On Inheritance In Knowledge Representation

Mark S. Fox
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvannia 15213

Abstract and Introduction

This paper examines the problem of inheritance in Knowledge representation. Research in the formellteton of Knowledge hes resulted in a small number of Knowledge classes and associated inheritance relations, e.g., INSTANCE IS-A, DBROTHERC, PERSPtCTIVE, Virtual-Copy, etc. (Brachman, 1977; Fahlman, 1977; Hayes, 1977; Levesque & Mylopoloue, 1978). The process of inheritance is defined by the procedures that access these inheritance relationa. This paper proposes that: 1) in some cases inheritance between concepts is *idiosyncratic* and does not fit predefined inheritance relations, 2) learning and discovery systems require information on how and why one concept wes *derived* from another, which again Is not represented in standard inheritance relations, and 3) current methods of specifying inheritance modification and similarity mappings are complex to specify and understand. Consequently, e decleretive approach to inheritance and similarity specificaton is presented as a solution to the above problems.

1. Specifying Idiosyncratic Inheritance

1 Experience with representing large varieties of Knowledge show that a small fraction (<10Z) escape standard representation schemes, requiring specialized "fixes" (Fahlman, 1979). The idiosyncratic nature of language and Knowledge precludes its complete structuring using a small set of classes and associated processes. We conjecture that a small set of inheritance types will not suffice. In a some cases the inheritance relation will have to be specialized to the particular concepts they relate. Hence, the inheritance link is context sensitive. Tailoring inheritance to its context requires the explication of exactly what \± *to* be transferred, excluded, added, and/or modified.

Current approaches to handling idiosyncratic inheritance rely on property classification to distinguish between properties to be inherited and those not to be inherited (e.g., structure vs assertion properties, set vs type properties). For example, a structural *properly* is inherited *among classes while assertions are not. But anomalous sub-classes may occur which do not inherit all inheritable attributes (classification is fuzzy at best). To handle anomalous sub-classes, *artificial sub-classes* are inserted between the original super-class and sub-classes. Appropriate attributes are moved from the super-class to the artificial classes. This phenomenon is called *anomaly induced class-splitting.* Typically, class-splitting is a bifurcation where one branch is a singleton set containing the anomaly. Class-splitting increases the size[2] and complexity of the representation thus increasing search time, obfuscating possible relationships among concepts, and negating the storage and description benefits of identifying concepts with class descriptions. It seems that Knowledge classification is an art. which tries to reduce anomaly induced class-splitting.

Secondly, a Knowledge representation must represent arbitrary mappings between concepts. For example a man is like a pig if you map nose onto snout and home onto sty.

To reduce the complexity of representing idiosyncratic concepts and their inheritance relations and to allow more expressive power in describing the similarity relationships between concepts, declarative, idiosyncratic inheritance relations are introduced. Current approaches to specifyng inheritance is implicit in the representation end explicit in the procedures that manipulate the representation. Our goal is to move the explication of inheritance from the procedure to *an* inheritance relation. This enables the context-sensitive specification of inheritance modification. Thus removing the need for class-splitting and any other complexity increasing methods. Secondly, the concept of inheritance is expanded to encompass similarity mappings between concepts (e.g., analogical relations). In the following, we focus not on one particular representation but attempt to describe the mechanism in a representation independent fashion.

We propose a single unidirectional INHERITANCE (INH) relation between two concepts (A — INH —> B) with an attached INHERITANCE CONCEPT (C). The inheritance concept C explicitly states what set of information is inherited, what is excluded, what is created, and what is modified. The inheritance concept can be viewed as a label on the inheritance link specifying a set of transformations. To allow specifications of this type, a language for manipulating representations must be created. While trying not to be pinned down to a single representation, we propose the following primitives:

*This research was aponaorod by tha Defense Advancad Roaoarch Projects Agency (DOD), Arpe Ordor No 3597, monitored by tha Air Force Avionics Laboratory Contract 539815-71.0-1551. In addition, the author it supported In part by a National Research Council of Canada Post graduate Scholarship.

[2]Worst COM Ia 2" closes for N attribute i. e,, a discrimation not.

1. **PASS**: Information passed from A to B.
2. **ADD**: New information added to B.
3. **EXCLUDE**: Information in A not passed to B.
4. **SUBSTITUTE**: Information in A replaced by new information in B.[3]
   - **RESTRICT**: Substitution results in a restriction of possible values.
   - **CONTRADICT**: Substitution results in a contradiction in semantics.
   - **GENERALIZE**: Substitution results in an expansion of possible values.
   - **MAP**: Substitution results in an analogical replacement of information.
   - **REFINE**: Replace with more detailed description.

These primitives are applied to any slot, description, link, node, and other structured primitives of a representation. They specify modifications of the physical structure of a concept to create a new concept. These primitives admit the description of arbitrary transformations among concepts. They are additions to existing representations and are to be interpreted as specifying modifications using the primitives of the particular representation language.

For example, role restriction would have an INHERITANCE link specifying all of the structure inherited using the PASS primitive, and the node being restricted by using the RESTRICT primitive. Differentiation, that is, the addition of SLOTS would use the ADD primitive. Analogical inheritance, would require the PASSing of some information, EXCLUSION of other information, ADDition of new information, and the SUBSTITUTION of information via a MAP (as found in B-structures (Moore & Newell, 1972)). Just what is the semantics of the information PASSED, ADDED, MAPPED, etc. depends on the underlying representation.

It is obvious from the current specification of the inheritance concept that a great deal of information would have to be specified by the PASS primitive. In almost all cases the PASS primitive will be the primary primitive used. The more information to PASS, i.e., the more complex the concept, the more cumbersome it is to write the inheritance concept. To alleviate this problem, we re-introduce the notion of information classes, which is the basis of current representations. The inheritance primitives would then specify that a class of representation structures, e.g., assertions, structures, etc., is to be PASSED, EXCLUDED, etc. But the inheritance concept can still refer to particular structures (node or link). Extending the classification concept to its logical conclusion, we can associate a type with the inheritance concept. For example, IS-A, DSUPERC, INSTANCE/OF, etc. can all be inheritance concept types whose inheritance definition correspond to their interpretation in other representations. But the typed inheritance concept may also specify exceptions to the type's inheritance definition. That Is, an inheritance relation could have an inheritance concept of type "is-a" (which has a standard definition composed of inheritance primitives) but is modified in the particular context by additional inheritance primitives. Since inheritance *types* are defined using inheritance primitives, new types can be defined for commonly occuring relations.

To illustrate these ideas an example is taken from zoology. Example 1 depicts a simplified representation language. A concept is divided into three parts: 1) the VIEW which specifies what the concept is related to. Each slot in the VIEW is a different inheritance concept, 2) the META-CORPUS which specifies wholistic (set, type) information, and 3) the CORPUS which specifies structure information. Example 2 represents a partial description of a *mammal (a denotes a concept). To add *platypus to the set of •mammals requires concept-splitting (the platypus is an exception to the mammal specification, It lays eggs): create a*onotreme with «egg-laying value for *birth-process (ex. 4), and another concept with *live-birth value for ebirth-process. The alternative approach taken in this *paper* results in ex. 5. The *platypus has an inheritance concept of type *!S-A, but the definition of *is-A is overidden by the CONTRADICT primitive specifying that the aplatypus lays eggs instead of live birth. The REFINE primitive is used to replace the *head slot with a *bill and askull slot.

2. Specifying Derivative Relations

A second motivation for describing the relationship between two concepts explicitly is to allow learning and discovery systems to analyse how concepts *are* derived from other concepts.

The goal of learning and discovery systems is to generate new concepts via specialization, generalization , or analogy. In particular, these systems *search* for derivations that are "interesting", where interesting is defined by some heuristic metric. To properly focus search, the method for deriving one concept from another must be recorded. This information is used to analyse how a concept was derived and what should be done to derive a different but related concept. In Lenat*s AM system (1976), a set of heuristics were used to decide how to alter (extend) existing concepts to derive new *and* interesting concepts. A similar approach is used by Fox (1978) to decide how to specialize concepts to create new and interesting concept hierarchies. In Winston's system (1978), transfer frames are hypotheses for what slots to transfer between concepts; heuristics are used for deciding candidate slots. In each system, there exists a set of actions whose application defines a space of new concepts. The action(s) chosen and reason(s) for the choice(s) are important pieces of information used by these systems in deciding, how to extend concepts. The INHERITANCE concept should store it.

We further define each of the INHERITANCE primitives as having the following three attibutes: 1) SET 2) VALUE 3) REASONS. The SET attribute specifies the information the primitive could act on. That is PASS, ADD, EXCLUDE, MAP, RESTRICT, GENERALIZE, or REFINE. The VALUE specifies the actual information chosen from the SET. REASONS specify what decisions led to the choice. Of the three attributes, REASONS is the least defined as it is dependent on both representation *and* Inference mechanisms. If a PASS was to be specified, then the SET attribute of the PASS would

[3] It should bo noted that substitute is the combination of exclude ond add. Wo Include it ■■ ■ primitive because separeting It into oxcludo ond odd would lose the information that there is a contingency, that one structure replace another

list all the structures, e.g., DATTRS the PASS could be applied to. The VALUE attribute would denote the actual structure chosen, the REASON attribute would "explain" why the value was chosen from the SET.

As pointed out earlier, the SUBSTITUTE primitive is equivalent to an EXCLUDE and ADD. Hence, it has two sets of attributes. (SET_e, $VALUE_e$, $REASON_e$) describe the information to be replaced, and (SET_a, $VALUE_a$, $REASON_a$) describes the information actually substituted.

By specifying each of those attributes for each of the inheritance primitives, it is hoped that more information will be made available for learning and discovery systems to make decisions intelligently. By no means are these attributes complete. Their purpose is to focus attention on the types of information needed in inheritance specifications.

Example 6 illustrates how the SET, VALUE and REASON primitives are specified in ADD. In this example, a •platypus is specialized as a *purple-platypus by choosing a color out of the set of possible colors.

3. Conclusion

Inheritance is the primary, most powerful representation primitive available in Knowledge representations. The explication of representation semantics has led to the creation of many types of inheritance links. The semantics of these inheritance types are defined by the procedures that manipulate the representation. Two problems arise in classifying inheritance relations. First, some inheritance relations are idiosyncratic and do not conform to popular classifications. Second, learning and discovery systems require an explication of how concepts are related, in particular what information is inherited, modified, added and/or excluded, and upon what information were these changes based. To adequately deal with these problems, the semantics of inheritance must be moved from the procedures to the representation. This view led to the creation of a general inheritance relation with an associated inheritance concept. The inheritance concept explicitly defines what information is inherited by concept, and what additions, deletions and substitutions are made. It also describes the set of choices available in making the alterations and why a particular alteration was chosen.

4. References

1. Brachman R.J., (1977), "A Structural Paradigm for Representing Knowledge," (Ph.D. Thesis), Harvard University, May 1977.

2. Fahlman S.E., (1977), "A System for Representing and Using Real-World Knowledge," (Ph.D. Thesis), Artificial Intelligence Laboratory, MIT, AI-TR-450.

3. Fahlman S.E., (1979), Private Communication.

4. Fox M.S., (1978), "Knowledge Structuring: An Overview," <u>Proceedings of the Second Conference of the Canadian Society for Computational Studies of Intelligence</u>, Toronto Ont., July 1978.

5. Hayes P.J., (1977), "On Semantic Nets, Frames and Associations," <u>Proceedings of the Fifth Internatonal Joint Conference on Artificial Intelligence</u>, Cambridge MA, Aug. 1977.

6. Lenat D.B., (1976), "AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search,", (Ph.D. Thesis), Computer Science Dept. Stanford U., A1M-286.

7. Levesque H, and J. Mylopoulos (1978), "A Procedural Semantics for Semantic Networks," AI MEMO 78-1, Dept. of Computer Science, University of Toronto.

8. Moore J., and A. Newell, (1973), "How can Merlin Understand," In L. Gregg (ed.), <u>Knowledge and Cognition</u>, Potomac MD.: Lawrence Erlbaum Associates.

9. Winston P.H (1978), "Learning by Creatifying Transfer Frames," <u>Artificial Intelligence</u>. Vol. 10, pp. 147-172.

Examples

```
1. {*Concept
    View: <view-slots>
    Meta-corpus: <meta-corpus-slots>
    Corpus: <corpus-slots>}
2. {*Mammal
    Corpus:
      »Nursing-Method: aBreast
      *Birth-Process: alive
      *color:
      •Head:}
3. {*Mammal
    Corpus:
      •Nursing-Method: •Breast}
4. {*Monotreme
    View:
      *Is-a: aMammal
    Corpus:
       *Birth-Process: *Egg-laying}
5. {*Platypus
   View:
     *Is-A: aMammal
      (Corpus:
        (CONTRADICT DESCRIPTION OF *Birth-Process SLOT
           WITH aEgg-laying)
        (REFINE SLOT *Head TO SLOT *Bill AND SLOT *Skull))}
6. {*Purple-Platypus
   View:
     •Is-a: aPlatypus
     (Corpus: (ADD VALUE ePurple TO DESCRIPTION OF
         SLOT *Color FROM SET {ared, *yellow, *purple}
         FOR REASON "environment is purple"))}
```