SECTION 4

Inheritance Theory

Introduction

Inheritance theory performs reasoning within a graphical framework. It's based upon the notion of an *inheritance hierarchy* or *inheritance network* which, in turn, was derived from the concept of semantic networks [7]. For example, suppose we wanted to construct an inheritance hierarchy for the following discourse:

> *Clyde is an elephant.*
> *Elephants are mammals and they are gray.*

The inheritance hierarchy could be depicted as follows:

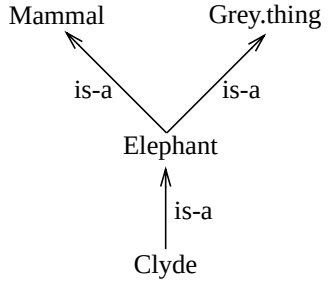

FIGURE 4.1: Inheritance Hierarchy for Clyde the Elephant.

Inheritance Hierarchies

The nodes of an inheritance hierarchy, such as the one depicted in Figure 4.1, represent knowledge and the arcs represent relationships between the nodes. We can think of an inheritance hierarchy as having three conceptual levels. At the first level the nodes represent instances or individuals such as Clyde. As we go higher in the hierarchy the nodes represent classes or types such as Elephants. Lastly, at the third level we find classes of properties. The following is some necessary definitions we need when talking about inheritance hierarchies [9].

General Information

More formally, an inheritance hierarchy is a *directed acyclic graph*. The nodes stand for individuals, classes of properties, or generic concepts. The arcs between nodes represent the relationship between them. These relationships are restricted to either *is-a* or *is-not-a*. For example, in Figure 1., the arc from Clyde to Elephant represents the information that Clyde *is-a* Elephant. That is, Clyde is a specific elephant. The arcs that represent the is-a relationship are referred to as *positive links* while the arcs that represent the is-not-a relationship are referred to as *negative links*.

A *path* within a hierarchy is defined like a path within a graph (see [11] for more on graph theory). For example, if the sequence of arcs $v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_{n-1} \rightarrow v_n$ are included in the hierarchy, then $v_0, \ldots, v_n$ is considered a path with v_0 as it's *start point* and v_n as it's *end point*. Paths can be referred to as a *positive path* or as a *negative path*. A path is defined as a positive path, like our previous example, unless it's last link is a negative link. Then it is a negative path. For example, if the sequence of arcs $v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_{n-1} \not\rightarrow v_n$ are included in the hierarchy, then $v_0, \ldots, \neg v_n$ is considered a negative path. The *polarity* of the path is based upon this same concept. If the path is a positive path, then the polarity is positive. Otherwise, the polarity is negative.

Inferences

The concept of drawing an *inference* or *conclusion* is based on a path and it's polarity. If we form an arc with the start point and the end point of a path along with the polarity of the path for the relationship, then we have formed an inference. Using our previous two examples, we could form the conclusions $v_0 \rightarrow v_n$ and $v_0 \not\rightarrow v_n$ which we could read as v_0 is-a v_n and v_0 is-not-a v_n respectively. An *extension* within a hierarchy is the set of all paths that support an inference. A class is said to be *inheritable* by the individual if all the paths from the individual to the class are positive. Otherwise, the class is said to be *un-inheritable*.

Paths can be constructed in a downward fashion or in an upward fashion. Hence, if the path $v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_{n-1} \rightarrow v_n$ is given within an inheritance hierarchy, following the downward concatenation of the path, we can infer is either $v_0 \rightarrow v_n$ or $v_0 \not\rightarrow v_n$ only if we infer $v_1 \rightarrow v_n$ or $v_1 \not\rightarrow v_n$ respectively. If we follow the upward concatenation of the path, we can infer either $v_0 \rightarrow v_n$ or $v_0 \not\rightarrow v_n$ only if we infer either $v_0 \rightarrow v_{n-1}$ or $v_0 \not\rightarrow v_{n-1}$ respectively.

Inferences are usually made with respect to a reasoner. Reasoners are divided into two categories: credulous and skeptical reasoners. Given the path $v_0 \rightarrow \ldots \rightarrow v_n$ and the path $v_0 \rightarrow \ldots \not\rightarrow v_n$ within an inheritance network, a *credulous reasoner* would infer either the conclusions $v_0 \rightarrow v_n$ or $v_0 \not\rightarrow v_n$ and a *skeptical reasoner* would infer neither the conclusion $v_0 \rightarrow v_n$ nor $v_0 \not\rightarrow v_n$. For example, in Figure 4.2, a credulous reasoner could infer that A is-a D and that A is-not-a D because there are paths to support these inferences. However, a skeptical reasoner would not infer either because, although there are paths to support the inferences, the inferences are contradictory each other.

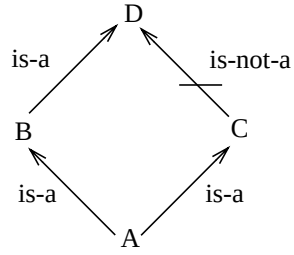

FIGURE 4.2: A Hierarchy with Contradictory Information.

Pre-emption

Lastly we need to consider pre-emption. Pre-emption is the basic idea that more specific information should override less specific information within a hierarchy. There are two methods of performing pre-emption: on-path and off-path. *On-path pre-emption* is that a path may pre-empt another path if there is a redundant link on the pre-empted path that would short circuit the pre-emptor. *Off-path pre-emption* is a path where explicit information is used whether there is a redundant link or not.

Reasoning in Inheritance Hierarchies

Negative arcs provide power that is highly desirable. They allows us to override inheritable properties which would have been gained through positive links. But

this power comes with a price because it poses two problems which we must overcome: ambiguity and redundant arcs [9, 10]. That is, the real question where negative arcs are involved, is which extension do we choose?

Redundant Arcs

Redundancy in inheritance hierarchies is the problem of choosing one class among related classes. Let us consider the following inheritance hierarchy. It represents the following discourse.

> *Tweety is a penguin.*
> *Penguins are birds that do not fly*
> *Birds are flying thing.*
> *Tweety is a bird.*

The inheritance hierarchy is depicted below:

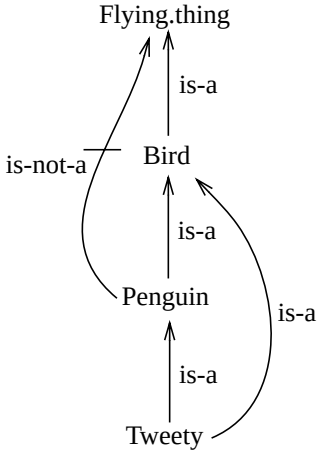

FIGURE 4.3: Inheritance Hierarchy for Tweety the Land Loving Bird

The inheritance hierarchy represented in Figure 4.3 is built directly from the discourse above. This is highly desirable because we have not lost or modified the meaning of what was original given to us regardless if the discourse was given all at once or in different pieces over time. However, this creates a problem within the hierarchy because we may be able to derive contradictory conclusions. For example, we can derive that Tweety is-a Flying.thing and the Tweety is-not-a Fly-

ing.thing. We can derive that Tweety is-a Flying.thing because we know that Twenty is-a Bird and that a Bird is-a Flying.thing. Deriving that Twenty is-not-a Flying.thing can be done because we know that Twenty is-a Penguin and we also know that a Penguin is-not-a Flying.thing. To overcome this problem, we need some methodology to make a choice between redundant links.

Several methods have been discussed to solve the redundancy problem. One suggestion, called on-path pre-emption, is to allow more specific information to override more general information. While this solution is dependent on a redundant link being present, another solution, called off-path pre-emption, does not depend upon a redundant link. The idea behind it is to allow specific explicit information to override more general information. Exceptional inheritance reasoning is another example of a method to handle non-monotonic reasoning within inheritance structures.

Ambiguity

Ambiguity is the problem of choosing one class among unrelated classes. Let us consider the Nixon diamond problem [8] which is based on the following discourse.

Quakers are pacifist.
Nixon is a Republican.
Nixon is a Quaker.

The discourse is depicted as an inheritance hierarchy in the following figure:

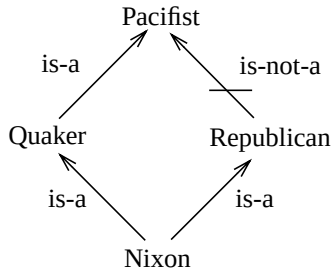

FIGURE 4.4: Inheritance Hierarchy for Nixon Diamond Problem.

Clearly, in Figure 4.4, we can derive that Nixon is-a Pacifist and also that Nixon is-not-a Pacifist. We can do this by acknowledging the paths from Nixon to Pacifist via Quaker and Republican respectively. But because no information is given as to

which class to choose over the other we could derive that Nixon is a pacifist, that Nixon is not a pacifist, or both.

There are two common methodologies to solve this problem: *ambiguity blocking inheritance* and *ambiguity propagation inheritance*. Ambiguity blocking prevents ambiguity as it happens. It does this by removing all incoming or outgoing arcs from an ambiguous node. Thus, it is an idea that hopes to stop more ambiguity at a later time. For example, in Figure 4.5, F and D are ambiguous with respect to A.

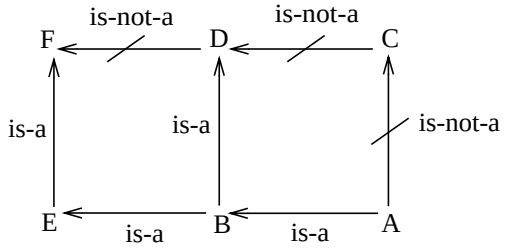

FIGURE 4.5: Inheritance Hierarchy with Ambiguity.

Using ambiguity blocking the following hierarchy depicted in Figure 4.6. would be produced. The incoming and outgoing arcs from D are removed because it has a shorter path than F from A. Thus, in the process, making F unambiguous with respect to A.

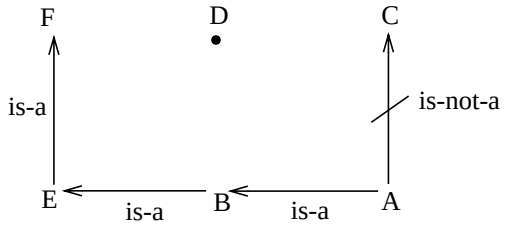

FIGURE 4.6: Ambiguity Blocking Inheritance applied to Figure 4.5.

Ambiguity propagation inheritance takes a similar but more drastic approach. It would simply remove all traces of ambiguity. Because ambiguity is essentially contradictory information ambiguity propagation takes the point of view that no real choice can be made. Thus, no choice should be made. For example, Figure 4.7 depicts the inheritance hierarchy that would be created using ambiguity propagation. Because F and D are ambiguous with respect to A all incoming and outgoing

arcs are removed. By doing this D and F are no longer ambiguous. Several other versions of ambiguity propagation have been introduced but are not discussed here

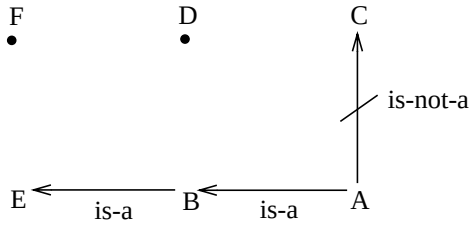

FIGURE 4.7: Ambiguity Propagation Inheritance applied to Figure 4.5.

Conclusion

Inheritance theory is like case based reasoning. It uses specific knowledge to reason. Although, unlike case based reasoning, it uses general rules to draw conclusions and makes no attempt to reason based on previous examples. The general rules it uses are based upon the concept of a path within in a graph. Thus, inheritance theory is reasoning based on specific knowledge that takes a step towards reasoning with general knowledge. As one of the newer models of human reasoning, inheritance theory has not really received as much attention as the others.

References

[1] Rips, L.J., Reasoning, *Annual review of psychology*, Volume 41, 1990, Pages 321-354.

[2] Kurtz, K.J., Genter, D., and Gunn, V., Reasoning, *Cognitive Science*, 1999, Pages 145-200.

[3] Aamodt, A. and Plaza, E., Cased-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *AI Communications*, Volume 7, Number 1, March 1994, Pages 39-59.

[4] Kolodner, J., *Case-Based Reasoning*, Morgan Kaufmann Publishers, Inc., 1993.

[5] Leake, D.B., *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press/MIT Press, 1996.

[6] Bergmann, R. and Wolfgang, W., On the role of Abstraction in Case-Based Reasoning. *Advances in Case-Based Reasoning*, Volume 1168 of Lecture Notes in Artificial Intelligence, Springer, 1996, Pages 28-43.

[7] Quillian, M.R., Semantic Memory, *Semantic Information Processing*, 1968, Pages 227-270.

[8] Reiter, R., A Logic for Default Reasoning, *Artificial Intelligence*, Volume 13, North Holland Publishing Company, 1980, Pages 81-132.

[9] Al-Asady, R., *Inheritance Theory: An Artificial Intelligence Approach*, Ablex Publishing Corporation, 1995.

[10] Schlechta, K., *Nonmontonic Logics, Basic Concepts, Results, and Techniques*, Springer, 1997.

[11] Harary, F., *Graph Theory*, Addison Wesley, 1969.

[12] Davis, E., www-formal.stanford.edu/leora/cs, New York University, September 1997.

[13] Hayes, P., and Shastri, L., www-formal.stanford.edu/leora/cs, University of West Florida and International Computer Science Institute, July, 1997.

[14] Davis, E., *Representations of Commonsense Knowledge*, Morgan Kaufmann Publishers, Inc., 1990, Pages 106-109.

[15] Barden, J., Lectures in Commonsense Reasoning, *Artificial Intelligence II*, New Mexico State University, Fall 1995.

[16] McDermott, D. and Doyle, J., Non-Monotonic Logic I, *Artificial Intelligence 13*, 1980, Pages 41-72.

[17] Ebbinghaus, H.-D., Flum, J.T., W., *Mathematical Logic*, Springer-Verlag, New York, 1980

[18] Gehrke, M., Lectures in Mathematical Logic, *Mathematical Logic*, New Mexico State University, Fall 1994.

[19] Reiter, R., On Closed World Data Bases, *Logic and Data bases*, Plenum Press, 1978, Pages 55-76.

[20] McCarthy, J., Circumscription - A form of Non-Monotonic Reasoning, *Artificial Intelligence*, Volume 13, North-Holland Publishing Company, 1980, Pages 27-39.

[21] Reiter, R., A Default Logic for Default Reasoning, *Artificial Intelligence 13*, North Holland Publishing Company, 1980, Pages 81-132.

[22] Marek, V.W., Non-monotonic Reasoning: Recent Advances, Questions and Future Directions, Invited Speaker, *8th International Workshop on Non-monotonic Reasoning*, 2000.

[23] McCarthy, J., Application of Circumscription to Formalizing Commonsense Knowledge, *Artificial Intelligence*, Volume 26, North-Holland Publishing Company, 1986, Pages 89-116.

[24] Gelfond, M., Lifschitz, V., The Stable Model Semantics for Logic Programming, *Proceedings of the 5th International Conference on Logic Programming*, The MIT Press, 1988, Pages 1070-1080.

[25] Gelder, A.V., Ross, K.A., Schlipf, J.S., The Well-Founded Semantics for General Logic Programs, *Journal of the ACM*, Volume 38, Number 3, July 1991, Page 620-650.

[26] Niemela, I., Simons, P., Smodels - An Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, 1996, Pages 289-303.

[27] Rao, P., Sagonas, K., Swift, T., Warren, D., and Freire, J., XSB - A System for Efficiently Computing Well-Founded Semantics, Logic Programming and Non-Monotonic Reasoning, *Proceedings of the Fourth International Conference*, Springer, 1997, Pages 430-440.

[28] Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F., A Deductive System for Non-Monotonic Reasoning, *Proceedings of the 4th International Conference on Logic Programming and Non-monotonic Reasoning*, Springer, 1997.

[29] Cholewinski, P., Marek, V.M., Mikitiuk, A., Truszczynski, M., Computing with Default Logic, *Artificial Intelligence*, Volume 112, Number 1 and 2, August 1999, Pages 105-146.

[30] Blum, J., A Machine-Independent Theory of the Complexity of Recursive Functions, *Journal of the ACM 14*, Volume 2, 1967, Pages 322-336.

[31] Papadimitriou, C.H., *Computational Complexity*, Addison-Wesley, 14, Volume 2, 1994, Pages 322-336.

[32] Hopcroft, J.E., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[33] Ranjan, D., Lectures in Computability, *Theory of Computation*, New Mexico State University, Spring 1995.

[34] Gottlob, G., Complexity Results for Non-Monotonic Logics, *Journal of Logic and Computation 2*, 1992, Pages 397-425.

[35] Stillman, J., The Complexity of Propositional Default Logics, *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992, Pages 794-800.

[36] Papadimitriou, C.H., On Finding Extensions of Default Theories, *Proceedings of the International Conference on Database Theory*, 1992, Pages 276-281.

[37] Kautz, H.A. and Selman, B., Hard Problems for Simple Default Logics, *Artificial Intelligence*, 1991, Pages 243-279.

[38] Stillman J., It's Not My Default: The Complexity of Membership Problems in Restricted Propositional Default Logics, *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990, Pages 571-578.

[39] Dimopoulos, Y. and Magirou, V., A Graph Theoretic Approach to Default Logic, *Journal of Information and Computation*, Volume 112, Number 2, 1994, Pages 239-256.

[40] Baader, F. and Hollunder, B., Embedding Defaults into Terminological Knowledge Representation Formalisms, *Journal of Automated Reasoning*, Volume 14, Number 1, 1995, Pages 149-180.

[41] Apt, K.R. and Blair, H.A., Arithmetic Classification of Perfect Models of Stratified Programs, Fifth International Conference and Symposium on Logic Programming, 1988, Pages 766-779.

[42] McCarthy, J., Applications of Circumscription to Formalizing Commonsense Knowledge, *Artificial Intelligence*, Volume 26, 1986, Pages 89-116.

[43] Schlipf, J.S., Decidability and Definability with Circumscription, *Annals of Pure and Applied Logic*, Volume 35, Number 14, 1987, Pages 173-191.

[44] Eiter, T. and Gottlob, G., Propositional Circumscription and Extended Closed World Reasoning are Π_2^P , *Theoretical Computer Science*, Volume 114, 1993, Pages 231-245.

[45] Schlipf, J.S., When is Closed world Reasoning Tractable?, *Proceedings of the 3rd International Symposium on Methodology for Intelligent Systems*, 1988, Pages 485-494.

[46] Papadimitriou, C.H., On Selecting a Satisfying Truth Assignment (Extended Abstract), *Proceeding of the 32nd Annual Symposium on the Foundations of Computer Science (FOCS-91)*, 1991, Pages 163-169.

[47] Gottlob, G. and Fermuller, C.G., Removing Redundancy from a Clause, *Artificial Intelligence*, Volume 61, Number 2, 1993, Pages 263-289.

[48] Kolaitis, P.G. and Papadimitriou, C.H., Some Computational aspects of Circumscription, *Journal of the ACM*, Volume 37, 1990, Pages 1-14.

[49] Cadoli, M. and Schaerf, M., A Survey of Complexity Results for Non-Monotonic Logics, *Journal of Logic Programming*, Volume 17, 1993, Pages 127-160.

[50] Chang, C.L., Lee, R.C.T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973, Pages 7-92.