event* (no temporal modulator) from a *category* point of view, and as a *limit to which* (“before July 2, 1993,” see the date fork) from a *perspective* point of view; see the “Categories and perspectives” subsection in Section 2.2.2.4. The first term of the fork, January 1st, 1993, is a *reconstructed date*, noted in parentheses in external format.

2.3.2.2 Priority Rule and Generalization to Templates

We will now introduce two important, general remarks about the binding operators introduced in the previous section.

Correspondence between “Expansion” and “Binding” Operators

A first remark concerns the correspondence between the operators of Tables 2.8 and 2.18. The basic semantics of ALTERN, COORD and ENUM are, obviously, *the same* whether they are used within expansion lists or binding occurrence lists. This is why, to avoid any possible confusion, these three operators are denoted in the “external” NKRL notation as ALTERN1, COORD1 and ENUM1 *when they are used as expansion operators*.

Moreover, the CAUSE, REFER, GOAL, MOTIV and COND operators of Table 2.8 correspond, in a sense, to the operator SPECIF(ication) of Table 2.8. This last operator is used, in fact, to *supply further information* about the *first argument* of a SPECIF list – in a list (SPECIF e_i a b c ...) in fact, the properties a , b , c , ..., concepts or individuals, introduce additional details about the item e_i (the first argument) that follows immediately the operator SPECIF. When dealing with binding lists labeled as CAUSE, GOAL, etc., we can consider the (single; see Table 2.18) predicative or binding occurrence c_2 that represents, for example, the second argument of a list (CAUSE c_1 c_2) as an element that, once again, *supplies further information* (the cause in this case, but also the goal, the possible condition for the realization ...) about what is stated in the first argument (occurrence c_1) of the list.

However, for the binding occurrences, no strict formal constraints in the style of the “priority rule” introduced in Section 2.2.2.3 can be imposed. This depends mainly on the possibility that binding occurrences labeled with COORD can be found, within a structured list of binding and predicative occurrences, in a position that contradicts the strict sequence of the operators imposed by the priority rule. In fact:

- In a binding list of the CAUSE, REFER, GOAL, MOTIV and COND type, one of the two symbolic labels (normally the second, c_2) *can correspond to another binding occurrence, in particular to a binding occurrence of the COORD type*. For example, in Table 2.19, the second argument conc5.c4 (the CAUSE) in the binding occurrence conc5.c2 could, in fact, *instead of a single predicative occurrence* (a single elementary event), *denote a long narrative development represented by several occurrences grouped in a COORD list*. This last situation would then be equivalent to the introduction of a COORD list within a CAUSE, REFER, GOAL, etc. binding list, i.e. *within a list that, according to what is expounded above, is supposed to correspond to a SPECIF list*. The priority rule as expressed in Section 2.2.2.3 would then be *implicitly violated*.
- This situation is quite frequent in practice – as evidenced also by the examples of Chapter 3 – because of the decision of admitting a *liberal use* of COORD binding lists within long sequences of predicative occurrences in order to *group together* (“factorize”), for both readability and logical coherence’s sake, occurrences that *are very close from a semantic content point of view*. For example, we could group together in a COORD binding occurrence (i) a PRODUCE predicative occurrence relating the creation of some

artifacts and (ii) several occurrences in the style of `skin7.c3` in Table 2.16 (derived then from `Own:Property` templates) *to specify the different properties of these artifacts*. The resulting COORD binding occurrence could, after that, correspond to the *second argument* of a CAUSE, GOAL, etc. list.

Concretely, the only restriction on the use of binding operators within binding occurrences (introducing, then, a sort of weakened form of the priority rule) concerns the impossibility of making use of arguments c_i denoting binding occurrences of the ALTERN and ENUM types within binding occurrences of the COORD type. Within these COORD binding occurrences, however, c_i labels can be freely used to denote binding occurrences of the COORD, CAUSE, REFER, GOAL, MOTIV and COND types – as already stated, the last five operators correspond (very roughly, and only from an operational point of view) to the SPECIF operator of the AECS sub-language.

Using Template Labels within Binding Structures

A second remark concerns the *generalization of the binding structures* that can be obtained by using as arguments the symbolic labels of “partially instantiated templates” t_i , instead of symbolic labels of predicative/binding occurrences. These *binding structures* are sometimes called *binding templates*. This possibility is particularly useful when *knowledge contents of a high level of abstraction and generality must be represented*; see the definitions associated with the “procedure type properties” in an HClass context, Section 2.2.1.3, and the “NKRL inference rules” in Chapter 4. We recall here that a “partially instantiated template” corresponds to a standard template (descriptive component) where *at least some of the explicit variables (var_i) originally associated with this template have been replaced by some of their constraints* (HClass terms) – or by specializations/instances (individuals) of these constraints.

As an example, we reproduce in Table 2.20 the “predicative” and “binding” templates included in the representation of a fragment of “normative” text, the beginning of article no. 57 of the French General Taxation Law – e.g. see Zarri [1992b] for a more complete analysis of this text. Article no. 57 is the main normative source used to settle cases concerning an “indirect transfer of revenues abroad”; as already stated in Section 2.2.1.3, the NKRL code of Table 2.20 is then part of the definition of the HClass concept `norms_for_indirect_transfer_of_revenues_abroad`. The beginning of this article reads as follows, according to a rough English translation: “*To determine the income tax payable by companies in France which are under the authority of, or which exercise a control over, companies domiciled abroad, the revenues indirectly transferred abroad must be added to the results registered in the books.*” For simplicity’s sake, the code reproduced in Table 2.20 concerns only the fragment in italics of the above definition.

Objects like `ind_trans.t4` and `ind_trans.t5` are then “*binding*” templates, i.e. binding structures where the arguments of the operators are the symbolic labels

Table 2.20 “Binding” and “predicative” templates

ind_trans.t1)	(GOAL ind_trans.t2 ind_trans.t3)		
ind_trans.t3)	(ALTERN ind_trans.t4 ind_trans.t5)		
ind_trans.t4)	(COORD ind_trans.t6 ind_trans.t7)		
ind_trans.t6)	PRODUCE	SUBJ	var1
		OBJ	(SPECIF calculation_income_tax)
		BENF	var2: (FRANCE_)
var1 = human_being_or_social_body			
var2 = company_			
Produce:Numerical/StatisticalProcess (6.44)			
ind_trans.t7)	OWN	SUBJ	var2: (FRANCE_)
		OBJ	control_
		TOPIC	var3: (var4)
var2 = company_			
var3 = company_			
var2 ≠ var3			
var4 ≠ FRANCE_			
Own:ControlOfCompany (5.221)			
ind_trans.t5)	(COORD ind_trans.t8 ind_trans.t9)		
ind_trans.t8)	PRODUCE	SUBJ	var1
		OBJ	(SPECIF calculation_income_tax)
		BENF	var2: (FRANCE_)
var1 = human_being_or_social_body			
var2 = company_			
Produce:Numerical/StatisticalProcess (6.44)			
ind_trans.t9)	OWN	SUBJ	var3: (var4)
		OBJ	control_
		TOPIC	var2: (FRANCE_)
var3 = company_			
var2 ≠ var3			
var4 = country_			
var4 ≠ FRANCE_			
Own:ControlOfCompany (5.221)			

t_i of *partially instantiated templates*. ind_trans.t1 says that the operations *consisting of the addition of the revenues transferred abroad to the results, etc.* (see the wording of the article) – these operations (denoted by ind_trans.t2) *are not explicitly represented* in the code of Table 2.20 – are *strictly necessary* (GOAL) to allow the realization of the main task described in article no. 57 (the calculation of the income tax, ind_trans.t3); moreover, they also *precede* the execution of this task (see also the next section). ind_trans.t3 represents the *alternative* that is expressed by the textual fragment in italics. According to ind_trans.t4, the calculation of the income tax (ind_trans.t6) may concern a French company, var2, that controls (ind_trans.t7) a generic foreign company, var3;

according to `ind_trans.t5`, the same calculation (`ind_trans.t8`) may concern a French company that is controlled by a foreign company (`ind_trans.t9`). Note that all the binding structures conform to the *syntactic restrictions* defined above.

In Table 2.20, `ind_trans.t6` and `ind_trans.t8` are *partial instantiations* (see the utilization of the individual `FRANCE_` and of `calculations_`, a specialization of the concept `numerical/statistical_process`) of the template `Produce:Numerical/StatisticalProcess`. `ind_trans.t7` and `ind_trans.t9` are partial instantiations of `Own:ControlOfCompany`; see the addition of specific locations, etc.

2.3.2.3 Binding Operators and Temporal Representation

If we consider the relationships between *temporal information* and *binding occurrences*, a first remark concerns the fact that the temporal attributes `date-1` and `date-2` necessarily associated with the predicative occurrences mentioned in the binding occurrences are *normally filled with explicit temporal indications* (explicit timestamps). Therefore, their *temporal arrangement* within the binding occurrence is not difficult to assess.

Moreover, the four binding operators that constitute together the NKRL *taxonomy of causality*, CAUSE, GOAL, REFER and MOTIV, are associated with *explicit temporal valences* as illustrated by the examples of Fig. 2.9. As we can see, CAUSE and GOAL respectively express a sort of *strict* causality or

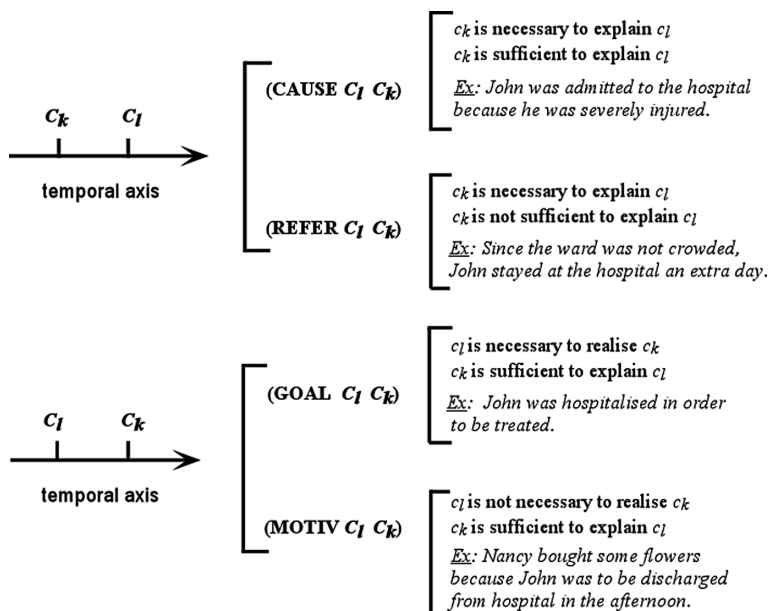


Fig. 2.9 Time and causality in NKRL

purpose; REFER and MOTIV express a *weak* causality and purpose respectively. Expressed in different terms, we can say that the use of CAUSE and GOAL is only permitted in the presence of a “material implication” $c_k \supset c_l$, where c_k and c_l denote two generic predicative occurrences (elementary events). It is evident that this last condition is not respected in the two examples of Fig. 2.9 that illustrate the use of REFER and MOTIV.

We can conclude this section with two simple remarks:

- The first is that the taxonomy outlined above looks, of course, *quite simplistic* compared with all the possible “classical” theories about causality, going back to Aristotle’s work (who considers four types of cause – material causes, formal causes, efficient causes, and final causes) and continuing with Hume, Kant, Gibbon, etc. Unfortunately, many of these theories are *too complex or not specific enough* to be used within a *practical and robust computational framework*; see, in this context, the recent “Force Dynamics” causal theory (of cognitive linguistics origin) proposed by Talmy [1988, 2000] and characterized, among other things, by a sophisticated dichotomy between prototypical and nonprototypical specializations of the “causing” and “letting” categories. “Practical” proposals nearer in their basic motivation to NKRL’s approach can be found, for example, in Schank and Abelson [1977: 30–32] and Lenat and Guha [1990: 240–243]; see also NKRL’s treatment of “wish, will,” etc. in Section 3.2.2.1.
- However, even the “causal taxonomy” proposed in this section has sometimes been considered as *too complex* to be used in practical applications, mainly (i) when “narrative” information to deal with is particularly abundant and (ii) when some form of automatic generation of the NKRL code from textual input must be considered. For example, in recent applications of the NKRL technology like the CONCERTO, EUFORBIA or PARMENIDES projects, *only the two “causal” operators CAUSE and GOAL have been really utilized*, reducing then the use of REFER to that of CAUSE and the use of MOTIV to that of GOAL, without producing, apparently, *a too important loss of information*.

2.4 In the Guise of Winding Up

In this chapter, we have described the *general architecture* of the NKRL language and the *syntactic rules* that govern the production of well-formed NKRL expressions – Chapter 3 will deal with the specific NKRL *semantic and ontological contents*. Should we want to summarize in a few sentences what was expounded in the sections above, we could say that:

- NKRL is structured into *four connected “components,”* where each of them takes into account *a particular category of narrative phenomena*. The *definitional component* concerns the formal definition of the *concepts* C_i

(according to the usual “ontological” meaning of this term), and the *enumerative component* concerns the definition of the *instances* l_i of these concepts (*individuals*). The *descriptive component* deals with the *formal representations* (*templates*, t_i) of *general classes of elementary events* like “moving a generic object,” “formulate a need,” “be present somewhere,” and the *factual component* deals with the representation of the *specific events that correspond to particular instantiations of these general classes* (predicative occurrences, c_i). With respect to description logics terminology, the definitional component corresponds roughly to Tbox and the enumerative component to Abox. *No description logics structures correspond to the descriptive and factual structures of NKRL.*

- Concepts and individuals are inserted into an “ontology of concepts,” a directed acyclic graph that, in NKRL, takes the name of HClass (hierarchy of classes). Their “binary” data structures are *relatively standard* and consist basically of *frame-like structures*, i.e. of bundles of *property/value* relationships where neither the number nor the order of the properties is fixed. The properties (i.e. the slots) of the frames describing specific concepts/individuals are obtained by specializing a set of *prototype slots*. From a semantic point of view, the properties are grouped in three categories: *relations*, *attributes*, and *procedures*. The latter are of particular interest, given that their (complex) values are described making use of the “*n*-ary” data structures proper to the descriptive and factual components. In NKRL, *concepts cannot be considered as instances of other concepts*; individuals are added, when necessary, as “leaves” of HClass by using *explicit local operations*.
- The *n*-ary data structures, templates and predicative occurrences, used (i) for the description of the general classes of events (*templates*) and (ii) for the description of their instances, i.e. the concrete events (*predicative occurrences*), are formed by bundles of “quadruples” connecting together the *symbolic name* of the template/occurrence, a *predicate* and the *arguments* of the predicate introduced by *named relations*, the *roles*. The quadruples have in common the “name” and “predicate” components. Predicates and roles are *primitive*; the arguments can be simple HClass elements, concepts or individuals, or well-formed structures (*structured arguments or expansions*), where *lists of HClass elements*, labeled using the four operators of the AECS sub-language, can be *interwoven* according to a *priority rule*.
- To assign a *valid semantic interpretation* (as an event or a class of events) to an NKRL descriptive/factual structure (template or predicative occurrence), the latter *must* at least consist of a predicate, a role and an argument. However, *determiners* (attributes) that (i) *introduce further details/precisions* concerning the “meaning” of these templates/occurrences but that (ii) are *never strictly necessary* for their basic semantic interpretation in NKRL terms can also be added. Determiners include *modulators* (*temporal, deontic and modal modulators*), *locations* and *temporal determiners*: the presence of two *temporal determiners*, *date-1* and *date-2*, is mandatory in order that a *predicative occurrence* can be considered as *well formed*. Original concepts of

the NKRL representation system for temporal data are the “*category of dating*,” which allows defining the relationships (*precedence*, *subsequence*, *contemporaneity*) between timestamps and events, and the “*perspective*,” which defines the *different ways (different degrees of precisions) of “capturing” a timestamp*.

- *Second-order structures* obtained through the *reification* of templates and predicative occurrences are used to deal with those *connectivity phenomena* (in NL terms: causality, goal, indirect speech, coordination and subordination, etc.) that, in a narrative represented by a stream of elementary events, cause the *global meaning of the narrative to go beyond the simple addition of the information conveyed by each elementary event*. A first type of second-order structure is the *completive construction*, where the symbolic label of a predicative/binding occurrence can be used directly (reification) as filler of a role in (different) predicative occurrences. A second, *more general way* of linking together NKRL occurrences consists of making use of “binding occurrences,” i.e. *binary structures under the form of lists labeled with specific “binding operators”*. The arguments of the binding operators are represented (reification) by symbolic labels of NKRL occurrences – and, more in general, of NKRL templates. Four of these “binding operators” (CAUSE, GOAL, REFER and MOTIV) form the NKRL “taxonomy of causality.”

To conclude about the four NKRL components, we present in Fig. 2.10 a graphical representation of their *main relationships*. We make use for this of a very simple narrative example where, for intelligibility’s sake, the temporal information has been suppressed: “Berlex Laboratories have performed an evaluation of a given compound.”

Many additional simplifications have been introduced in this figure. For example, the organization of the HTemp and HClass hierarchies is extremely simplified with respect to their real structure (see Chapter 3); the template “evaluate an artifact” that appears in the “descriptive component” layer as an offspring of the Produce:PerformTask/Activity template is an approximation of the “real” template Produce:Assessment/Trial; the “individual” (enumerative component) COMPOUND_27, which represents the specific compound mentioned in the event, is supposed here to be a direct instance of the (very high level) concept *artefact_*, etc. Note that, for uniformity’s sake, we have considered that the proper assessment activities are well specified and that, therefore, they can be described using a particular *individual*, ASSESSMENT_1. The concept *assessment_* is a specific term of *activity_*, and *company_* is a specific term of *social_body*.

The arrowhead lines represent, in general, *have an instance* links. This means, for example, that the predicative occurrence “Berlex Laboratories have performed an evaluation...” of the *factual component* has been created from the corresponding “evaluate an artifact” template of the *descriptive component*, which is, in turn, a *specialization* of the generic template, Produce:PerformTask/Activity, used to describe the fact/situation/event of “performing a task

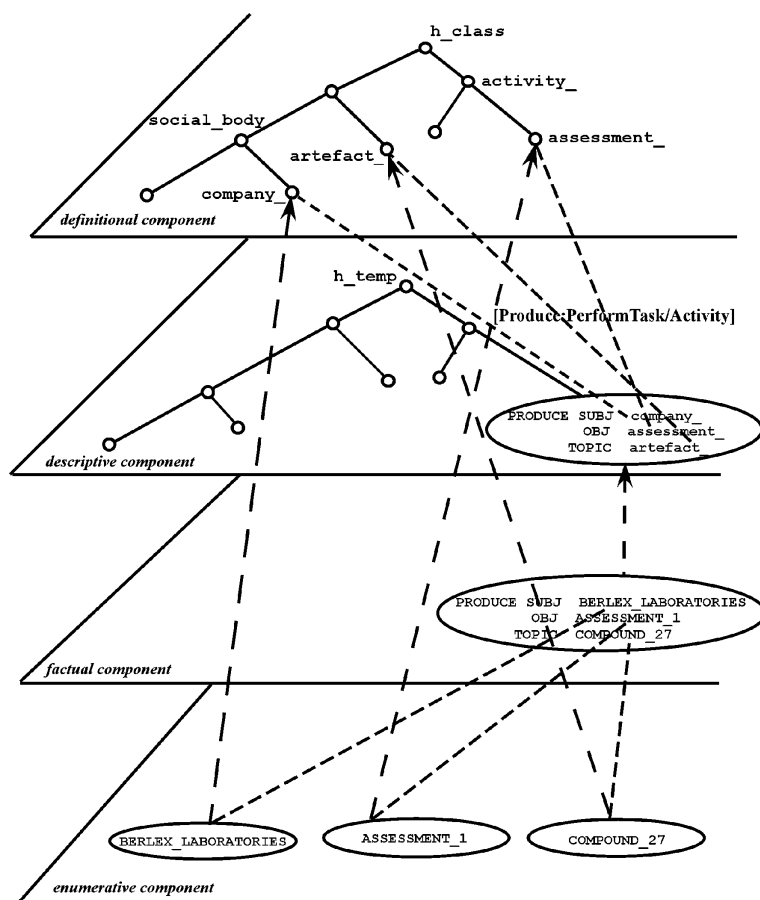


Fig. 2.10 Relationships among the NKRL components

or an activity.” More precisely, the predicative occurrence has been obtained by replacing – for each role SUBJ(ect), OBJ(ect), TOPIC (à propos of...) of the original “evaluation” template – the concepts indicating the general classes of legal fillers (constraints) for these roles, e.g. `company_`, with (in this case) particular individuals, e.g. `BERLEX LABORATORIES`.

The dotted lines without any arrow represent a sort of “coreference link.” This means that, for example, in the HTemp “evaluation” template (descriptive component), `company_` is a symbolic label allowing us to access, from the descriptive component environment, information stored in the frame-like structure `company_` that pertains to the definitional component (HClass). Similarly, the label `COMPOUND_27` in the “evaluation” occurrence (factual component) refers to the corresponding frame-like object pertaining to the enumerative component environment.

Representation and Management of Narrative
Information

Theoretical Principles and Implementation

Zarri, G.P.

2009, X, 302 p. 55 illus., Hardcover

ISBN: 978-1-84800-077-3