International Journal of Robotic Computing
Vol. 1, No. 1 (2019) 1-22
© KS Press, Institute for Semantic Computing Foundation
DOI: 10.35708/RC1868-126248

Advanced Knowledge Representation and Reasoning Making Use of an Advanced N-ary Model

Gian Piero Zarri

STIH Laboratory, University Paris-Sorbonne, 1 rue Victor Cousin, 75005 Paris, France
`zarri@noos.fr, gianpzarri@gmail.com`

Received (11/02/2018)
Revised (06/13/2019)
Accepted (07/15/2019)

Abstract. We discuss in this paper some aspects of NKRL, the Narrative Knowledge Representation Language. This is a high-level n-ary conceptual tool specially conceived for the representation and management of real world, dynamically characterized entities like situations, events and complex events, actions (e.g., in a robotics context) scripts/scenarios/ narratives etc. After having pointed out some shortcomings of the standard ontological solutions for dealing with this sort of information, and having recalled some general characteristics of NKRL (like the addition of an "ontology of events" to the usual "ontology of objects"), we focus on the rules/inferential aspects proper to this language. We introduce, then, the general, formal model of "rule" used in an NKRL context and we show how this can be appropriately adapted to the setup of advanced types of inference operations based, e.g., on "analogical" and "causal" reasoning.

Keywords: Ontologies; binary and n-ary approaches; inference rules and inference techniques.

1 Introduction

In any real-world domain involving entities that communicate with each other (robotics for example), one of the basic requirements for being able to create effective applications is the availability of some sort of *common vocabulary*, supplemented with clear and exhaustive definitions of its terms. Access to this vocabulary will ensure common understanding among the different entities, like humans and robots, involved in a given application. Moreover, it will guarantee the possibility of a uniform description of the properties/characteristics of these entities and of their global context, and facilitates more efficient data integration and transfer of information within the application's environment.

Research within the Artificial Intelligence (AI) and Knowledge Representation (KR) domains has identified in "*ontologies*" and "*ontological engineering*" the tools allowing us to create common vocabularies endowed with the above characteristics. According to a modern computer-science meaning, a consensus definition of ontology says that, "Ontologies represent a formal and explicit specification of a shared conceptualization" [7]. "Conceptualization" refers here to an *abstract model* of some phenomenon/situation in the world, where the model results by the identification of the relevant "concepts" that characterize this particular phenomenon/situation. "Explicit" and "formal" mean that the type of concepts used and the constraints on their use are *unambiguously defined* and that the ontology should be machine-usable. "Shared" signifies that an ontology captures *consensual knowledge*, that is, this knowledge is not private but must be accepted by a group. In recent times, ontologies have been defined and used in the most different application domain – see, e.g., Section 4 below for information about some current ontological systems used in a robotics context.

Unfortunately, in spite of some attempts in the past to define general methodologies for creating well-formed, completed and shareable ontologies – see, for example, [5,8,27] – there is at present no agreement on how to achieve this goal. Any general approach to build up this sort of wide-ranging ontologies should in fact be capable to take effectively into account all the *pertinent knowledge* actually needed to implement real world applications. However, when we look at the existing ontologies, we see that their vast majority are implemented by using, from a knowledge representation point of view, the so-called "*binary*" approach. In this, a concept is defined through a set of properties/attributes; when the concept is instantiated into a concrete individual, each associated property can only link this individual to another individual or a value, *individual1-property-individual2/value* – note that this is also the format of the so-called "RDF-triples" [13]. This approach is, undoubtedly, wholly justified when the ontologies to be defined concern only the (so-called) "*static*" entities. These are important *unique* notions ("concepts") – like physical objects, animated entities, social bodies, political theories, university courses, types of illness etc. – that, at a given point in time, can be defined in terms of binary properties and autonomously inserted within a hierarchical structure of similar notions/concepts (the ontology). Their original definitions are then *enhanced* through the grid of implicit/explicit relationships (essentially, IsA links) with the other terms of the ontology – further information can also be added making use of natural language (NL) annotations see, e.g., the definitions of the "classes" of the Schema.org initiative [9].

The binary model – a sort of modern reinterpretation of the traditional "terminological" approach – faces, however, important "expressiveness" problems when the *pertinent knowledge* to be taken into account includes also some forms of "*dynamic*" knowledge. This knowledge concerns, in fact, the use of *evolving structured entities* that denote, within a given spatio-temporal context, a coherent set of mutual connections between *multiple* (>2) "static" entities. These dynamic entities can correspond to events, situations, actions, behaviors, attitudes, scripts, scenarios, narratives etc., and are particularly significant in a robotics context. A simple example is, e.g., the "purchase relation" that concerns events where a seller, a buyer, a good, a price, and a timestamp are involved. In an AAL (Ambient Assisted Living) context, another example concerns a specific case of the "transmission of a message" relation that associates a robot, a warning message, an elderly person and a timestamp. In all these cases, a simple representation in form of binary relationships is no more sufficient, and actual (complex) n-ary relationships must then be used to represent the dynamic knowledge, see the details in Section 2 below. Note that a common misunderstanding consists in asserting that the definition of specific n-ary knowledge representation structures for dealing with dynamic knowledge is not necessary given that any n-ary relationship can be simply reduced to a set of binary relationships. This sort of decomposition can be important for *very practical problems* like storing efficiently n-ary relationships into standard databases. However, binary and n-ary relationships are *conceptually irreconcilable*, and an n-ary relation cannot be reduced to the simple addition of binary elements without losing its "*deeper meaning*" – it is impossible to reason about, e.g., the possible reasons of a purchase or the context of a warning message without considering the initial event in its whole conceptual entirety. This is well-known in the AI/KR milieus where the researchers have tried vainly, for a long time, to transform automatically/semi-automatically binary representations into n-ary ones; a well-known proposal in this context is, e.g., [20]. The questionable character and the practical ineffectiveness of this sort of proposals are described in full in [29].

All the above should make us recognize that the standard Semantic Web (SW) languages, OWL, OWL 2, SPARQL, SKOS, SWRL etc., the "W3C languages", do not represent an optimal solution for representing in formal terms those "states of affairs" that imply the presence of dynamic knowledge. Being based on-top of the binary RDF these languages are, in fact, unable to represent *natively, in a compact and natural way,* the n-ary relationships between multiple related entities within a given spatio-temporal framework. In this paper, we introduce then a knowledge representation tool, the Narrative Knowledge Representation Language (NKRL) [32], which is supported by a fully implemented computer science environment. NKRL has been expressly created to formalize as accurately as possible and to manage in an integrated efficient way *both* those static and dynamically characterized entities evoked above; in this paper we will deal, in particular, with the *NKRL rule-based inferencing techniques*, referring the reader to other NKRL publications for a complete understanding of this language.

In the following, Section 2 will be devoted to a short description of the main conceptual features of NKRL. Section 3 will describe NKRL's inference techniques. Section 4 concerns some comparisons with work related, in some way, to the specific NKRL's approach; Section 5 supplies, eventually, a short "Conclusion".

2 A short revue of the main features of NKRL

NKRL innovates with respect to the current ontological paradigms by adding an "*ontology of elementary events*" taking in charge, mainly, the dynamic knowledge, to the usual "*ontology of concepts*" devoted to the representation of the background static knowledge.

The ontology of concepts is called HClass (hierarchy of classes) in an NKRL context and includes presently (June 2019) more than 7,000 standard concepts – "standard" means here that the properties or attributes used to define a given concept are simply expressed according to the usual binary approach. From a purely operational point of view HClass – see [32, 43-55, 123-137] – is not so different, then, from the ontologies that can be built up by using the *frame version* of Protégé [19].

The ontology of elementary events is, by contrast, a *new sort* of hierarchical organization where the nodes correspond to *n-ary structures called "templates"*: their basic structure is depicted by Eq (1). This ontology is then denoted in NKRL as HTemp (hierarchy of templates). In opposition to the usual ontological function of the static/background notions denoted by the HClass concepts like "human being", "amount", "color", "artefact", "control room", "valve", "level of temperature" ..., templates' function concerns the representation in machine-understandable format of real world *dynamically characterized entities* like events, situations, actions, behaviors, scripts, scenarios, narratives etc. More precisely, they must be conceived as the canonical, formal representation of *general classes of elementary events* like "move a physical object", "be present in a place", "having a specific attitude towards someone/something", "asking/receiving an advice", etc. – for the sake of simplicity, we group here under the label "elementary event" all the constituent elementary entities of actions, situations, behaviors, narratives etc. that can be described fully by Eq (1). As we will see later, these elementary events and their instances can be assembled in turn into more complex formal dynamic structures.

$$(L_i(P_j(R_1a_1)(R_2a_2)\ldots(R_na_n))) \tag{1}$$

In Eq (1), L_i is the symbolic label identifying (reifying) the particular n-ary structure corresponding to a specific template. P_j is a conceptual predicate. R_k is a generic functional role [36] used to identify the specific logico-semantic function performed by its "filler" a_k, a generic predicate argument, with respect to the predicate P_j, see the example below. a_k is then the particular predicate argument introduced by the role R_k. When a template denoted as Move:TransferMaterialThingsToSomeone in NKRL is *instantiated* to provide the representation of a quite simple elementary event like "Bill gives a book to Mary", the predicate P_j (MOVE) will introduce its three arguments a_k, JOHN_, MARY_ and BOOK_1 ("*individuals*", i.e., instances of HClass concepts) via, respectively, the three functional relationships (R_k roles) SUBJ(ect), BEN(e)F(iciary) and OBJ(ect). These last specify, then, the function/task/role of each of the three individuals with respect to the predicate and, as a consequence, with respect to the global meaning of the (formalized) elementary event. The global n-ary construction is then reified through the symbolic label L_i. The instances of templates are called *predicative occurrences* and correspond then to the representation of specific elementary events; note that the same line of reasoning could be used for an example like “a robot sends a warning message to an elderly person” see, e.g., [34, 2373].

Note that the use of functional roles in Eq (1) is the most important innovation introduced by NKRL with respect to other n-ary proposals, those in a “frame” style for example – see Section 4. They allow us, in fact, to obtain a *high degree of accuracy* in the representation of the “deeper meaning” of the original elementary events (actions, situations, behaviors etc.). Note also the importance of L_i in the context of Eq (1)’s symbolism. This label has, in fact, the function of the “*unique internal handle*” already proposed by Woods in 1975 [32] to allow us to manage all the constituents of the formal representation of an elementary event as a *unique and coherent* block. This block can, then, be associated with similar blocks within (potentially very large) consistent conceptual structures.

To avoid the ambiguities of natural language and any possible combinatorial explosion problem, see [32, 56-61] in this context, both the conceptual predicate of Eq (1) and the associated functional roles are “*primitives*”. Predicates P_j pertain to the set {BEHAVE, EXIST, EXPERIENCE, MOVE, OWN, PRODUCE, RECEIVE}, and the roles R_k to the set {SUBJ(ect), OBJ(ect) SOURCE, BEN(e)F(iciary), MODAL(ity), TOPIC, CONTEXT}. Fig. 1 reproduces a fragment of the HTemp hierarchy that displays the conceptual labels of some templates included in the Move: and Produce: sub-hierarchies. As it appears from this figure, HTemp is structured into seven branches, where each branch includes only the templates created, according to the general syntax of Eq (1), around one of the seven predicates P_j admitted by the NKRL language.

Table 1 represents the template Move:TransferMaterialThingsToSomeone (see also the branch Move:TransferToSomeone in Fig 1) used to produce the predicative occurrence formalizing the elementary event “Bill gives a book to Mary”. The items (as SOURCE, MODAL, (*var2*) etc. in Table 1) included in square brackets are optional. HTemp consists presently (June 2019) of more than 150 templates, very easy to specialize and customize, see [32, 137-177]. As we can see from Table 1, the arguments of the predicate (the a_k terms in Eq (1)) are actually denoted by *variables (var$_i$) with associated constraints*. These last are expressed as concepts or combinations of concepts, i.e., using HClass terms. When creating a predicative occurrence as an instance of a template, the constraints linked to the variables are used to specify the legal sets of HClass terms that can be substituted for these variables within the occurrence. In our example, e.g., we must verify that JOHN_ and MARY_ are HClass instances of individual_person, a specific term of human_being_or_social_body, see the constraints on the SUBJ and BENF roles of the template of Table 1.

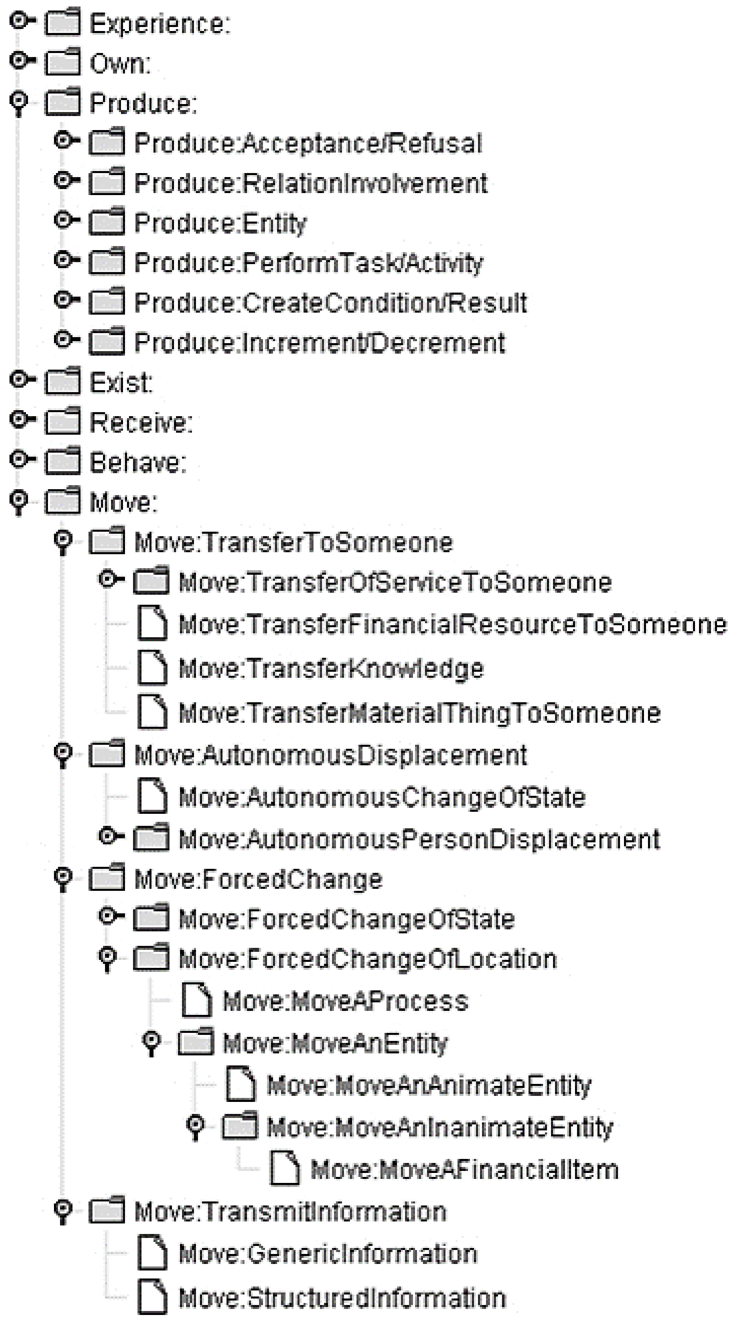
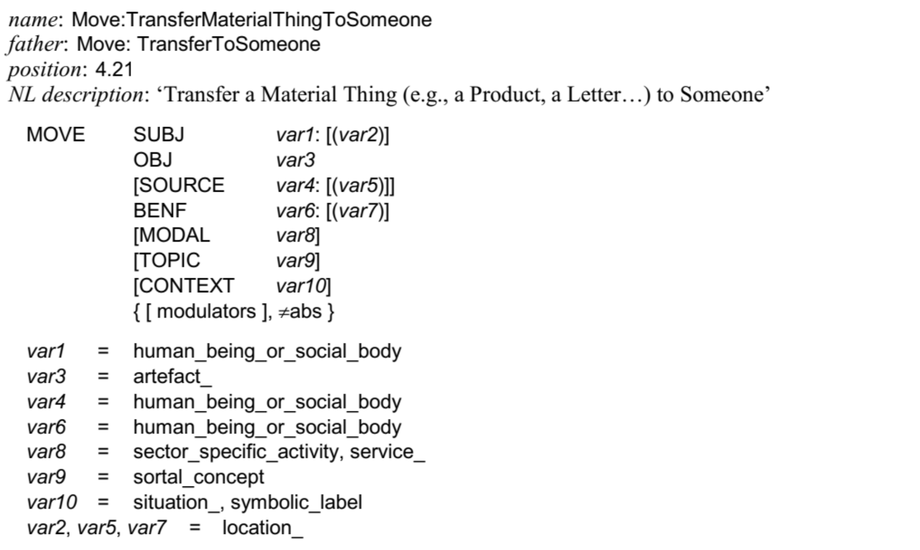

Experience:
Own:
Produce:
Produce:Acceptance/Refusal
Produce:RelationInvolvement
Produce:Entity
Produce:PerformTask/Activity
Produce:CreateCondition/Result
Produce:Increment/Decrement
Exist:
Receive:
Behave:
Move:
Move:TransferToSomeone
Move:TransferOfServiceToSomeone
Move:TransferFinancialResourceToSomeone
Move:TransferKnowledge
Move:TransferMaterialThingToSomeone
Move:AutonomousDisplacement
Move:AutonomousChangeOfState
Move:AutonomousPersonDisplacement
Move:ForcedChange
Move:ForcedChangeOfState
Move:ForcedChangeOfLocation
Move:MoveAProcess
Move:MoveAnEntity
Move:MoveAnAnimateEntity
Move:MoveAnInanimateEntity
Move:MoveAFinancialItem
Move:TransmitInformation
Move:GenericInformation
Move:StructuredInformation

Fig. 1. Partial image of HTemp, with the Produce: and Move: branches partly unfolded.

Table 1. A template of the Move: branch of HTemp.



name: Move:TransferMaterialThingToSomeone
father: Move: TransferToSomeone
position: 4.21
NL description: ‘Transfer a Material Thing (e.g., a Product, a Letter…) to Someone’

```
MOVE      SUBJ         var1: [(var2)]
          OBJ          var3
          [SOURCE      var4: [(var5)]]
          BENF         var6: [(var7)]
          [MODAL       var8]
          [TOPIC       var9]
          [CONTEXT     var10]
          { [ modulators ], ≠abs }

var1    =   human_being_or_social_body
var3    =   artefact_
var4    =   human_being_or_social_body
var6    =   human_being_or_social_body
var8    =   sector_specific_activity, service_
var9    =   sortal_concept
var10   =   situation_, symbolic_label
var2, var5, var7   =   location_
```

What described until now illustrates the NKRL solutions to the problem of providing a complete representation of single elementary events. New problems arise when, in the context of larger structured/dynamic arrangements like *complex events, narratives, scripts, scenarios etc.*, several elementary events must be linked through “connectivity phenomena” operators as causality, goal and indirect speech. In this case, we make use of second order structures created through reification of the predicative occurrences corresponding to the events to be associated. The reification is implemented using their symbolic labels (the L_i terms of Eq (1)) according to two important conceptual mechanisms.

The first concerns the possibility of referring to an elementary (or complex) event *as an argument* of another (elementary) event – the corresponding connectivity phenomenon in natural language terms is the “*indirect speech*”. An example can concern an elementary event X where someone speaks about Y, where Y is itself an elementary event (or a complex event, i.e., a logically-semantically coherent set of events). The second (more general) mechanism allows us to associate together, through several types of connectivity operators, elementary (or complex) events that, at the difference of the previous case, can still be regarded as *independent entities*. As an example, we can consider an elementary or complex event X being linked to another elementary or complex (independent) event Y by standard connectivity cues as causality, goal, coordination, alternative etc. relationships. In NKRL, the first relational mechanism is called “*completive construction*” and the second “*binding occurrences*”. Detailed explications about the two mechanisms can be found, e.g., in [34,35], where the NKRL representations of complex narratives are also supplied.

3 The NKRL querying/inferencing system

The techniques used for exploiting a knowledge base (*KB*) of elementary/complex events represented in NKRL format range from direct questioning using specific data structures called "*search patterns*" to high-level inference procedures, e.g., "*transformations*" and "*hypotheses*". These last utilize advanced reasoning processes grounded, e.g., on analogical reasoning (transformation rules) and causal-like explication (hypothesis rules) techniques.

3.1 "Search patterns" and the filtering unification module

The basic building block for all the NKRL querying and inference procedures is represented by *Fum*, the "Filtering Unification Module" [32, 187-194]. *Fum* takes as input specific NKRL data structures called "*search patterns*", p_i; search patterns' task consists in specifying the syntactic format of the information to be searched for within an NKRL knowledge base. Search patterns are used according to two different modalities:

- A search pattern p_i can be created by the user through a suitable query interface. In this case, it must be interpreted as a standard *formal query*, to be used according to the usual querying modalities of an (advanced) information retrieval system.
- Search patterns p_i with the same formal characteristics of the "manual" patterns can also be generated *automatically* by the different versions of the NKRL *InferenceEngine* when a particular step of a high-level inference procedure must be executed, see the next sub-sections. More exactly, the validation of this step will be performed by *InferenceEngine* by automatically transforming the step into a "standard" p_i.

Formally, search patterns p_i correspond to specialized/partially instantiated templates where all their "*explicit variables*" – identified by conceptual labels in the var_i style, see Table 1 above – have been replaced by concepts/individuals compatible with the original constraints imposed on these variables. The HClass concepts included in this way in the p_i play now the role of "*implicit variables*". This means that, when the comparison between p_i and the predicative occurrences c_j of the *KB* takes place, the concepts of the p_i can match not only the identical c_j concepts, if any, but also the instances (individuals) of these last concepts and all their specific terms with the corresponding instances. More details about search patterns, including the detailed specifications of the matching operations with the occurrences of the *KB* can be found, e.g., in [32, 183-201].

3.2 General properties of the NKRL inference rules

These rules are qualified by the following general properties.

- All the NKRL high-level inference rules, e.g., transformations and hypotheses, can be described as:

$$X \star Y_1 \text{ and } Y_2 \ldots \text{ and } Y_n \tag{2}$$

- In Eq (2), the head X corresponds either to a predicative occurrence c_j (formal representation of an elementary event) in a hypothesis context or to a search pattern p_i (formal representation of a generic query) in a transformation context. The logical connective star operator, "$\star$", must be interpreted as a "*biconditional*" ($\leftrightarrow$) in a hypothesis context and as a "*material implication*" ($\rightarrow$) in a transformation context, see below for details.
- The n-ary atoms $Y_1 \ldots Y_n$ correspond to *partially instantiated templates*; these atoms represent the translation into NKRL terms of the "reasoning steps" that make up the rule. At the difference, however, of the partially instantiated templates used for the search patterns p_i, they include now *explicit variables of the form var$_i$*. We can note that each $Y_1 \ldots Y_n$ atom is created independently according to the (rich) n-ary format of Eq (1). With respect then to the expressiveness problems, the NKRL high-level inference rules are now free from any of the *binary limitations* mentioned in Section 1, see also the concrete examples thereafter.
- Given the presence of co-reference constraints that link together the different Y_i atoms and are represented, among other things, by the names of the explicit variables, the Eq (2) rules must be executed by a full *InferenceEngine*. According to the procedural interpretation normally utilized in a logic programming context [17, 112-121], the different versions of the *InferenceEngine* understand each rule as a *procedure*. This allows us to reduce "*problems*" of the form X to a succession of "*sub-problems*" of the form Y_1 and ... Y_n.
- In an NKRL rule context, each Y_i is interpreted in turn as a procedure call that tries to convert Y_i into (at least) a *successful* search pattern p_i, able then to match one or several predicative occurrences c_j of the *KB*. Backtracking techniques (see below) will be used, in case, during this conversion process.
- The success of the matching operations p_i/c_j for a p_i pattern derived from Y_i means that the reasoning step represented by Y_i has been validated. *InferenceEngine* continues then its work trying to validate the reasoning step corresponding to sub-problem Y_{i+1}. In line with the presence of the operator "and" in Eq (2), the logical expression represented by Eq (2) is *fully validated* iff all the reasoning steps $Y_1,\ Y_2 \ldots\ Y_n$ are validated.

An in-depth description of the different versions of *InferenceEngine* is beyond the possibilities of the present paper, see [31,32, 201-234] in this context. We will only note that, from an algorithmic point of view, all these versions work according to a *backward chaining approach with chronological backtracking*. The differences with respect to other applications of this approach (see, e.g., PROLOG) are mainly linked with the unusual complexity and expressiveness proper to the n-ary NKRL data structures.

For example, after a deadlock due to the failure of a p_i/c_j matching actions when trying to validate a reasoning step corresponding to Y_i, *InferenceEngine* must carry out some complex restoring procedures of the program environment in order to return to the previous choice point. In this context, four *main* "environment variables" are used:

- VALAFF holds the values *provisionally* affected to the var_i included in the n-ary atoms $Y_1 \ldots\ Y_n$ that implement the reasoning steps of the inference rules. These values can be *deleted*, in case, after a backtracking operation.
- DESVAR holds the *final* values associated with the variables var_i when the successful processing of one of the reasoning steps has been definitively completed.
- RESTRICT holds the constraints (HClass terms) associated with the variables var_i of the different atoms/reasoning steps: these will be used to build up all the different search patterns that can be derived from these schemata, see below.
- OCCUR holds the list of the symbolic names of all the c_j predicative occurrences involved in the p_i/c_j matching operations. The values bound to var_i that have been retrieved in these occurrences are then used to build up the VALAFF lists.

Let us suppose that, during the processing of a high-level inference rule, we are trying to validate the reasoning step (n-ary atom) Y_i reproduced in Table 2 – without worrying whether Y_i is now part of a "transformation" or "hypothesis" or any other type of rule. As already stated, the validation succeeds iff (at least) a p_i derived from Y_i can successfully match, using *Fum*, (at least) a predicative occurrence c_j stored in the KB. For the sake of simplicity, we suppose here that *InferenceEngine* has already successfully performed the "pre-selection" operations aimed at building up a subset of the NKRL *KB* including only the occurrences surely compatible with the Y_i to validate – e.g., those having the same semantic predicate [32, 194-201].

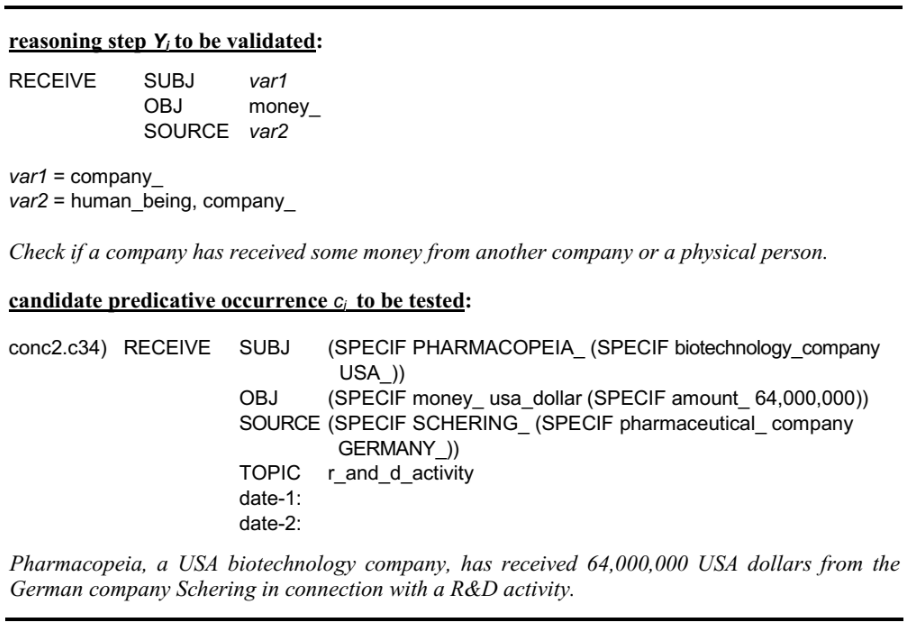
Under these conditions, we can assume that one of the occurrences of this subset is conc2.c34 of Table 2; conc2.c34 is then stored within OCCUR. Note, in this occurrence, the recursive utilization of the SPECIF(ication) operator to build up "*structured arguments*" a_k of the predicate, see Eq (1), under the form of lists of properties/attributes that characterize the first element of each SPECIF list [32, 68-70]. The different search patterns needed for the p_i/c_j matching operations corresponding to this occurrence are now derived by systematically substituting to the variables *var1* and *var2* included in the Y_i we are trying to validate, see Table 2, their associated constraints (suitably stored within RESTRICT). We recall here, see sub-section 2.1 above, that the search patterns p_i only admit implicit variables.

Variable $var1$ can be substituted only by the constraint company_; in contrast, two substitutions, $var2$ = human_being and *var2* = company_ are possible for *var2*. A first search pattern will be then built up by substituting company_ for *var1* in Y_i and, according to the chronological approach, the first *var2* constraint, i.e., human_being, for *var2* (company_ and human_being are also provisionally associated, respectively, with *var1* and *var2* in VALAFF). *Fum* will then try to execute a match with conc2.c34 using a search pattern corresponding to a payment to a company *done by an individual person* (SOURCE human_being) instead of a company (SCHERING_, see conc2.c34). This match obviously fails. The engine then backtracks making use of a particular module, called *Reexec*, which is the same in all the versions of *InferenceEngine*. *Reexec* is systematically used to:

- carry out the backtracking operations when a deadlock occurs, and
- reconstruct, making use of the environment variables, the data structures (the environment) proper to the previous choice point.

The association *var2* = human_being is then removed from VALAFF and, using the constraints stored in RESTRICT, the engine builds up a new pattern from Y_i using now the values *var1* = *var2* = company_. This time, *Fum* will succeed in unifying the new pattern with conc2.c34. SCHERING_ is, in fact, stored in HClass as an instance of pharmaceutical_company: this last is, obviously, a company_. The two values *var1* = PHARMACOPEIA_ and *var2* = SCHERING_ will then be stored in DESVAR for use in the Y_{i+1} etc. inference steps of the rule.

Table 2. Backtracking and automatic construction of a search pattern.



reasoning step Y_i to be validated:

```
RECEIVE      SUBJ      var1
             OBJ       money_
             SOURCE    var2
```

var1 = company_
var2 = human_being, company_

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

candidate predicative occurrence c_j to be tested:

```
conc2.c34)  RECEIVE   SUBJ     (SPECIF PHARMACOPEIA_ (SPECIF biotechnology_company
                                 USA_))
                      OBJ      (SPECIF money_ usa_dollar (SPECIF amount_ 64,000,000))
                      SOURCE   (SPECIF SCHERING_ (SPECIF pharmaceutical_ company
                                 GERMANY_))
                      TOPIC    r_and_d_activity
                      date-1:
                      date-2:
```

Pharmacopeia, a USA biotechnology company, has received 64,000,000 USA dollars from the German company Schering in connection with a R&D activity.

The NKRL inference procedures concern mainly two classes of rules, “transformations” and “hypotheses”; see, e.g., [32, 234-239] for other types of rules, in particular, the “filtering rules”.

3.3 Transformation rules

Transformation rules try to adapt, from a semantic point of view, a well-formed search pattern p_i that failed (that was unable to find a match within the *KB*) to the real contents of this base making use of a sort of analogical reasoning. Hence, in a transformation context, the head X of Eq (2) is now expressed in search pattern format: the aim of this type of rules is to automatically transform a given p_i into one or more different $p_1,\ p_2\ \dots\ p_n$ that are not “syntactically equivalent” but only “semantically close” to the original one. Transformation rules can be interpreted according to a production rules paradigm: if a search pattern p_i exists (i.e., it has been previously created, manually or automatically, by an *InferenceEngine*) then a suitable transformation rule can substitute p_1, $p_2\ \dots\ p_n$ for the original p_i. The generic “$\star$” connective of Eq (2) can therefore be changed into the *material implication symbol*, “$\rightarrow$”.

For an example of transformation, let us suppose we ask: “Search for the evidence of the existence of a serious alarm situation in an industrial oil/gas premises”, see also [33]. If it is impossible to obtain a direct answer, and if the appropriate transformation rule exists, the original, manually created p_i search pattern can be automatically changed by the transformation version of *InferenceEngine* into two new, logically linked patterns: p_1) “search for information reporting that the working staff is moving massively to a *new location*”; p_2) “search for information confirming that the new location is *outside* the original gas/oil premises”. If the two new patterns are both able to unify some congruent information in the KB, we can consider that this information is a sort of indirect, useful answer to the query originally posited.

For the sake of clarity, and in compliance with the usual production rules practices, it can be useful to denote the transformation rules as made up of a left-hand side, the “*antecedent*” and one or more right-hand side(s), the “*consequent(s)*”. The antecedent is then the general formulation, in search pattern format, of the formal query to be transformed, and the consequent(s) is/are the NKRL representation(s) of one or more search patterns to be substituted for the transformed one. Denoting with A the antecedent and with Cs_i all the possible consequents, for transformation rules Eq (2) can be expressed as:

$$A(var_i) \rightarrow Cs_i \qquad var_i \subseteq var_j \tag{3}$$

The restriction $var_i \subseteq var_j$ – all the variables declared in the antecedent A must also appear in the Cs_i accompanied, in case, by additional variables – corresponds to the usual safety condition that assures the logical congruence of the rules. The “transformation arrow” of Eq (3), “$\rightarrow$”, can be interpreted according to a double reading:

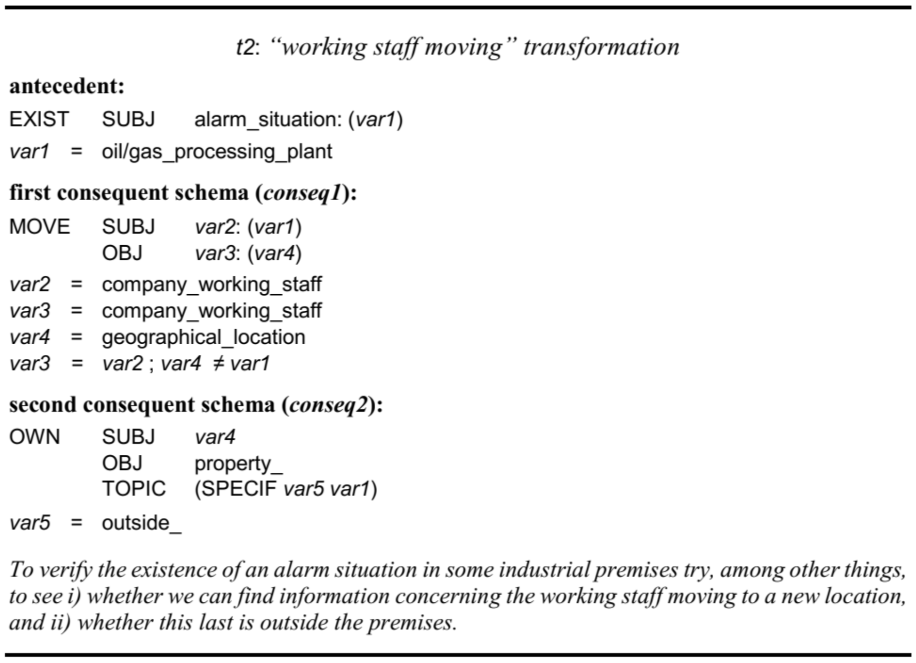
- Operationally speaking, the arrow indicates the direction of the transformation. The original search pattern (a specialization of the left-hand side A

of the transformation rule) is then removed and replaced by one or several new patterns obtained through the updating, using the parameters of the original pattern, of the right-hand side Cs_i.

- From a semantic point of view, we assume that between the information retrieved through Cs_i and the information we wanted to obtain via the original p_i there is a sort of implication relationship – that, normally, denotes solely a possible (a weak) implication.

An NKRL representation of the above "working staff moving" transformation is reproduced in Table 3.

Table 3. An example of transformation rule.



t2: *"working staff moving" transformation*

antecedent:

```
EXIST   SUBJ    alarm_situation: (var1)
var1  =  oil/gas_processing_plant
```

first consequent schema (*conseq1*):

```
MOVE    SUBJ    var2: (var1)
        OBJ     var3: (var4)
var2  =  company_working_staff
var3  =  company_working_staff
var4  =  geographical_location
var3  =  var2 ; var4 ≠ var1
```

second consequent schema (*conseq2*):

```
OWN     SUBJ    var4
        OBJ     property_
        TOPIC   (SPECIF var5 var1)
var5  =  outside_
```

To verify the existence of an alarm situation in some industrial premises try, among other things, to see i) whether we can find information concerning the working staff moving to a new location, and ii) whether this last is outside the premises.

The antecedent of this rule corresponds to a specialization of the Exist:SituationBePresent template; it must be matched to the original search pattern that failed. The Exist: templates [32, 155-159] are necessarily characterized by the presence of a "*location of the SUBJ(ect)*", represented in Table 3 by the variable var_1. In NKRL, determiners/attributes of the "location" type are associated through the "colon" operator, "**:**", with the arguments of the predicate (i.e., the fillers) introduced by the SUBJ, OBJ, SOURCE and BENF functional roles of a template or predicative occurrence [32, 75-76]. According to the associated constraint, *var1* can be instantiated by specializations or instances pertaining to the oil/gas_processing_plant sub-hierarchy of HClass. The first consequent schema (*conseq1*) is a specialization of the Move:AutonomousPhysicalPersonDisplacement template, used to characterize the autonomous movement of a person or social body. A characteristic of this template is that the person or social body, as a SUBJ(ect), is seen as moving herself/himself/itself as an OBJ(ect), see the constraint that forces the equality between the values bound to the *var2* and *var3* variables. The "location of the SUBJ(ect)" corresponds then to the starting point l_1, the "location of the OBJ(ect)" to the arrival point l_2. The second consequent schema (*conseq2*) is a specialization of the Own:CompoundProperty template. It is used to find, among the predicative occurrences c_j of the *KB*, elementary events signaling that location l_2 is really outside the endangered plant (l_1). In this consequent schema, the identification of the property_ (the filler of the TOPIC role) is represented by a SPECIF(ication) list where the first argument (*var5*) must be a specialization of concepts like part/whole_relationship, relational_property and spatio/temporal_relationship. outside_ is, accordingly, a specific term of spatio/temporal_relationship.

3.4 Hypothesis rules

Hypothesis rules allow us to build up automatically a sort of *causal explanation* for information (i.e., for a predicative occurrence c_j) already retrieved making use of a search pattern p_i in a querying-answering mode. In a hypothesis context, the head X of Eq (2) is then expressed in predicative occurrence format. We can now interpret Eq (2) according to the following analysis: *X* *(*c_j*)*, an elementary event, occurs iff other elementary events corresponding to the reasoning steps Y_1 and Y_2 ... and Y_n have actually taken place. Conversely, the occurrence of *X* *(*c_j*)* implies that some events corresponding to Y_1 and Y_2 ... Y_n really happened. Hence, Eq (2) has the following format for the hypothesis rules:

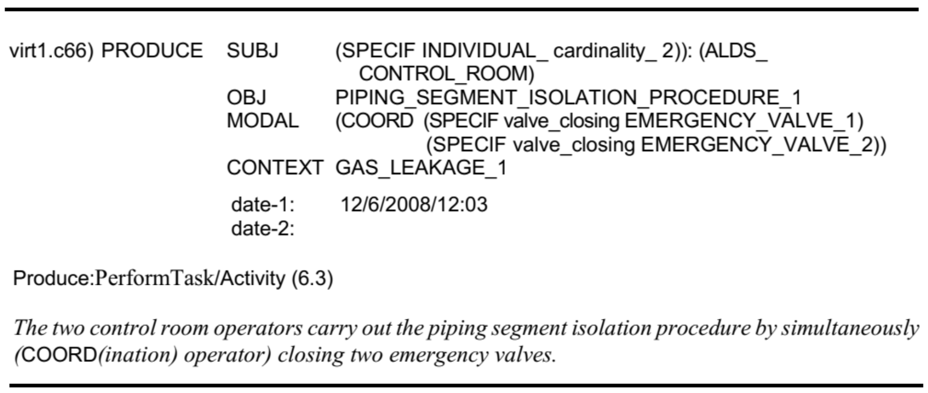
$$X \leftrightarrow Y_1 \text{ and } Y_2 \ldots Y_n \tag{4}$$

where the "*biconditional*" operator, "$\leftrightarrow$", has been substituted for the generic "$\star$". As usual, the reasoning steps (n-ary atoms) Y_i of Eq (4) – called "condition schemata" in a hypothesis context – must all be satisfied (for each of them, at least one of the corresponding search patterns p_i must find a successful unification with the *KB*) in order that the set of c_1, c_2 ... c_n occurrences retrieved in this way can be interpreted as a context/causal explanation of the original occurrence c_j corresponding to X in Eq (4).

Let us suppose, for example, we have directly retrieved, in a querying-answering mode, information like: "The control room operators of a particular gas/oil station (ALDS) have carried out a piping segment isolation procedure in the context of an industrial accident", see Table 4 and [33]; virt1.c66 is then the occurrence c_j to be explained.

A hypothesis rule whose "*premise*" can match c_j, i.e., virt1.c66, is reproduced in Table 5; the premise is the triggering pattern corresponding to the head X of Eq (4).

Table 4. Execution of an emergency procedure.



```
virt1.c66) PRODUCE  SUBJ     (SPECIF INDIVIDUAL_ cardinality_ 2)): (ALDS_
                                 CONTROL_ROOM)
                    OBJ      PIPING_SEGMENT_ISOLATION_PROCEDURE_1
                    MODAL    (COORD (SPECIF valve_closing EMERGENCY_VALVE_1)
                                    (SPECIF valve_closing EMERGENCY_VALVE_2))
                    CONTEXT  GAS_LEAKAGE_1
                    date-1:  12/6/2008/12:03
                    date-2:

Produce:PerformTask/Activity (6.3)
```

*The two control room operators carry out the piping segment isolation procedure by simultaneously (*COORD*(ination) operator) closing two emergency valves.*

It is easy, in fact, to verify (using *Fum*) that occurrence virt1.c66 in Table 4 is an instance of the search pattern p_i obtained from the premise of the rule of Table 5. We can then try to build up, using this rule, a sort of "*causal explanation*" of the triggering event (the isolation procedure) by retrieving information in the style of: i) "someone has previously attempted to activate a (less drastic) corrective maintenance procedure" (*cond1*); ii) "this corrective maintenance has failed" (*cond2*) and iii) "the accident is considered as a serious one" (*cond5*). We must also verify that the person at the origin of the "corrective maintenance procedure" is a field operator (*cond4*) and that those who have implemented the "piping segment isolation procedure" are control room operator/s (*cond3*).

Matching the premise of the rule of Table 5 with the occurrence of Table 4 allows us to associate the value INDIVIDUAL_PERSON_98 to *var1* and GAS_LEAKAGE_1 to *var2,* by verifying also the constraints associated with *var1* (human_being) and *var2* (industrial_accident). These values are transmitted to the condition schemata and used – along with those found, making use of backtracking techniques when needed, for the residual variables – to build up the corresponding search patterns p_i. Iff all the Y_i have been satisfied, we can affirm that a *possible* explication of the original event has been built up. Note that, as normal in a hypothesis context, the explication proposed by the rule of Table 5 corresponds to only one of all the possible hypotheses about the "causes" of the original event: a particular hypothesis rule must always be conceived as a member of a family of possible explications.

To conclude about the inference procedures, we can also note that an interesting feature of the NKRL rule system concerns the possibility of making use of "transformations" when working in a "hypothesis" context – i.e., of utilizing these two modalities of inference in an integrated way. This means that, whenever a search pattern p_j is derived from a condition schema Y_i of a hypothesis to implement a step of the reasoning process, we can use this pattern as it has been automatically built up by *InferenceEngine* from its father condition schema, but

Table 5. An example of hypothesis rule.

h1: *"isolation procedure" hypothesis*

premise:

```
PRODUCE   SUBJ      var1
          OBJ       isolation_procedure
          CONTEXT   var2
var1  =  human_being
var2  =  industrial_accident
```

An individual has carried out an isolation procedure in the context of an industrial accident.

first condition schema (*cond1*):

```
PRODUCE   SUBJ    var3
          OBJ     var4
          TOPIC   var2
var3  =  human_being
var3  ≠  var1
var4  =  corrective_maintenance_procedure
```

Another individual had carried out a (milder) corrective maintenance procedure.

second condition schema (*cond2*):

```
EXPERIENCE   SUBJ    var3
             OBJ     var5
             TOPIC   var4
var5  =  failure_
```

This second individual has experienced a failure in this corrective maintenance context.

third condition schema (*cond3*):

```
BEHAVE   SUBJ    var1
         MODAL   var6
var6  =  control_room_operator
```

The first individual was a control room operator.

fourth condition schema (*cond4*):

```
BEHAVE   SUBJ    var3
         MODAL   var7
var7  =  field_operator
```

The second individual was a field operator.

fifth condition schema (*cond5*):

```
OWN   SUBJ    var2
      OBJ     property_
      TOPIC   (SPECIF strength_ var8)

var8  =  important_
```

The industrial accident is considered as a serious one.

also in a *transformed form* if the appropriate transformation rules exist. In this way, a hypothesis that was deemed to fail because of the impossibility of deriving a successful p_j from one of its condition schemata can continue if a new p_j, obtained by transformation, will find a successful unification within the base [32, 216-231] and [31].

4 Some comparisons

What expounded in the previous Section could be summarized by suggesting that the use of a rich n-ary approach in the NKRL style could be very important for creating *highly expressive inference rules*. In these, the "*atoms*" can represent directly complex situations and events, actions (e.g., in a robotics context), scenarios, narratives etc. without being limited to the use of (inexpressive) *binary clauses*.

The advantages linked to the setup of knowledge representation languages utilising rule systems that are not systematically restrained to make use of "binary atoms" are known for a long time. We can mention in this context the development of those languages that can be denoted (broadly speaking) as *frame-based languages*, see the well-known Minsky's paper [15]. They make use of an n-ary form of knowledge representation consisting of a collection (a "*frame*") of slot/value pairs that are semantically related (i.e., the collection can be understood as a *single unit*) given that they all refer to the same symbolic name identifying the frame. This representation schema can be likened to a *simplified form* of Eq (1), where the conceptual predicate P_j has been combined with the symbolic label L_i to give rise to the generic "name" of the frame, like Eiffel-Tower, Event234, Want-01, Human-97 etc. The NKRL functional roles R_k are now reduced, in practice, to generic *properties/attributes slots*, even if in some frame systems used in a specific Computational Linguistics context they are still equated to "roles" and identified by purely numerical labels like Arg0, Arg1, Arg2..., see [2] for example. Important knowledge representation and reasoning systems that can be equated to frame systems are [12,18,21]. Several *n-ary, frame-based programming languages* have also been developed after the publication of the Minsky's paper; we will only mention here the (probably) best-known one, F-Logic [10]. This language, originally developed in a deductive databases framework, combines the declarative semantics of the deductive database languages with the data modelling capabilities of the frame/object-oriented data models.

To find, then, valid analogies between NKRL and comparable tools, it is necessary to look at that class of pure AI-based n-ary systems that agree on the *paramount importance* of a precise definition of the *"function" (role)* played by the different entities involved in events, actions, scenarios, narratives etc. for a *complete understanding* of the situation under consideration – see [36] for a complete analysis of this topic. This independently, by the way, from all the possible existing differences with respect to the actual list of "roles" and their detailed descriptions. Among the first concrete realizations of the notion of "role" according to an Eq (1) meaning, we can recall Silvio Ceccato's "*correlators*" that he used, in the context of some experiments of Mechanical Translation (MT) in the fifties-sixties, to represent general kinds of narratives as recursive networks of triadic structures [4]. Another n-ary, Eq (1) congruent, knowledge representation model used to encode narrative-like structures that was very popular in the seventies is the "*Conceptual Dependency*" theory of Roger Schank [23]. In this, the underlying meaning ("*conceptualization*") of a given narrative/scenario/complex event was expressed as the association of semantic predicates chosen from a set of twelve formal "primitive actions" (like INGEST, MOVE, ATRANS, abstract relationship transfer, PTRANS, physical transfer, etc.) with seven role relationships ("*deep cases*") in the Case Grammar style [6]. The seven roles were Object (in a state), Object (of a change of state), Object (of an action), Actor, Recipient/Donor, From/To and Instrument. Unfortunately, Schank's theory was, on the one hand, insufficiently specified and, on the other, unnecessarily complicated because of the influence of psychological (introspective) considerations according to a characteristic trend of AI in those years. Nevertheless, Schank's work had a particularly strong influence on the development of formalized (and at least partly computerized) systems for the representation and management of storylines and connectivity phenomena making use of all sorts of scripts, scenarios, TAUs (Thematic Abstraction Units), MOPs (Memory Organization Packets) etc.

Conceptual Graphs (CGs) represent a recent [25,26] high-level n-ary modelling paradigm based on a powerful graph-based representation scheme that makes use of two kinds of nodes, "*concepts*" and "*conceptual relations*" (i.e., functional roles). For example, a CG corresponding to the narrative "John is going to Boston by bus" is represented by a conceptual structure where a "concept node", "Go" (having a function similar to that of an NKRL conceptual predicate, but denoted by a natural language term) is associated with three "relation nodes" like Ag(e)nt, Dest(ination) and Instr(ument). They introduce the three arguments of the predicate, i.e., three new concept nodes representing, respectively, the constant John (the "agent") as an instance of the concept Person, the constant Boston (the "destination") as an instance of the concept City and the concept Bus (the "instrument"). The resemblance to the NKRL representation of elementary events is quite evident. Other similarities concern, e.g., the existence, for any CGs system, of a hierarchy of *concept-types* comparable to HClass, the use for the modelling of the *contexts* of second order (nested graphs) extensions that bear some resemblance to NKRL's constructs like completive construction and binding occurrences, the analogies about the structure of some inference techniques, etc. However, essential differences also exist. They concern, in particular, the choice of leaving *completely free* in CGs, for the sake of generality, the selection of those "predicates" that, in CGs as in NKRL, represents the focal element of the representation of an elementary event. In the CGs encoding of the "John is going to Boston..." event, the predicate is then simply represented by the generic surface element "Go" – it would be a primitive like MOVE in NKRL. As a consequence, it becomes extremely difficult to create an exhaustive and authoritative list of CGs "canonical graphs", equivalent to NKRL's "templates", for evident reasons of *combinatorial explosion*. A tool like HTemp is, then, practically inconceivable in a CGs context.

We can conclude this "comparisons" Section by some remarks about the ontological systems currently used in a "cognitive robotics" domain. The IEEE Standard Ontology for Robotics and Automation (R&A) [24] includes a Core Ontology for R&A (CORA) and three additional specialized hierarchies. CORA [22] has been developed to provide unambiguous definitions of core notions of robotics and related topics, such as robot-robot and robot-human communication, robot design, and integration of data about robots. It is based on SUMO, the Suggested Upper Merged Ontology [4], a large top-level ontology of general concepts that "merges" several previous ontological works and aims at defining the main ontological categories describing the world. SUMO has been written in SUO-KIF language, a first-order language that can be assimilated to the frame-based programming languages mentioned above. An extension of NKRL to take into account, in particular, spatial reasoning in a specific robotics/AAL domain is [1]. The majority of the ontology-based applications in the "cognitive robotics" domain are now, however, OWL-based. KnowRob, KNOWledge processing for ROBots [28], is a well-known system that introduces a common vocabulary for representing knowledge about robot actions, events, objects, environments, and the robot's hardware as well as inference procedures able to operate on this common representation. The definition of this common vocabulary is mainly based on the root concepts of the OpenCyc upper ontology, an open source subset rewritten in OWL of the original frame-based Cyc ontology [14]. OpenCyc/OWL has also been used for implementing ORO, the OpenRobots Ontology system [11]. ORO is an event-oriented platform for symbolic knowledge storage and reasoning that focuses on modelling robot perceptions and that has been used in several human-robot interaction and dialogue scenarios. A recent, interesting work that extends KnowRob is described in [3]. Dealing with events, actions and situations, this work has some contact points with NKRL – even if, at the difference of this last language, is strictly limited to deal with a pure "*physical world*". It concerns, in fact, how to represent the "episodic memories" of specific agents that perform *manipulation tasks*, and deals then with perceived objects, performed physical actions, their duration and possible failures. Actions – like, e.g., arm movements – are decomposed in motion phases with different subgoals. The final aim of the work is to show how those memories can be used to improve the robot's action models by getting insights about their manipulation activities.

5 Conclusion

After having recalled some general characteristics of NKRL (like the addition of an "ontology of events" to the usual "ontology of objects, or the possibility of dealing in an efficient way with some sorts of contextual information), we have then focused the discussion on the specific *rules and inferential aspects* proper to this language. We have then introduced the general, formal model of "rule" used in an NKRL context and we have shown how this can be appropriately adapted to the setup of advanced types of inference operations based, e.g., on analogical and causal types of reasoning.

In particular, we have emphasized how the use of the advanced n-ary features of NKRL in a rule/inference context has a very important consequence: we are no more constrained to express the atoms of these rules in terms of *binary clauses* using just monadic (unary) and dyadic (binary) predicates (as usually carried out in an Artificial Intelligence context), and these atoms can directly represent, now, complex and meaningful situations. Note that modelling the rule atoms according to a simple binary format implies, among other things, the impossibility of stating even relatively simple implications of the "if-then" type making use of a reduced number of clauses. This leads in practice to the necessity of the (systematic) decomposition of the original formulation of the rules into (possibly very large) amounts of simple binary clauses, implying serious problems both from a logical (and operational) point of view. It is evident, in fact, that techniques of this type are relatively simple to use when the rules to be built up are quite simple, but they can be *out of place* in more realistic, complex and dynamic situations like those typically dealt with in an NKRL context (see the Section 3 examples). In these cases in fact, very often, the decomposition techniques would be unable to deal with the specific situations using a *limited number* of binary clauses.

References

1. Ayari, N., Chibani, A., Amirat, Y.: Semantic Management of Human-Robot Interaction in Ambient Intelligence Environments Using N-ary Ontologies, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2013), pp. 1172-1179. Piscataway (NJ), IEEEXplore, 2013.
2. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N.: Abstract Meaning Representation (AMR) 1.2 Specification, The Linguistic Data Consortium (LDC), Philadelphia (PA), 2014.
3. Bessler, D., Koralewski, S., Beetz, M.: Knowledge Representation for Cognition and Learning-enabled Robot Manipulation, in Proceedings of the 11th International Workshop on Cognitive Robotics – CogRob 2018 (CEUR vol. 2325), pp. 11-19, CEUR-WS, Aachen, 2018.
4. Ceccato, S.: Automatic Translation of Languages. Information Storage and Retrieval 2(3),1964, 105-158.
5. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering, in Proceedings of the 1997 AAAI Spring Symposium on Ontological Engineering, pp. 33-40, AAAI Press, Menlo Park (CA), 1997.
6. Fillmore, C.J.: The Case for Case, in Bach, E., Harms, R.T. (eds.) Universals in Linguistic Theory, pp. 1-88, Holt, Rinehart and Winston, New York, 1968.
7. Gruber, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies 43(5/6),1995, 907-928.
8. Guarino, N., Welty, C.: Evaluating Ontological Decisions with OntoClean. Communications of the ACM 45(2), 2002, 61-65.
9. Guha, R.V., Brickley, D., Macbet, S.: Schema.org – Evolution of Structured Data on the Web. ACMQueue 13(9), 2015, 1-28.
10. Kifer, M., Lausen, G., Wu, J.: Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM 42(4), 1995, 741-843.
11. Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., Beetz, M.: ORO, a Knowledge Management Module for Cognitive Architectures in Robotics, in Proceedings of

the IEEE/RSJ International Conference on Intelligent Robots (IROS 2010), pp. 3548-3553, Piscataway (NJ), IEEEXplore, 2010.
12. Lenat, D., Guha, R., Pittman, K., Pratt, D., Shepherd, M.: Cyc: Toward Programs With Common Sense. Communications of the ACM 33(8), 1998, 30-49.
13. Manola, F., Miller, E. (eds.): RDF Primer (W3C Recommendation 10 February 2004), http://www.w3.org/TR/rdf-primer/ (accessed 10 June 2019).
14. Matuszek, C., Cabral, J., Witbrock, M.J., DeOliveira, J.: An Introduction to the Syntax and Content of Cyc, in Proceedings of the AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, pp. 44-49. Menlo Park (CA), AAAI Press, 2006.
15. Minsky, M.: A Framework for Representing Knowledge, in Winston, P. (ed.) The Psychology of Computer Vision, pp. 211-277, McGraw-Hill, New York, 1975.
16. Niles, I., Pease, A.: Toward a Standard Upper Ontology, in Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS 2001), pp. 2-9, New York, ACM Press, 2001.
17. Nilsson, N.J.: Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto (CA), 1980.
18. Nirenburg, S., Raskin, V.: Ontological Semantics, MIT Press, Cambridge (MA), 2004.
19. Noy, N., Fergerson, R.W., Musen, M.A.: The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility, in Proceedings of EKAW 2000 (LNCS vol. 1937), pp. 17-32, Springer, Berlin, 2000.
20. Noy, N., Rector, A. (eds.), Hayes, P., Welty, C. (contributors): Defining N-ary Relations on the Semantic Web (W3C Working Group Note 12 April 2006), http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/ (accessed 10 June 2019).
21. Pepper, S.: Topic Maps, in Bates, M.J. (editor-in-chief) Encyclopedia of Library and Information Sciences, 3rd edition, pp. 5247-5260, Taylor & Francis, Abingdon, 2010.
22. Prestes, E., Carbonera, J.L., Rama Fiorini, S., Jorge, V.A.M., Abel, M., Madhavan, R., Locoro, A., Gonçalves, P., Barreto, M.E., Habib, M., Chibani, A., Gérard, S., Amirat, Y., Schlenoff, C.: Towards a Core Ontology for Robotics and Automation. Robotics and Autonomous Systems 61(11), 2013, 1193-1204.
23. Schank, R.C.: Identification of Conceptualizations Underlying Natural Language, in Schank, R.C., Colby, K.M. (eds.) Computer Models of Thought and Language, pp. 187-247, W.H. Freeman, San Francisco, 1973.
24. Schlenoff, C., Prestes, E., Madhavan, R., Gonçalves, P.J.S., Li, H., Balakirsky, S., Kramer, T., Miguelanez, E.: An IEEE Standard Ontology for Robotics and Automation, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots (IROS 2012), pp. 1337-1342, Piscataway (NJ), IEEEXplore, 2012.
25. Sowa, J.F.: Conceptual Structures – Information Processing in Mind and Machine, Addison-Wesley, Reading (MA), 1984.
26. Sowa, J.F.: Knowledge Representation – Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing, Pacific Grove (CA), 1999.
27. Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D.: OntoEdit: Collaborative Ontology Development for the Semantic Web, in The Semantic Web – Proceedings of ISWC 2002 (LNCS vol 2342), pp. 221-235, Springer, Berlin, 2002.
28. Tenorth, M., Beetz, M.: KnowRob, A Knowledge Processing Infrastructure for Cognition-enabled Robots. International Journal of Robotics Research 32(5), 2013, 566-590.

29. Trame, J., Kessler, C., Kuhn, W.: Linked Data and Time – Modeling Researcher Life Lines by Events, in Proceedings of COSIT 2013 (LNCS vol. 8116), pp. 205-223, Springer, Berlin, 2013.
30. Woods, W.A.: What's in a Link: Foundations for Semantic Networks, in Bobrow, D.G., Collins, A. (eds.) Representation and Understanding – Studies in Cognitive Sciences, pp. 35-82, Academic Press, New York, 1975.
31. Zarri, G.P.: Integrating the Two Main Inference Modes of NKRL, Transformations and Hypotheses, in Spaccapietra, S. (ed.) Journal on Data Semantics IV (LNCS vol. 3730), pp. 304-340, Springer, Berlin, 2005.
32. Zarri, G.P.: Representation and Management of Narrative Information – Theoretical Principles and Implementation, Springer, London, 2009.
33. Zarri, G.P.: Knowledge Representation and Inference Techniques to Improve the Management of Gas and Oil Facilities. Knowledge-Based Systems 24(7), 2011, 989-1003.
34. Zarri, G.P.: Conceptual and Content-Based Annotation of (Multimedia) Documents, Multimedia Tools and Applications 72(3), 2014, 2359-2391.
35. Zarri, G.P.: Using the Formal Representations of 'Elementary Events' to Set up Computational Models of Full 'Narratives', in Hai-Jew, S. (ed.) Data Analytics in Digital Humanities, pp. 39–64, Springer, Berlin, 2017.
36. Zarri, G.P.: Functional and Semantic Roles in a High-Level Knowledge Representation Language. Artificial Intelligence Review 51(4), 2019, 537-575.