The Concept of Reference Objects in the Representation of Natural Language Information in L-LILOG

Claus-Rainer Rollinger
Rudi Studer
Ipke Wachsmuth

IBM Deutschland GmbH
Science and Technology - LILOG
Schloßstr. 70
D-7000 Stuttgart 1, W. Germany

VNET: ROLLING, SHSTUD, WACHSMUT at STUTVM1

Abstract

We present a brief description of the knowledge representation language L-LILOG. In particular, the concept of reference objects is introduced as a central means to build semantic representations for textual information. We describe how to construct internal representations as directed acyclic graphs and how to realize operations on reference objects to be performed in the process of constructing and utilizing semantic representations for natural language texts.

The Knowledge Representation Language L-LILOG

The main objective of the project LILOG (Linguistic and Logic Methods) is to develop concepts and methods for understanding German texts and dialogs. For representing the different types of knowledge needed within a natural language understanding system, a knowledge representation language, L-LILOG has been designed which is currently being implemented. L-LILOG is based on a many sorted first-order predicate calculus and includes structuring principles for organizing large knowledge bases. On the short run these efforts are not directly concerned with expert system applications. As an explicit long-range objective, however, the concepts and methods developed shall serve as a basis for knowledge acquisition in the area of advanced expert systems.

The knowledge representation language L-LILOG has been designed to fulfill two general requirements:

- The language should provide a framework for representing different kinds of domain knowledge which have to be handled in a natural language understanding system. I.e. concepts have to be offered for representing domain specific as well as domain independent real world knowledge which is typically incomplete, vague and/or uncertain. According to specific topics investigated in the LILOG project, the representation of temporal and spatial knowledge is of particular importance.

- The language should be based on a sound theoretical basis which provides means (i) for formally specifying the semantics of the language constructs as well as (ii) for formally defining the semantics of the inference processes.

Further, general design requirements for knowledge representation languages (Steels 1984) and several knowledge representation formalisms, especially KRYPTON (Brachman et al. 1983), SRL (Habel 1986), Conceptual Graphs (Sowa 1984), and Discourse Representation Theory (Guenthner et al. 1986), were screened and analyzed. Based on this analysis it was decided to design the core of L-LILOG according to the following principles:

1. L-LILOG is based on a many-sorted predicate calculus.

2. Sorts in L-LILOG are partially order-sorted.

3. Knowledge packets offer means for structuring knowledge bases in a hierarchical way.

4. Within the sort hierarchy an inheritance mechanism for attributes is defined.

5. Reference objects are used to represent information available about existing world entities in an object-centered way.

6. Attributes are used for specifying relationships between components of sort specifications.

7. A role-value notation for specifying arguments is offered to support variable arities of predicates.

8. L-LILOG includes different types of notational formalisms among which well-defined transformation rules are given.

Subsequently, we describe some of these principles in more detail. Basically, a L-LILOG knowledge base is defined as follows:

Knowledge-Base ::= Sort-Declaration
Reference-Object-Declaration
Knowledge-Packet-Structure
Knowledge-Elements

The 'Sort-Declaration' component represents the main part of the LILOG concept lexicon. And contains a specification for each concept known to the system. It consists of a set of sort descriptions each specifying the name of the sort ('sort-name'), its supersort(s), the associated knowledge packets (possibly several), and the sort specification. The partial ordering of sorts has to be a semi-lattice and is intepreted as a set-subset relationship in the models of L-LILOG. It was decided to use a sort concept within the predicate calculus approach in order to be able to represent and manipulate taxonomic knowledge more appropriately, e.g., by replacing general deduction processes by more specialized processes like type checking (Aït-Kaci 1984). This information is used to define assertions which are included in the 'Knowledge-Elements' component.

Attributes are specified by their name, optional coreference markers for defining an equality relation between attributes, and a domain specification. Actually, an attribute is interpreted as a partial function from the sort for which it is defined to the sort specified by its domain specification. Sort specifications may be nested to any level.

The 'Knowledge-Elements' component is comprised by a set of first-order formulas representing facts and rules which express propositions about real world entities. Roughly speaking, the 'Sort-Declaration' component and the 'Knowledge-Elements' component correspond to the T-Box and A-Box components of representations in KRYPTON-like formalisms (Brachman et al. 1983).

The 'Knowledge-Packet-Structure' defines a hierarchy of knowledge packets (Wachsmuth 1987) organizing conceptual as well as assertional knowledge in a modular way. To this end, all elements defined in the different knowledge base components are associated with one or several knowledge packet(s) to define the context in which these elements are available to the knowledge processing component.

The 'Reference-Object-Declaration' component introduces internal identifiers for the real world entities mentioned in a natural language text. Each identifier is associated with a sort defined in the 'Sort-Declaration' component as well as with a set of designations which have been used in the text to refer to a particular entity. The 'Reference-Object-Declaration' component is based on the referential net concepts described in (Habel 1986). This construct is particulary well-suited to handle hard-to-deal-with phenomena of reference characteristic for natural language. To some extent, such a construct is also provided by Discourse Representation Theory (Guenthner et al. 1986). In the remainder of the paper we highlight how reference objects are employed in L-LILOG constructing and using semantic representations for natural language texts.

The Concept of Reference Objects

Reference objects are introduced as unique internal representatives for external objects of a domain to account for the fact that natural language provides various means to designate a given real world object. The current version of L-LILOG accounts for multiple designations of objects by proper nouns. So far, it does not give consideration to any other definite or any indefinite descriptions of objects. The syntax is given below.

reference-object-declaration ::= "REFERENCE OBJECTS"
reference-object +

reference-object ::= reference-object-id "e" sort-name
"DESIGNATED-BY <" [designation +] ">"

reference-object-id ::= "r"digit-string

designation ::= object-name
[knowledge-packet-assignment]

These definitions introduce two relations defined between reference objects and other constructs of L-LILOG. The first one is the element relation "e" between reference objects and sorts. By this relation we can specify the sort that a reference object belongs to. The second relation is the "DESIGNATED-BY" relation between reference objects and object names. Object names are proper nouns used in the external world to designate the real object. Instances of this relation enables the system to find out which internal object is referred to when a certain proper noun is used in a discourse to designate an external object. Object names are also needed when we want to generate a natural language sentence that speaks about a real world object.

Below we describe how attributes of reference objects are represented internally as directed acyclic graphs (DAGs) and discuss the operations for constructing such internal representations and for extracting information from them. Internal representations are based on the Stuttgart Type Unification Formailsm, STUF, described elsewhere (Uszkoreit 1981). For each particular piece of knowledge to be expressed in L-LILOG with its particular syntactic constructions we construct an internal STUF representation along with operations suited to process this knowledge. For reasons of efficiency and economy these internal representations are process-oriented.

The declaration of a reference object includes (1) an instantiation of the element relation which relates a particular reference object identifier (e.g., r1) to a particular sort name, and (2) an (optional) instantiation of the DESIGNATED-BY relation between r1 and a set of strings (e.g., proper nouns designating this reference object). We handle each case separately.

For each sort Si we define a graph with Si as its name. We take the reference object identifiers of the reference objects that are elements of that sort and associate them as labels with those edges that start at the top node of Si. That way each reference object explicitly belongs to exactly one sort Si and it implicitly belongs to all supersorts of Si. This part of our knowledge base is not static: As more information about a reference object is obtained (e.g., that it has a certain attribute) it can be moved down the sort lattice to a subsort. That is, we have to replace "ri e Sj" by "ri e Sk", given that "subsort(Sk,Sj)" is true. To do this, the edge <ri> is removed from the graph Sj and attached to the graph Sk by graph unification. All reference objects of a sort, including the ones attached to its subsorts can be obtained by unifying graph Si with all graphs of its subsorts.

For each reference object we can define a set of strings designating it. These designations do not have to be definite, that is, one string may designate more than one reference object. For each reference object a graph is defined with the reference object identifier as its name. The designations are attached as names to all edges of that graph that emanate from the top node.

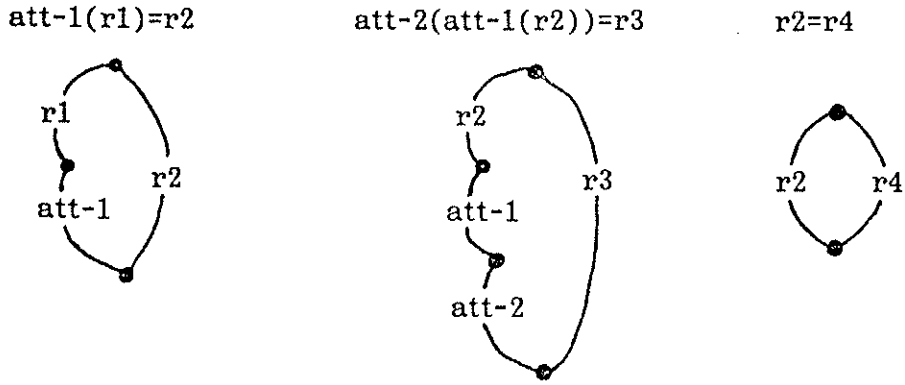
Directed graphs differ from trees in that they can encode equality. Equality is represented as reentrancy; a shared value is pointed at by all attributes that share this value. In this context we interpret reentrancy as the equality of paths in a graph that start at the top node and join each other at the first node they have in common.

Attributes of reference objects are defined as one-place functions. To instantiate the attribute of a reference

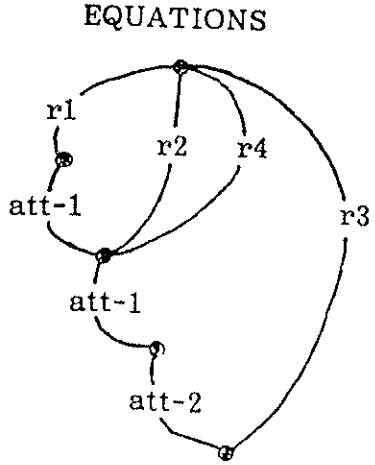
object means to write expressions of the special kind "term = term". Here, a term can only be a reference object identifier or a one-place function. Specifying the known attributes of the reference objects (either used in the the background knowledge or in the discourse knowledge) means to formulate a special system of equations. This system need not be complete. That means, we need not know all attributes of an object, there may be reference objects for which only the sorts they belong to are known. We are interested in the equality of the attribute values but not in the equality of the attributes as functions. Reference objects of a certain sort may obtain attributes that are not defined for its sort. These additional attributes are required to be attributes of a subsort so we can be sure that the objects with this attribute are elements of that subsort. At the current stage we do not allow to define "free" attributes that are not inherited since we do not yet allow the blockage of inheritance.

The system of equalities is to satisfy the following consistency conditions: (1) a reference object cannot be used as its own attribute value and (2) if a reference object ri is the attribute value of some other reference object rj, then rj cannot be the value of any attribute of ri. In case the second restriction turns out too strong it can be weakened by introducing inverse functions for those functions which are injective. In the internal representation we have chosen for this system of equalities the consistency conditions are easily verified since their violation yields cycles in the resulting graph which has to be acyclic by definition.

For each atomic formula we construct as internal representation a graph with the two reference object identifiers as labels of the two edges emanating from the top node. These are followed by edges labeled with those attribute names the reference object identifiers are imbedded in. The resulting two paths corefer to the bottom node of the graph. Then all such graphs are unified. The resulting graph will contain for each reference object, if at least one of its attributes is known, exactly one edge with its reference object identifier as label. In this way all equations of interest are made explicit. The unification of two graphs fails if a cycle is produced as defined above.



The unification of these graphs results in the graph EQUATIONS:



Now we can prove equalities such as att-2(att-1(att-1(r1))) = r3. To do this, we first construct the corresponding graph Gi. Then we test whether or not this graph subsumes the graph EQUATIONS. If "subsumes(EQUATIONS,Gi)" is true we have proved that the equation holds. This is the most important operation on the graph Gi. Other operations are indicated below:

- Which term is the value of an attribute att-i of an object rj?
  get(att-i(rj),x)
- Which reference object has ri as value of an attribute att-j?
  get(att-j(x),ri)
- What are the attributes of the reference object ri?
  get(x(ri),y)
- What is the relation between ri and rj?
  get-rel(ri,rj,z), z is a graph
- Which terms embedded in att-i have which value?
  get(att-i(x),y)
- Add an equation!
  unify(EQUATIONS,Gi) AND store the resulting graph

The above constructions constitute part of the inventory to deal with more complex tasks such as evaluating if two terms (in the restricted sense used here) can be equal. As a more specific task the question, whether or not two reference objects can be equal, can be dealt with in this framework.

Conclusions and future perspectives

In this paper, we have sketched the basic design features of a representation system that integrates different kinds of knowledge. A language that is based on a many-sorted predicate calculus has been equipped with the additional descriptive power of a graph-unification-based sort hierarchy with equality. The integration of the two systems provides a high degree of conceptual clarity and supports a modular implementation in which the underlying data type and its operations are shared by all components of the system. Work is well underway to use L-LILOG for representing temporal and spatial information which is prominent in many natural language texts. Appropiate ontological categories have been included in the sort lattice and have proved successful in first applications. Topics to be addressed next include extending the expressive power of L-LILOG to provide means for handling vagueness and uncertainty.

References

Aït-Kaci, H. (1984). A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures. Ph.D. Thesis, University of Pennsylvania.

Brachman, R.J., Fikes, R.E., & Levesque, H.J. (1983). KRYPTON: Integrating terminology and assertion. Proceedings AAAI-83 (pp.31-35).

Guenthner et al (1986). A theory for the representation of knowledge. In: IBM Journal of Research and Development, Vol.30, No.1, 1986, pp 39-56.

Habel, Ch. (1986). Prinzipien der Referentialität: Untersuchungen zur propositionalen Repräsentation von Wissen. Berlin: Springer.

Sowa, J. (1984). Conceptual Structures: Information Processing in Mind and Machine. Reading, Mass.: Addison-Wesley.

Uszkoreit, H. (1987). STUF: A Description of the Stuttgart Type Unification Formalism (LILOG-Report 16). Stuttgart: IBM Deutschland.

Steels, L. (1984). Design Requirement for Knowledge Representation Systems. In: Proc. GWAI-84, pp. 1-19.

Wachsmuth, I. (1987). On structuring domain-specific knowledge (LILOG-Report 12). Stuttgart: IBM Deutschland.