Representing Reified Relations in Loom

Robert M. MacGregor

USC/Information Sciences Institute

4676 Admiralty Way

Marina del Rey, CA 90292

(310) 822-511

macgregor@isi.edu

Abstract

This paper discusses the semantics and usage of reification as applied to relations and tuples. The reification of a tuple is a proposition object possessing a case role for each domain attribute in the tuple. The reification of a set of fillers of a role is an object sometimes referred to as a "roleset". In the course of defining reification mechanisms for the Loom knowledge representation system, we have unearthed several open issues that come into focus when considering equivalence relations between these kinds of reified objects. Another type of reification produces an individual that represents a view of another individual filling a particular role. We present a number of semantic variations of this reification operation, and argue that the unbridled application of such reification operators has the potential to overwhelm the representation mechanism. We suggest that a regimen that merges various similar but non-equivalent classes of individuals might be preferable to a system that insists on unique representations for each possible abstraction of an individual.

1 Introduction

This paper discusses semantic issues that have surfaced with regard to the representation of propositions in the Loom knowledge representation system [MacGregor 91a, MacGregor 91b]. Loom is a terminologically-based KR system (a member of the KL-ONE family). Although KL-ONE embodied some very direct assumptions about how propositional knowledge should be treated [Brachman & Schmolze 95], the terminological logic community has paid relatively little attention to propositional representations in more recent times. As we continue to increase Loom's reasoning abilities we are finding it necessary to take positions on these issues. The primary topics we discuss here are the treatment of reified tuples and rolesets and the semantics of qua-induced classes and individuals.

Our running example contains concepts/classes named `Person` and `Company`, a binary relation `companies` with domain `Person` and range `Company`, and its inverse `workers`. Also, we have some labelled propositions about `Fred`:

```
P1:  companies(Fred, IBM)
P2:  companies(Fred, Apple)
P3:  about(Fred, (exactly 2 companies)),
```

i.e., `Fred` works for `IBM`, `Fred` works for `Apple`, and `Fred` works for exactly two different companies.

By convention, which may or may not be standard, we equate the proposition labelled `P1` with the reification of the tuple `<Fred,IBM>` belonging to the `companies` relation. Statements about a tuple (e.g., "Joe knows that Fred works for IBM"), are represented by making statements about its reification (e.g. `knows(Joe, P1)`). Many KR systems (especially, the more expressive ones) reify each binary proposition entered into their knowledge bases. This provides a direct means for representing certain cases of nested propositions (propositions about propositions).

2 Reification of Binary Relations

With respect to the problem of representing reified objects, the Loom project has thus far focused primarily on the semantics of reification for binary relations. Our work in this area has been motivated by the needs of natural language (NL) processing applications. The intent of the discussion below is to provide a feel for the scope of the problem, rather than to arrive at a definitive solution.

Currently, Loom does not provide a representation for reified binary tuples. This architectural decision significantly reduces the space requirements of Loom knowledge bases, but sacrifices representational power. A straightforward reification mechanism would enable users to represent (non-modal) statements about propositions. For example, if one wishes to state that `Fred` is working for `IBM` only on a temporary basis, we might represent this by (i) reifying the <`Fred, IBM`> tuple, producing what might be called a works-for "event", and then (ii) attaching the property `temporary` to that event. We plan to upgrade the system to support what might be termed *lazy* reification: The default implementation of binary tuples will be the non-reified representation. The first attempt to attach a property or role value to a proposition that corresponds to a non-reified binary tuple will trigger the creation of a new object representing the reification of that tuple. Reifying the tuple results in a very slight overhead for future accesses to the non-reified representation, since the non-reified representation is henceforth interpreted as a derivative computation rather than as just a slot access.

Loom also does not provide a means for reifying sets of role fillers, but in this case that does not mean that Loom cannot reason about such objects: Let `S1` be the reification of the set of fillers of the `companies` role attached to `Fred` (`S1` has members `IBM` and `Apple`). Proposition `P3` can be restated as

```
cardinality(S1, 2).
```

The reified object `S1` is sometimes referred to as a *role set* (KL-ONE uses this term). Terminologically-oriented description logics such as those found in Loom, CLASSIC [Borgida 92] and BACK are unusual in that they allow one to state propositions about role set objects without explicitly representing (reifying) those objects. In Loom, one can formulate descriptions that refer to the upper and lower bounds on the cardinality of a role set, the type of the fillers of a role set, and specific fillers or non- fillers of a role set. Our proposition `P3` above asserts that the role set "companies of `Fred`" has cardinality 2 (i.e., the upper bound is 2 and the lower bound is 2).

Some KR systems explicitly attach "role" or "slot" objects to classes, where they function roughly in the manner we have attributed to role sets. A question of identity arises when we refer to such "roles" as being inherited from a class to a subclass. For example, if `Employee` is a subclass of `Person`, and if `age` is a role both of `Person` and of `Employee`, is it the same role? We claim not. For example, we can attach the restriction "`age` is at least 18" to the role "`age` of `Employee`" without attaching it to the role "`age` of `Person`".

Loom users would like a capability to retrieve the roles of a class, and to inquire about the restrictions attached to a specific role. Because roles do not exist as objects of discourse in Loom, Loom does not provide this capability (although it does provide hooks for retrieving equivalent information). Our justification is again motivated by a desire to conserve space—Loom class hierarchies can be relatively deep, and a role attached to a class at the top of a hierarchy will have a distinct identity at each of the subclasses. Rather than creating role objects for each subclass that inherits a role, we plan to devise a strategy wherein role objects would be materialized only on demand (e.g., when needed by a class browser), and garbage collected when the demand ceases. Again, we witness a theme that consistency motivates our architecture: As knowledge bases increase in size, the need to economize on space will grow in importance. Reification (which makes explicit what was previously implicit) is an operation that consumes space. Strategies for reifying only on demand, or better yet, only while a demand exists, may become essential as we address increasingly large-scale applications.

3 Equivalence Relations for Reified Tuples

Consider again the hard-working Fred. Although the propositions

```
P4:  companies(Fred, IBM)
```

and

```
P5:  workers(IBM, Fred)
```

correspond to distinct tuples, the Loom implementation does not distinguish between a binary tuple and its inverse. As a consequence, Loom treats `P4` and `P5` as equivalent propositions. We suggest that this consequence has merit independent of Loom's representational choice for tuples. Let us refer to the assumption that "the reification of a binary tuple and the reification of the inverse tuple are the same object" as the "Inverse-Equivalent Assumption" (IEA). The IEA means, for example, that the reification of <`Fred, IBM`> in the relation `companies` equals the reification of <`IBM, Fred`> in the relation `workers`. At the linguistic level, IEA is almost surely undesirable, e.g., the sentence "Fred works for IBM" must be distinguished from the sentence "IBM employs Fred." Below, we argue the benefits of IEA at the semantic level.

By equating `P4` and `P5`, we not only save space (by a factor of two over all reifications of roles having inverses), we also obviate the need to propagate assertions made about one proposition to another equivalent proposition, e.g., generating an axiom such as

```
probability(P4,0.8) implies probability(P5,0.8)
```

becomes unnecessary.

The consequences of IEA percolate up to the meta level. At the meta level, conceptual objects representing classes of tuples (i.e., relations) are distinct from conceptual objects representing classes of reified tuples. To provide support for NL applications such as PENMAN [Hovy 90], Loom implements a construct called `defreified-relation` which, upon invocation, defines a pair of meta objects, one denoting a binary relation, and one denoting the corresponding class of reified tuples. Let us name the class of reified `companies` tuples `Workers/companies` and name the class of reified `workers` tuples `Companies/workers`. IEA implies that these two just-named classes have identical extensions. We propose that they should be intensionally equivalent as well, i.e., that they denote the same class. We plan to modify the current `defreified-relation` to account for this semantics in a future version of Loom.

Digression: How do we plan to support distinct representations of linguistic knowledge while merging them at the semantic level? We don't! While many natural language researchers advocate making the representational power of a KR system sufficiently complex as to enable the representation of arbitrary linguistic phenomena, we consider this to be a mistake. Instead, we advocate drawing a sharp boundary between a linguistic representation and its semantic interpretation(s). A single linguistic expression can correspond to several different semantic representations. We consider it the job of the NL processor, not the KR system, to define these mappings and to postulate which mappings are correct in the context of the linguistic expression.

The notion that a reified binary tuple and its reified inverse are equivalent extends as well to higher arity relations. Consider the ternary relation `gives`, with domains labelled donor, object, and recipient. In many KR systems, the preferred representational style is to reason about the reification of `gives`, which we will call `Giving`. `Giving` has three unordered case roles (donor, object, and recipient). The inherent symmetry of these case roles suggests that instances of `Giving` correspond equally well to tuples in relations that are permutations of `gives`, e.g., `receives` (recipient, object, donor). Our interpretation of this common usage of reified relations is that the reified relation maps to all permutations of the corresponding non-reified relations, e.g., `Giving` maps to six different non-reified relations. If we reflect this style of representation back to the binary case, we might refer to the reification of `companies` as `Working-for` (with case roles `workers` and `companies`), and we would consider `Working-for` to be the reification of both `companies` and `workers`. In other words, the IEA assumption (which applies only to binary relations) corresponds to a commonly adopted semantics for reification of higher arity relations (those with arity greater than two). The "Permutation-Equivalence Assumption" (PEA) generalizes IEA to apply to relations of arbitrary arity.

4 The Qua Relation

We next consider the *qua* (role-as-concept) relationship [Freeman 81]. Let us define the class `Employee`, to be "a person who works for a company", (i.e., `Employee` is qua-induced by the relation `companies`). `Fred` as an instance of `Person` corresponds to two distinct individuals in the class `Employee`: `Fred-as-IBM-worker` and `Fred-as-Apple-worker`. The usual assumption that the descriptions "`Fred` as `Person`" and "`Fred` as `Employee`" denote the same individual is not tenable—the two `Employee` individuals will likely have distinct employee numbers, distinct managers, etc. On the other hand, they inherit attributes from `Fred-as-Person` in ordinary IS-A link fashion. Hence, while we find it convenient to assume that the meta level proposition `IS-A(Employee,Person)` holds, the relationship between `Person` and `Employee` is clearly not a class/subclass relation. We consider the epistemological status of qua-induced classes vis-a-vis how they relate to class/subclass inheritance hierarchies to be an open issue for the KR community. We plan to add to Loom a limited form of qua semantics—the *view* relation will, for example, relate a single `Person` view to multiple `Employee` views.

The introduction of qua opens up new vistas for generating reified individuals. For `Fred`, we can immediately count the following: Two propositions denoting the reification of the tuples <`Fred, IBM`> and <`Fred, Apple`>; two `Employee` individuals; and the reifications of the "virtual" tuples <`Fred-as-IBM-worker, IBM`> and <`Fred-as-Apple-worker, Apple`>. Then, of course, there are individuals denoting the set of `companies` that `Fred` works for, the set of `companies` that `Fred-as-IBM-worker` works for, and on and on. Noting that the total information content of this edifice is wholly contained in two binary tuples, we suggest that there is a definite need for some sort of population control mechanism.

5 Conclusions

We conclude with a proposal and an observation. Our proposal is that the reification of a binary tuple and its inverse should be the same individual (IEA, or better yet, PEA). Our observation is that a strategy of eagerly reifying binary tuples may prove to be increasingly untenable as KR systems come to implement increasingly sophisticated role-related semantics. As the distinctions among the proliferating classes of reified individuals become increasingly abstract, we may reach a point where a decision to merge various similar but non-equivalent classes of reified individuals might turn out to be preferable to requiring humans and/or machines to distinguish between the large number of semantic possibilities. IEA/PEA and lazy reification each provide a means for reducing to some degree the semantic complexity of a knowledge base.

References

[**Borgida 92**] Borgida, Alex, "From Types to Knowledge Representation: Natural Semantics Specifications for Description Logics", *Intern. Jrnal on Cooperative and Intelligent*

Information Systems **1**:1, 1992.

[**Brachman & Schmolze 85**] Brachman, R.J. and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science*, 171-216, 1985.

[**Freeman 81**] Freeman, Michael, "The QUA link", in J.G. Schmolze and R.J. Brachman (eds.), *Proceedings of the 1981 KL-ONE Workshop*, 55-65, 1981.

[**Hovy 90**] Hovy, E.H., "Natural Language Processing at ISI", *Finite String* **16**:4, 37-42, 1990.

[**MacGregor 91a**] MacGregor, Robert, "The Evolving Technology of Classification-based Knowledge Representation Systems," in John Sowas, (ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, 385-400, Morgan-Kaufman, 1991.

[**MacGregor 91b**] MacGregor, Robert, "Using a Description Classifier to Enhance Deductive Inference," *Proceedings Seventh IEEE Conference on AI Applications*, Miami, Florida, 141-147, 1991.