An Efficient Feature-based Connectionist Inheritance Scheme

Ron Sun

Department of Computer Science

The University of Alabama

Tuscaloosa, AL 35487

To appear in: IEEE Transaction on System, Man and Cybernetics

Abstract

The paper describes a connectionist model that deals with the inheritance problem in an efficient and natural way. Based on the connectionist architecture CONSYDERR, we analyze the problem of property inheritance and formulate it in ways facilitating conceptual clarity and connectionist implementation. A set of "benchmarks" is specified for ensuring the correctness of solution mechanisms; parameters of CONSYDERR are formally derived to satisfy these benchmark requirements. The paper also discusses how chaining of *is-a* links and multiple inheritance can be handled in this architecture. This paper shows that CONSYDERR with a two-level dual (localist and distributed) representation can handle inheritance and cancellation of inheritance correctly and extremely efficiently, in constant time instead of proportional to the length of a chain in an inheritance hierarchy. It also demonstrates the utility of a meaning-oriented intensional approach (with features), for supplementing and enhancing extensional approaches. [1]

[1] ACKNOWLEDGEMENTS. I wish to thank Dave Waltz, James Pustejovsky and Tim Hickey for many helpful discussions. I also thank the anonymous reviewers for their detailed and helpful comments.

1 Introduction

Inheritance is one of the central problems in artificial intelligence, and it has been receiving a lot of attention. Many different variations of the inheritance problem have been studied and many different solutions have been proposed. Most knowledge representation systems nowadays embody some kinds of inheritance mechanisms; various implementations exist, ranging from simple ones to highly complex parallel spreading activation systems. The intensity of activities in this area signifies its importance — inheritance serves as a foundation for building intelligent systems.

In a nutshell, inheritance is the derivation of information for one concept (class) based on known information associated with another concept (class) in a hierarchically structured knowledge base. Hierarchical structuring of knowledge allows economical representation, by minimizing the redundancy of information stored, and efficient reasoning, by directing inference in a predetermined way. Knowledge representation systems rely on inheritance to enable certain useful and frequently made inferences (i.e. inheritance of properties) to be made. The research in this area centers on developing sound mechanisms that can handle various inheritance situations correctly and efficiently.

Among all the work in this area, Touretzky [25] is by far the most comprehensive study of inheritance. In his work, inheritance networks are construed to be *lattice* structures, enabling full mathematical analysis of them, and a clear semantics is developed for the problem. This analysis leads to the rule of *the shortest inferential distances* for deducing correct results; various kinds of inheritance scenarios are analyzed in his work to show how each of them can be handled correctly by the rule.

Instead of dealing only with top-down inheritance (inheritance from a superclass to a subclass), Shastri [13] presents an evidential framework for the inheritance problem in both top-down and bottom-up directions and a connectionist implementation of the evidential formulation. His treatment of inheritance is based on counting the numbers of objects with different property values, and the conclusion is reached with information maximization based on the counts. The connectionist implementation is localist and makes use of several specially designed node types for evidential combinations. Cottrell [31] also implements a connectionist solution.

Yager [28], on the other hand, presents a *possibilistic* treatment of the inheritance problem based on the fuzzy set theory (Zadeh [29], [30]). Basically, results of inheritance are rated with a possibilistic measure and the highest rated result is adopted; other apparatuses are also needed, one of which is the ordering of facts from the most specific to the most general. The idea of a fuzzy solution to the inheritance problem is very interesting, but the particular solution seems somewhat ad hoc. Mili & Rada's work [11] represents a different approach in applying the fuzzy set theory to inheritance and deals with inheritance as *generalized fuzzy regularity*; that is, hierarchical relationships between concepts are to reflect relationships between the properties of these concepts and vice versa. It treats both top-down and bottom-up inheritance in one framework. It is a mathematically motivated treatment, not a cognitively motivated one.

In this paper, we advocate the view that inheritance is better dealt with intensionally, instead of extensionally. Briefly, *extension* is the class of objects that fits a concept; [2] and *intension* is the "meaning" of a concept (and of the class it describes), or more precisely it is a set of *features* that is common and jointly peculiar

[2] Note that *extension* as used here (see [10] for more explanation) is a totally different concept from the one used in Touretzky [25].

to a concept [10]. Extensional approaches thus only deal with the classes (of objects) per se, while an intensional approach also takes into account the intensions of classes. This distinction can be likened to that of syntactics vs. semantics, in that extensional approaches deal only with syntactical relations while intensional approaches also deal with semantical content of concepts. We believe that inferences in an inheritance system should be based on intensions (features, semantics, etc.) of concepts involved at least to a certain extent, rather than based solely on syntactical, extensional relations (such as *is-a*); for intensional approaches represent a deeper and more fundamental (meaning-oriented) understanding of concepts and of relationships between concepts. The problems with the existing purely extensional approaches are the following:

- The complexity and often ad hoc-ness of mechanisms for handling various cancellation scenarios ([28], [2], and [13]).
- The difficulty in finding correct solutions to various situations ([24] and [13]), such as multiple inheritance.
- The computational inefficiency resulting from the complex mechanisms employed.

As will be shown in the paper, intensional approaches provide ways for remedying these problems.

CONSYDERR is a connectionist architecture developed for dealing with commonsense reasoning in general [16], [3] which is used in this work to carry out the intensional approach with a two-level representation. We adopt this architecture for the following reasons (as will be demonstrated later on):

[3] For similar connectioinist models, see [1], [5], [6], etc.

- It provides a computationally efficient (massively parallel) mechanism.
- It provides a way for implementing an intensional/semantical (vs. extensional/syntactical) approach to the inheritance problem.
- The inheritance problem is reduced to the more general problem of rule application and similarity matching, and is handled uniformly along with other problems.

The plan for the rest of the paper is as follows: we first analyze the problem of property inheritance and formulate it in ways facilitating its solution; a set of "benchmarks" is specified for ensuring the correctness of inheritance mechanisms we develop. The connectionist architecture CONSYDERR developed in [16] and [17] is presented as a means of implementing our formulation of inheritance; the parameters of CONSYDERR are then derived to satisfy the benchmark requirements specified. The chaining of *is-a* links and the problem of multiple inheritance are also analyzed. Some comparisons with other approaches are made at the end.

2 Basic Inheritance

2.1 Two Types of Links

First, some clarifications shall be made here regarding the basic notations used in formulating the inheritance problem. Touretzky [25] describes inheritance as inheritance between classes (concepts), with only *is-a* links used to connect various classes. In our discussion, however, a slightly different formulation of the problem will be used: instead of all *is-a* links, we will have two types of links: *is-a* and *has-property-value*; *is-a* is used here to denote relations between superclass and subclass; *has-property-value* is used to denote a property value of a class. Although mathematically through some transformation one type can be turned into another, the resulting representation is unnatural. [4] These two types of relations are fundamentally different.

For example, the elephant example can be expressed in Touretzky's formalism as:

> elephant *is-a* gray-thing
>
> royal-elephant *is-a* white-thing
>
> royal-elephant *is-a* elephant

Here "gray things" or "white things" are not natural categories, although one can always make up such categories if so desired. In our formalism, the example can be expressed, more naturally, as

> elephant *color* gray
>
> royal-elephant *color* white
>
> royal-elephant *is-a* elephant

Or equivalently,

> elephant *has-property-value* color-gray [5]

[4] Let p represent a particular property, v its value, and c a class. We can make a class out of all classes that have a certain value for a certain property, i.e.

$$c = c_1 \cup c_2 \cup \ldots\ldots \cup c_n$$

such that

$$p(c_i) = v \quad i = 1, 2, \ldots\ldots, n$$

and

$$p(x) \neq v \quad if \quad x \neq c_i$$

[5] It means that elephant has a property *color* whose value is *gray*. The same applies below.

royal-elephant *has-property-value* color-white

royal-elephant *is-a* elephant

Here we treat a property-value pair (e.g., color-gray) as one single concept.

Now we want to show that the above formulation of inheritance can be described by *rules* and *similarities*. The elephant example can be translated into the following rules for expressing property values,

$$elephant \longrightarrow color\text{-}gray$$

$$royal\text{-}elephant \longrightarrow color\text{-}white$$

and similarities between royal elephants and elephants (for *is-a* or subclass/superclass relations):

$$elephant \sim royal\text{-}elephant$$

That is, *is-a* (the subclass/superclass relation) is handled here as a special case of similarity. Furthermore, the similarity can be described by feature overlapping, and thus the superclass/subclass relation (*is-a*) is a special case of feature overlapping; it is special because the feature set of the superclass is always contained in that of the subclass (the reverse containment). This is based on a well known principle of philosophical logic: the larger the extension, the smaller the intension, and the extension of a concept with an intension that is a proper subset of the intension of another concept is a superset of the extension of the other concept. Here intension means the "meaning" (or common features) of a concept, and extension means the class of objects that is described by a concept. Similarity in general can be attributed to the overlap in the meanings of two concepts; when we decompose "meanings" into features, similarity amounts to the overlap of the feature sets of the two concepts. Therefore, in this sense, a superclass and a subclass are similar to each other, as a special case of the general similarity. In terms of the example, the feature set of "royal elephant" is a superset of the feature set of "elephant", because "elephant" is a more general concept (a larger class) and hence has less features (less specificity) associated with it; "royal elephant" is a subclass of "elephant" and hence contain all the features of "elephant" plus some others that are unique to it (for similarity issues, see [26], [27], and [23]).

We can express an inheritance problem in an "inheritance graph", which is a directed acyclic graph where nodes represent classes (concepts) and two types of conceptual links, *is-a* and *has-property-value*, are used to connect pairs of nodes. See Figure 1 for an example. In such a graph, *is-a* links can be implemented implicitly through the containment relations in the corresponding feature sets of the two concepts; the other type of links (*has-property-value*) can be implemented explicitly as rules as shown above, in which the premise of a rule is a concept node and the consequent is a property-value pair (for example, *elephant* $\longrightarrow$ *color-gray*).

2.2 A Set of Benchmarks

We shall investigate inheritance data to be accounted for in our solution, requirements they impose, and theoretical considerations and desiderata for the system we are aiming for. [6] The approach we take is: (1) first determining the range of commonsense reasoning problems to be solved (not limited to inheritance), (2) studying the data and problem characteristics, (3) figuring out a set of quantitative constraints, and finally (4) deriving solutions from those constraints. We discuss the first three steps in this subsection.

The range of problems to be solved is determined in [16]: first a set of psychological protocols (from [3]) that are difficult to account for by existing systems

[6] For clarity, in the following discussion we will assume concepts involved are binary; situations involving graded concepts are discussion in [16].

were analyzed, and the difficulty is identified and further analyzed into a number of aspects (since this is a separate topic, we will not repeat the analysis here). According to the analysis [16], the difficulty can be dealt with by *rule application* supplemented with *similarity matching*. Inheritance (including cancellation) can be expressed as mixed rule application and similarity matching as shown before; however, it also imposes some more stringent requirements as to the outcome of rule application and similarity matching, because it involves competitions of mutually conflicting concepts (property values), which gives rise to some subtle requirements ([25], [13] and [28]; See [16] for more analyses). So we have to deal with inheritance separately, in addition to the problems of similarity matching and rule application.

The requirements for the more general problems of similarity matching and rule application (not necessarily related to inheritance) were identified in [16] and can be summarized as follows:

Similarity. A similarity measure s_{AB} for "$A \sim B$" has the following requirements:

> $s_{AB} \propto |F_A \cap F_B|$, that is, the similarity between two concepts is proportional to the amount of their feature overlapping.
>
> $s_{AB} \propto \frac{1}{|F_B|}$, that is, the similarity is inversely proportional to the number of features B has, when everything else is equal.

Rules. The following cases of rules and mixed rules/similarities have respective requirements: [7]

[7] We only deal with rules with only a single premise here; rules with multiple premises are just extensions of these cases (see [16], [20] and [21] for a detailed treatment).

(1) A $\longrightarrow$ B: if A is activated, then $ACT_B = r_{AB} * ACT_A$, where ACT_x is the activation value of x and r_{AB} is the strength of the rule between A and B (the same below);

(2) A $\sim$ B, B $\longrightarrow$ C: if A is activated, then $ACT_B = s_{AB} * ACT_A$, and $ACT_C = r_{BC} * ACT_B$, where s_{AB} is the similarity between A and B (the same below);

(3) A $\longrightarrow$ B, B $\sim$ C: if A is activated, then $ACT_B = r_{AB} * ACT_A$, and $ACT_C = s_{BC} * ACT_B$;

(4) A $\longrightarrow$ B, B $\longrightarrow$ C: if A is activated, then $ACT_B = r_{AB} * ACT_A$, and $ACT_C = r_{BC} * ACT_B$;

(5) A $\longrightarrow$ B, B $\longrightarrow$ C, C $\longrightarrow$ D: if A is activated, then $ACT_B = r_{AB} * ACT_A$, $ACT_C = r_{BC} * ACT_B$, and $ACT_D = r_{CD} * ACT_C$;

(6) A $\sim$ B, B $\longrightarrow$ C, C $\longrightarrow$ D: if A is activated, then $ACT_B = s_{AB} * ACT_A$, $ACT_C = r_{BC} * ACT_B$, and $ACT_D = r_{CD} * ACT_C$;

(7) A $\longrightarrow$ B, B $\sim$ C, C $\longrightarrow$ D: if A is activated, then $ACT_B = r_{AB} * ACT_A$, $ACT_C = s_{BC} * ACT_B$, and $ACT_D = r_{CD} * ACT_C$;

(8) A $\longrightarrow$ B, B $\longrightarrow$ C, C $\sim$ D: if A is activated, then $ACT_B = r_{AB} * ACT_A$, $ACT_C = r_{BC} * ACT_B$, and $ACT_D = s_{CD} * ACT_C$;

(9) A $\sim$ B, B $\longrightarrow$ C, C $\sim$ D: if A is activated, then $ACT_B = s_{AB} * ACT_A$, $ACT_C = r_{BC} * ACT_B$, and $ACT_D = s_{CD} * ACT_C$.

On the other hand, in terms of **inheritance**, we should be able to deal with (1) inheritance of properties (top-down), (2) percolation of properties (bottom-up inheritance), and (3) cancellation of inheritance. Let us look into these cases in details (cf. [25]):

Let A be a superclass of B (i.e., A $\supset$ B), therefore the feature set of A is a subset of that of B (i.e., $F_A \subset F_B$). Suppose A has a property value C, and B has no corresponding property value. If B is activated then C should be activated to a certain extent from (top-down) inheritance. For example, temperate regions lie north to tropical regions in the northern hemisphere; subtropical regions constitute a subclass of temperate regions; so subtropical regions also lie north to tropical regions in the northern hemisphere.

Conversely, suppose B has a property value D and A has no corresponding property value, if A is activated, D should be activated from the percolation of property values (bottom-up inheritance) from B, a subclass of A, to A. For example, given that subtropical regions lie north to tropical regions in the northern hemisphere, and that the subtropical regions are a subclass of temperate regions, when there is no information to the contrary, we can infer, with certain confidence, that temperate regions lie north to tropical regions in the northern hemisphere.

The cancellation in property inheritance is more difficult to handle. Again suppose A is a superclass of B (A $\supset$ B).

1. As shown in Figure 2 (which depicts the feature sets), A has a property value C and B has a property value D $\neq C$; F_C and F_D do not overlap (C and D are not similar). If A is activated, C should win over D.

   For example, A = a warm, flat, fertile area with ample fresh water supply, B = a warm, flat, fertile area with fresh water supply but rugged, mountainous (i.e., A is a superclass of B), C = rice-growing area, and D = non-rice-growing area. $A \longrightarrow C$ and $B \longrightarrow D$. If given A, we should be able to deduce C not D.

2. As shown in Figure 3, A has a property value C and B has a property value D

$\neq C$; F_C and F_D do not overlap (C and D are not similar). If B is activated, D should win over C.

For example, A = a warm, flat, fertile area with fresh water supply, B = a warm, flat, fertile area with fresh water supply but rugged, mountainous (i.e., A is a superclass of B), C = rice-growing area, and D = non-rice-growing area. $A \longrightarrow C$ and $B \longrightarrow D$. If given B, we should be able to deduce D instead of C.

3. As shown in Figure 4, A has a property value C and B has a property value D which is a subclass of C (thus F_D is a superset of F_C). If A is activated, C should win over D.

   For example, A is a particular region (say, the South), B is a subarea of A (say, the Southeast), C is fruit-growing area, and D is orange-growing area (i.e., A is a superclass of B and C is a superclass of D). $A \longrightarrow C$ and $B \longrightarrow D$. So if A is given, C should have a higher activation value than D.

4. As shown in Figure 5, A has a property value C and B has a property value D which is a subclass of C (thus F_D is a superset of F_C). If B is activated, D should win over C.

   For example, A is a particular region (say, the South), B is a subarea of A (say, the Southeast), C is fruit-growing area, and D is orange-growing area (i.e., A is a superclass of B and C is a superclass of D). $A \longrightarrow C$ and $B \longrightarrow D$. So if B is given, D should have a higher activation value than C.

2.3 A Connectionist Implementation

CONSYDERR [8] provides a means for implementing a solution to the inheritance

[8]It stands for *a CONnectionist SYstem with Dual representation for Evidential Robust Rea-*

problem satisfying the above requirements. It is a two-level, dual-representation connectionist architecture for commonsense reasoning (see Sun [16]); one level of it, called CL, uses localist representation and the other level, called CD, uses distributed (feature-based) representation. The localist level (CL) is mainly for representing rules and concepts. Rules are carried out via links between nodes. The distributed level (CD) is mainly for carrying out similarity-based reasoning, to supplement and enhance rule-based reasoning mechanisms. Each node in CL is connected to all the relevant feature nodes in CD; once a CL node is activated, the related CD nodes will be activated subsequently from inter-level connection, and vice versa. Links in CL are replicated diffusely in CD by multiple links between two sets of feature nodes representing a premise and a consequent of a rule respectively. Similarity matching is actually accomplished through the interaction between the two levels, by a top-down/settling/bottom-up cycle. See Figure 6.

A set of equations for the computation of the three phases is as follows:

For the top-down phase,

$$ACT_{x_i}(t+1) = \max_A(td_A * ACT_A(t))$$

where ACT is the activation value of a node and A is any node in CL that has $x_i \in F_A$; td is a weight (to be determined later). That is, a feature node in CD receives activation from the corresponding nodes in CL, and chooses the largest value.

For the settling phase, in CL

$$ACT_C(t) = \sum_i r_i * A_i(t)$$

---

soning.

and in CD

$$ACT_{y_i}(t) = \sum_i lw_i * x_i(t)$$

where r's and lw's are link weights (representing rule strength), and lw's can be determined from corresponding r's (as will be shown later); A_i's and x_i's are the activations of nodes that are related respectively to C and y_i's by links (rules). That is, in this case, each node receives activations from other nodes at the same level (which are related to it by rules) and does a weighted-sum for computing its own activation. (This is actually a simplified description. For the lengthy detail, see [16] and [19].)

For the bottom-up phase,

$$ACT_C(t+1) = \max(ACT_C(t), \sum_{y_i \in CD_C} bu_C * ACT_{y_i}(t))$$

where C is any node in CL, y_i's are its corresponding feature nodes; bu is a weight (to be determined later). That is, a CL node receives activation from its corresponding CD node, and chooses the value as its activation if it is greater than its original activation.

Applying this cycle results in the following scenario:

First some nodes in CL get activated by external inputs (and clamped). Then the top-down phase will activate (and clamp) the CD nodes corresponding to the active CL nodes. In the settling phase, links representing rules related to those activated nodes take effect in both CL and CD. Concepts may have overlapping CD (feature) representations because they share some common features due to similarity; so some of the CD representations of concepts will be partially activated if a concept *similar* to them is activated in CD. Finally in the bottom-up phase, fully or partially activated CD (feature) representations will go back up to activate the corresponding nodes in CL. The result can be read off from CL. [9]

Notice the massive parallelism in the above specified architecture: activations are propagated, in a massively parallel fashion, from all pre-link nodes to all post-link nodes; each node receives inputs as soon as it can, and therefore fires as soon as it can, ensuring a maximum degree of parallelism in terms of rule application. In terms of similarity matching, all similar concepts are activated (in their CD representations) immediately once an original concept is activated, and simultaneously matched with the original one (through top-down and bottom-up flows); thus the architecture is extremely efficient by employing the two level structure. The parallelism in this architecture accounts well for the similar parallelism and spontaneity in human reasoning processes as identified in, for example, [16] and [3].

Moreover, fine-grained subsymbolic level (see Smolensky [14] and Dreyfus & Dreyfus [4]) in CONSYDERR enables intensional representation and meaning-oriented reasoning (vs. pure syntactical symbolic manipulation). By integrating extensional representation (at the conceptual, symbolic level) and intensional representation (at the subconceptual, subsymbolic level), and thus forming a two-level architecture, one can solve problems by utilizing the synergy between the two components ([16], [17] and [22]).

CONSYDERR handles all inheritance relations with two primitives: rules

[9] Each node in the system has one or more sites (cf. [5]), each of which computes the weighted-sum (or any other similar functions whenever needed) of the inputs. The maximum of the values computed by all the sites is taken to be the activation value of the node.

(e.g. *elephant* $\longrightarrow$ *color-gray*), implemented with links in CL and CD, and similarity (e.g. *elephant* $\sim$ *royal-elephant*), implemented implicitly through CD representations. Figure 8 shows an example of implementing inheritance in CONSYDERR: an inheritance graph, its description with rules and similarities, and its implementation in CONSYDERR with rules and feature set containment. This example corresponds to the last two scenarios of cancellation which we analyzed earlier, where the feature set of one property value is the subset of that of the other and thus one property value is a superclass of the other (the reverse containment relation). In that case, we have *dislike-elected-crook* $\subset$ *dislike-crook* and $F_{dislike\text{-}elected\text{-}crook} \supset F_{dislike\text{-}crook}$.

2.4 Deriving Parameters

With all the requirements (benchmarks) established in the previous sections, we can proceed to derive the exact specifications of the parameters for CONSYDERR, including top-down weights (denoted as td), bottom-up weights (denoted as bu), and weights for links between two CD nodes (denoted as lw) which diffusely replicate the rule links of CL (see Figure 7).

The formal derivation of these parameters, which is rather long, is in [16]. Let us summarize the derivation briefly: to determine these above parameters, we look at the requirements regarding rule application, similarity matching, and inheritance as enumerated before. Based on the desiderata associated with similarity matching, we choose a simple formula. Two cases of inheritance are analyzed to come up with the link weights (lw) in the CD level, and two other cases of inheritance are then analyzed to determine the bottom-up weights (bu). We then determine the top-down weights (td) in relation to the similarity measure and the other two parameters. Finally, the requirements for rule application are verified to be fully fulfilled.

The result from this derivation is surprisingly simple: suppose A and C are two arbitrary concepts and there is a link from A to C (i.e., a rule $A \longrightarrow C$), then

$$td_A = 1$$

$$lw_{AC} = \frac{r_{AC}}{f(|F_A|)}$$
$$bu_C = \frac{1}{g(|F_C|)}$$

where f and g are functions that are slower than but close to linear functions, and f is much closer to linear functions than g; r_{AC} is the link weight (the strength of the rule) from A to C; in case of binary rules, r_{AC} is either 1 or -1.

It is easy to verify that this set of parameters will work correctly. As an illustration, we will consider the first case of cancellation. When A is activated (but not B), we want C to be activated more strongly than D in CD. Then during the bottom-up process, these activation values will be transmitted to the corresponding CL nodes. [10]

Suppose A is activated, ACT_A is the activation of A and the activations of the feature nodes of A in CD due to top-down flow (ACT_A is 1, if we consider only binary cases), ϵ is the bottom-up weight (which is close to 1, the same for both C and D, because they both have only one feature node), and the rule strength $r_{AC} = r_{BD} = 1$:

$$\begin{aligned} ACT_C &= \epsilon \sum_{F_A} lw_{AC} * ACT_A \\ &= \epsilon |F_A| \frac{r_{AC}}{f(|F_A|)} ACT_A \end{aligned}$$

[10] As a general assumption, we take the MAX of the corresponding CL values and the bottom-up values, which represents the combination of the results of the conceptual level and the subconceptual level reasoning (see [16]).

$$
\begin{aligned}
&= \epsilon |F_A| \frac{1}{f(|F_A|)} ACT_A \\
ACT_D &= \epsilon \sum_{F_A} lw_{BD} * ACT_A \\
&= \epsilon |F_A| \frac{r_{AC}}{f(|F_B|)} ACT_A \\
&= \epsilon |F_A| \frac{1}{f(|F_B|)} ACT_A
\end{aligned}
$$

So C is activated more strongly than D (since $|F_A| < |F_B|$). Other cases of inheritance and cancellation can be verified in the same fashion. [11]

In short, we have derived the structural parameters in the CONSYDERR architecture necessary for solving the problem of property inheritance and cancellation. This connectionist architecture with a simple two-level structure solves a range of representation and reasoning problems, including inheritance, in a massively parallel manner and within a unified framework (Sun [16]).

3 Further Analysis of Inheritance

This section further analyzes how inheritance is dealt with by rules and similarities along the line we have sketched above, in terms of implementing a complete hierarchy (for either top-down inheritance or bottom-up inheritance, or both together), dealing with multiple inheritance, and forming new concepts.

3.1 Chaining of *is-a* Links

Now we will analyze how inheritance *hierarchies* can be formed with rules and similarities in CONSYDERR. An inheritance hierarchy is an acyclic structure

[11] In order to deal with inheritance correctly, the activation of nodes will be allowed to be slightly more than one, so that we can distinguish two different inherited properties (see [16]). For final output from a system, any activation greater than one will be treated as exactly one.

with nodes representing either a concept (a class of objects) or a particular property value pair (e.g. color-gray), connected by two types of links. The chaining of *is-a* links forms the backbone of an inheritance hierarchy (see Figure 9). The chaining of *is-a* links is handled in CONSYDERR, the same way as before, by utilizing overlapping feature representations (see Figure 10). The question now is how property values of a concept can be inherited correctly along this chain upwards or downwards.

Let us look at top-down inheritance first. By top-down we mean that the property values of a class are inherited by its subclasses. It seems easy to accomplish this task: based on similarity between the superclass and the subclass in their corresponding feature representations (one is contained in the other), the subclass can easily inherit the property values. So can the subclass of the subclass, the subclass of the subclass of the subclass, ... and so on. Troubles arise when there are one or more cancellations of the property values somewhere along the chain. In such cases, we have to make sure that correct values are inherited.

Suppose, in an inheritance graph, A is a class and B is its closest superclass that has a value for a property we want; that is, $A \subset B$ and $p(B) \neq \emptyset$, and there is no C such that $A \subset C$, $C \subset B$ and $p(C) \neq \emptyset$; where p(x) stands for the value of the property p for the class x. See Figure 11. To simplify the discussion, we assume there is no bottom-up inheritance. We want to have, regardless of other superclasses of A,

$$p(A) = p(B)$$

The question is how we can accomplish this. The only situation we have to look into is that there is D such that $A \subset B \subset D$ and $p(D) \neq \emptyset$. If p(D) =p(B), there is no difficulty at all accomplishing the above goal. If p(D) $\neq$ p(B), A has to inherit p(B), not p(D); in other words, the node representing p(B) has to fire more strongly than the node representing p(D). This is exactly what CONSYDERR will do, given the feature representation in CD. Intuitively, since we have $F_A \supset F_B \supset F_D$, the feature set of A is closer to the feature set of B than to the feature set of D; therefore A will inherit B's property values instead of D's. Mathematically, suppose p(B) and p(D) are represented in CD by one feature node each, according to the formulas specified before,

$$ACT_{p(B)} = \sum_{F_B} \frac{ACT_A}{f(|F_B|)}$$

$$= |F_B| \frac{ACT_A}{f(|F_B|)}$$

and

$$ACT_{p(D)} = \sum_{F_D} \frac{ACT_A}{f(|F_D|)}$$

$$= |F_D| \frac{ACT_A}{f(|F_D|)}$$

where ACT_x represents the activation of x, and ACT_A represents the initial activation of A. Since $F_D \subset F_B$, and f is a monotonic increasing function (slower than linear), it follows that $ACT_{p(B)} > ACT_{p(D)}$.

Let us now look into bottom-up inheritance. By bottom-up we mean that the property values of a class are "inherited" by its superclasses. Based on the similarity between the superclass and the subclass (one feature set is contained in the other), the superclass can easily activate the same property values. This inheritance can occur further and further along the *is-a* chain. When there is a cancellation along the chain, however, we have to make sure that correct values are inherited. Since this case is symmetrical to the top-down case, it is easy to see why CONSYDERR will work. Suppose, in an inheritance graph, A is a class and B is its closest subclass that has a value for a property we want; that is, $A \supset B$ and $p(B) \neq \emptyset$, and there is no C such that $A \supset C \supset B$ and $p(C) \neq \emptyset$, where p(x) stands for the value of the property p for the class x. See Figure 12. To simplify the discussion, we assume there is no top-down inheritance for the time being. We want to have, regardless whether or not there are other subclasses of A,

$$p(A) = p(B)$$

Suppose there is D such that $A \supset B \supset D$ and $p(D) \neq \emptyset$. If p(D) =p(B), there is no difficulty at all. If p(D) $\neq$ p(B), A has to inherit p(B) not p(D); in other words, the node representing p(B) has to fire more strongly than the node representing p(D). Mathematically, suppose p(B) and p(D) are represented in CD by one feature node each, according to the formulas specified before,

$$ACT_{p(B)} = \sum_{F_A} \frac{ACT_A}{f(|F_B|)}$$

$$= |F_A| \frac{ACT_A}{f(|F_B|)}$$

and

$$ACT_{p(D)} = \sum_{F_A} \frac{ACT_A}{f(|F_D|)}$$

$$= |F_A| \frac{ACT_A}{f(|F_D|)}$$

where ACT_A represents the initial activation of A. Since $F_D \supset F_B$, and f is a monotonic increasing function (slower than linear), it follows that $ACT_{p(B)} > ACT_{p(D)}$.

When there are both top-down inheritance and bottom-up inheritance, there are some complications that have to be analyzed. Suppose A is a class and B is its closest superclass that has a value for a property we want and C is the closest subclass that has a value for the property; that is, on one hand, $A \subset B$ and $p(B) \neq \emptyset$, and there is no D such that $A \subset D$, $D \subset B$ and $p(D) \neq \emptyset$; on the other hand, $C \subset A$ and $p(C) \neq \emptyset$, and there is no D such that $D \subset A$, $C \subset D$ and $p(D) \neq \emptyset$. If p(C) =p(B), there is no difficulty at all in having the correct property values inherited. If p(C) $\neq$ p(B), we have to make sure the right property value is inherited. Mathematically, suppose p(C) and p(B) are represented in CD by one node each, according to the formulas specified before,

$$ACT_{p(C)} = \sum_{F_A} \frac{ACT_A}{f(|F_C|)}$$

$$= |F_A| \frac{ACT_A}{f(|F_C|)}$$

and

$$ACT_{p(B)} = \sum_{F_B} \frac{ACT_A}{f(|F_B|)}$$

$$= |F_B| \frac{ACT_A}{f(|F_B|)}$$

Since $F_C \supset F_A \supset F_B$, and f is a monotonic increasing but slower-than-linear function, it follows that the result can go either way. This ambiguity is the direct result of the ambiguity in the knowledge given originally in the problem specification; we can resolve it once all the feature sets are known. For example, we will have $ACT_{p(B)} > ACT_{p(C)}$ if and only if $\frac{|F_B|}{f(|F_B|)} > \frac{|F_A|}{f(|F_C|)}$; or intuitively, to inherit from top-down, the feature set of the superclass should be as close to the feature set of A as possible. By the same token, to inherit from bottom-up, the feature set of the subclass should be as close to the feature set of A as possible.

In the above discussion we assume that each property value is represented by one node in CD. What if those property values form a hierarchy too? In case of top-down inheritance, suppose $A \subset B \subset D$, and p(B) and p(D) are represented in

CD by feature sets with $F_{p(B)} \supset F_{p(D)}$. [12] When A is activated, [13]

$$ACT_{p(B)} = \sum_{F_{p(B)}} \frac{\sum_{F_B} \frac{ACT_A}{f(|F_B|)}}{g(|F_{p(B)}|)}$$

and

$$ACT_{p(D)} = \sum_{F_{p(D)}} \frac{\sum_{F_B} \frac{ACT_A}{f(|F_B|)}}{g(|F_{p(D)}|)}$$

Since $F_D \subset F_B$ and $F_{p(D)} \subset F_{p(B)}$, and g is a monotonic increasing function (slower than linear), it follows that $ACT_{p(B)} > ACT_{p(D)}$. So the result is correct. See Figure 13.

In case of bottom-up inheritance, suppose $A \supset B \supset D$, and p(B) and p(D) are represented in CD by feature sets with $F_{p(B)} \subset F_{p(D)}$. When A is activated, we have

$$ACT_{p(B)} = \sum_{F_{p(B)}} \frac{\sum_{F_A} \frac{ACT_A}{f(|F_B|)}}{g(|F_{p(B)}|)}$$

and

$$ACT_{p(D)} = \sum_{F_{p(B)}} \frac{\sum_{F_A} \frac{ACT_A}{f(|F_B|)}}{g(|F_{p(D)}|)} + \sum_{F_{p(D)}-F_{p(B)}} \frac{\sum_{F_A} \frac{ACT_A}{f(|F_D|)}}{g(|F_{p(D)}|)}$$

Since $F_D \subset F_B$ and $F_{p(D)} \subset F_{p(B)}$, and f and g are monotonic increasing functions (slower than linear), we have $ACT_{p(B)} > ACT_{p(D)}$. So we obtain the correct result. See Figure 14.

Having analyzed all these cases, we are certain that CONSYDERR captures accurately inheritance in hierarchies. CONSYDERR provides an "implementation" of inheritance which is not exactly an implementation of explicitly *path-based* inheritance, but merely rules and similarities intermixed.

[12] It cannot be otherwise. See [25]. The same applies below.

[13] In this case, those nodes in $F_{p(D)}$ receive inputs from two different rules: $B \longrightarrow p(B)$ and $D \longrightarrow p(D)$, and the former is stronger than the latter. We use MAX in combining results from different rules. The same applies below

This analysis also shows that CONSYDERR provides a computationally efficient solution to inheritance: even in face of a huge hierarchy with long *is-a* chains, it can accomplish inheritance within constant time. All inferences are done in parallel and within one cycle, and there is no need to trace long chains since they are represented implicitly.

3.2 Multiple Inheritance

Multiple inheritance is a difficult problem for all inheritance systems. When different answers are obtained from inheritance through different paths, an inheritance system is in a ambiguous situation. CONSYDERR can provide a way for resolving ambiguity in multiple inheritance based on feature similarities, that is, deciding which inheritance path has the strongest strength, depending on which superclass (or subclass) is closer in feature representation to the focal concept. (In case that graded concepts and graded rule weights are allowed, we can take into consideration both the similarities of feature sets and the strengths of rules.)

A typical example of multiple inheritance is the Nixon Diamond (see Figure 15), which can be stated as follows (we treat *Nixon*, which is an instance, as a class of one element):

> Nixon is a Quaker. Nixon is a Republican. Republicans are non-pacifists. Quakers are pacifists. Is Nixon a pacifist or non-pacifist?

We shall represent the problem as follows:

$$Nixon \subset Republican$$

$$Nixon \subset Quaker$$

$$Quaker \longrightarrow Pacifist$$

$$Republican \longrightarrow Non\text{-}Pacifist$$

Our approach is to compare the feature sets of Republicans and Quakers, to see which one is closer to Nixon. [14] The general principle is: the property values of a class are to be inherited from a class that is the closest to it in terms of feature overlapping, when all rules connecting classes to their property-values have equal strengths.

Generally speaking, suppose that $A \supset C$ and $B \supset C$, and $A \longrightarrow E$ and $B \longrightarrow F$ (A and B are both superclasses of C, and each has a different property value E or F). Then in CD, $F_A \subset F_C$ and $F_B \subset F_C$. Assume all links have the same weights. Assume E and F are each represented by one single feature node in CD. Let us further assume that $|F_A| > |F_B|$. Since $|F_C| > |F_A|$ and $|F_C| > |F_B|$, A is closer to C in feature representation. Now if C is activated, after a cycle we have

$$ACT_E = \sum_{F_A} \frac{ACT_C}{f(|F_A|)}$$

and

$$ACT_F = \sum_{F_B} \frac{ACT_C}{f(|F_B|)}$$

where f is a monotonic increasing function slower than linear. So under the assumption that $|F_A| > |F_B|$, E is activated more strongly than F.

3.3 Emergent Concepts

For most inheritance systems, only concepts explicitly structured into an inheritance hierarchy can be brought out. However for many AI applications, some sort

[14] If we deal with non-binary cases, we also have to consider how strong each tendency is (a Quaker being a pacifist or a Republican being a non-pacifist), and then combine the scores to reach the final conclusion.

of a generative capability is very useful or even crucial; new concepts and classes need to be formed and generated on the fly, when the need for them arises.

For example,

> If carrying cargo, buy utility vehicles. If carrying passengers, buy passenger vehicles. If carrying both cargo and passengers, what shall one buy?

What we want is to have a system come up with a solution such as "van", which is both a passenger vehicle and a utility vehicle, without the need for a rule to take care of the particular situation (in case the rule base is incomplete or inconsistent for various reasons, as often seen in reality). In CONSYDERR, certain new concepts can be generated dynamically through the interaction in representation. An informal description is as follows (see [16] for the formal analysis): different types of vehicles are represented as nodes in CL and as features in CD; rules (*carrying-cargo* $\longrightarrow$ *utility-vehicle* and *carrying-passengers* $\longrightarrow$ *passenger-vehicle*) are represented as links; in response to the question, two concepts, *carrying-cargo* and *carrying-passengers*, are both activated, and their corresponding feature sets are activated in CD (in the top-down phase); then the two rules are activated (in the settling phase), so all the features corresponding to both utility and passenger vehicles will be activated in CD; all this activation goes up to CL (in the bottom-up phase). Things corresponding to the intersection of utility and passenger vehicles will be activated strongly, because they have all the activated features. So finally something like "van" will win in CL. This is only possible when an intensional approach is taken, and further shows that CONSYDERR is not just a straight "implementation" of the existing inheritance theories.

4 Concluding Remarks

Finally, we will have a brief comparison with other works, and then we will summarize the paper.

4.1 Comparisons

Touretzky [25], Shastri [13], and Thomason [24] are all path-based solutions; that is, in order to inherit from a concept, a path has to be found that connects the concept to inherit from and the concept to inherit to. This kind of schemes requires two things: a fixed, static hierarchy, and a mechanism for tracing the path. First of all, a path-based scheme entails the necessity of tracing paths step by step in performing inheritance; thus the computational complexity is proportional to the length of the longest path, to say the least. Contrarily, a feature-based scheme takes only constant time for all kinds of inheritance scenarios as shown above. This computational efficiency matches human commonsense reasoning well [16]. Second, a static hierarchy can be unwieldy in many respects. For instance, it is difficult to take context effects into consideration in performing inheritance when using a static hierarchy, because of its fixed topology. On the other hand, a feature-based scheme (implemented in a connectionist network such as CONSYDERR) can easily adapt itself by dynamicallly modifying connections from concept nodes to feature nodes, as connectionist systems customarily do ([16], [18] and [19]).

Most of the existing schemes of inheritance take extensional approaches. For example, the system in Shastri [13] counts the numbers of things in each class, and then by evidential combination, calculates the plausibility of a particular property values for a particular class; his system will conclude with the property value with the highest score. Contrary to that, our approach here is intensional; that is, the intensions (or semantics) of concepts are explored, and inheritance is performed based on the semantic closeness (i.e. feature overlap) of concepts. The intensional approach can result in better accuracy, because it can take account of meaning and semantics as well as syntactical, extensional relations. For example, one may see a lot more chickens that do not fly than birds that do fly (imagining living in a chicken farm). Still, one will conclude that birds usually fly, because it is an essential feature of birds. It is also my contention that the intensional approach represents a deeper and more fundamental understanding of concepts involved and thus better for any task that is suitable for the approach. Only when one does not or can not understand the nature of things, then one takes the extensional approach as a way out (and thus extensional approaches are complementary to our intensional approach).

While most of the existing solution schemes are specialized mechanisms for inheritance, our solution to the inheritance problem is provided as a part of the solution to problems of much broader scopes [16]. Therefore, it is not an ad hoc trick. The solution is actually based on a principled dichotomy of conceptual and subconceptual processing, utilizing rules and similarities (see [14] and [16]).

Recent work on skeptical inheritance (Stein [15], Horty et al [7] and [8], etc.) aims at deriving sound conclusions despite the existence of ambiguity. Roughly speaking, skeptical inheritance reaches one single set of unambuiguous conclusions from an inheritance hierarchy, by eliminating all ambuiguous paths in some way; an ideally skeptical inheritence system reaches those and only those conclusions that are justified by every possible inheritance scenarios ("credulous extensions"). While this is an important goal for some purposes, it is different from our goal of commonsensical evidential reasoning [16]. We are more interested in the question of how sensible (but fallible) conclusions can be reached efficiently in a cognitively plausible way (see [16] for arguments related to cognitive pluasibilities).

Most of connectionist (neural network) models use optimization as a means for reaching the most plausible conclusion (see e.g. [9]). However, as clear from the description of CONSYDERR above, we adopt a different approach, in which rule- and similarity-governed spreading activation implemented in a neural network fashion does the job of finding plausible conclusions; there is no overall energy function to minimize in our model. One of the relative advantages of this approach is that it avoids a host of problems associated with energy minimizing neural networks: for example, long settling time, local minima, global interaction (cf. [12]).

4.2 Summary

In this paper, a new inheritance scheme is proposed, which is based on a two-level, dual representation connectionist architecture CONSYDERR. This scheme utilizes features and the interactions between feature nodes and concept nodes. This scheme can perform inheritance extremely efficiently, with constant time. To summarize the main points:

- Intensional (feature-based) approaches are prefered over purely extensional aproaches, because they employ more detailed knowledge, which is oftentimes readily available.
- The feature-based approach has better computational efficiency, which matches the speed of human commonsense reasoning better.
- The solution in this scheme is formally derived, and thus has guaranteed correctness.

Although this approach has shown its promise, it is not a complete solution to all the problems within the scope; it is a solution to a subset of the most common problems that can be solved efficiently with a simple computational mechanism. Specifically, we do not deal with many intricate cases as in [24], [7], or [8] (such as various blocking scenarios, skeptical inheritance, alternative formulations, etc.); we do not handle evidential reasoning as in [13], and so forth.

Further exploration is definitely needed to better understand the approach. Future research will address more on the issue of multiple inheritance, delineating its boundary conditions; we will also further explore the issue of emergent concepts in CONSYDERR; we will analyze more difficult cases; and we will extend CONSYDERR to fuzzy concepts and fuzzy rules, to accommodate uncertainty in real world situations.

References

[1] J. Barnden, The right of free association: relative-position encoding for connectionist data structures, *Proc.10th Conference of Cognitive Science Society*, pp.503-509, Norwood, NJ: Lawrence Erlbaum Associates, 1988

[2] P. Besnard, *An Introduction to Default Logic*, Berlin, FRG: Springer-Verlag, 1989

[3] A. Collins and R. Michalski, The Logic of Plausible Reasoning: A Core Theory, *Cognitive Science*, Vol 13, No 1, pp.1-49, 1989

[4] H. Dreyfus and S. Dreyfus, *Mind Over Machine*, New York, NY: The Free Press, 1987

[5] J. Feldman and D. Ballard, Connectionist models and their properties, *Cognitive Science*, pp.205-254, July 1982

[6] S. Gallant, Connectionist Expert Systems, *Communication of ACM*, V.31(2), pp.152-169, 1988

[7] J. Horty, R. Thomason, and D. Touretzky, A skeptical theory of inheritence in nonmonotonic semantic networks, Proc.AAAI, pp.358-363, San Mateo, CA: Morgan Kaufman, 1987

[8] J. Horty, R. Thomason, and D. Touretzky, A skeptical theory of inheritence in nonmonotonic semantic networks, *Artificial Intelligence*, Vol.42, pp.311-348, 1990

[9] J. Hopfield, Neural Networks and Physical Systems with Emergent collective Computational Abilities, *Proc.National Academy of Science*, Vol.79, 1982

[10] H. Leonard, *Principle of Reasoning*, New York, NY: Dover, 1967

[11] H. Mili and R. Rada, Inheritance generalized to fuzzy regularity, *IEEE Trans.SMC*, Vol.20 No.5, pp.1184-1197, 1990

[12] D. Rumelhart, J. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Cambridge, MA: MIT Press, 1986

[13] L. Shastri, A Connectionist Approach to Knowledge Representation and Limited Inference, *Cognitive Science*, 12, pp.331-392, 1988

[14] P. Smolensky, On the proper treatment of connectionism, *Behavioral and Brain Sciences*, 11, pp.1-43, 1988

[15] L. Stein, Skeptical inheritance, Proc.AAAI, pp.1153-1158, San Mateo, CA: Morgan Kaufman, 1990

[16] R. Sun, Integrating Rules and Connectionism for Robust Reasoning, Ph.D Thesis, Brandeis University, Waltham, MA, 1991

[17] R. Sun, Connectionist Models of Rule-based reasoning, *Proc. 13th Cognitive Science Conference*, Norwood, NJ: Lawrence Erlbaum Associates, 1991

[18] R. Sun, Chunking and Connectionism, *Neural Network Review*, Vol.4, No.2, pp.76-78, 1991

[19] R. Sun, The Discrete Neuronal Models and the Probabilistic Discrete Neuronal Models, in: B. Soucek ed. *Neural and Intelligent System Integration*, New York, NY: John Wiley & Sons, 1991

[20] R. Sun and D. Waltz, Neurally Inspired Massively Parallel Model of Rule-Based Reasoning, in: B. Soucek ed. *Neural and Intelligent System Integration*, New York, NY: John Wiley & Sons, 1991

[21] R. Sun, Beyond associative memories: logics and variables in connectionist networks, *Information Sciences*, 1992, in press

[22] R. Sun, Connectionist Models of Commonsense Reasoning, *Knowledge Acquisition*, 1992, in press

[23] P. Thagard and K. Holyoak, How to compute semantic similarity, *Proc.DARPA Case-Based Reasoning Workshop*, pp.85-88, San Mateo, CA: Morgan Kaufman, 1989

[24] R. Thomason, Fahlman's NETL and Inheritance Theory, Manuscript, 1990

[25] D. Touretzky, *The Mathematics of Inheritance*, San Mateo, CA: Morgan Kaufman, 1986

[26] A. Tversky, Features of Similarity, *Psychological Review*, Vol.84, No.4, pp.327-352, 1977

[27] S. Vosniadou and A. Ortony, (eds.) *Similarity and Analogical Reasoning*, Cambridge, UK: Cambridge University Press, 1989

[28] R. Yager, Inheritance and possibility, *IEEE Trans.SMC*, pp.248-252, 1989

[29] L. Zadeh, Fuzzy Sets, *Information and Control*, Vol.8, pp.338-353, 1965

[30] L. Zadeh, Fuzzy Logic, *Computer*, Vol.21, No.4, pp.83-93, April 1988

[31] G. Cottrell, Parallelism in inheritance hierarchies with exceptions, Proc.9th IJCAI, pp.194-202, San Mateo, CA: Morgan Kaufman, 1985.

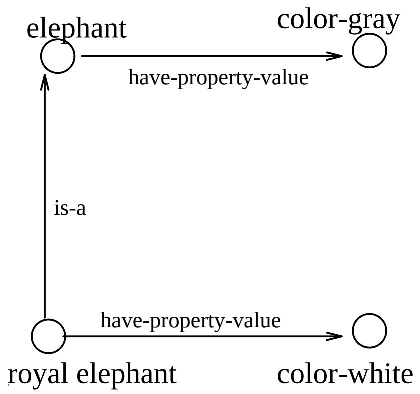

Figure 1: An Inheritance Graph

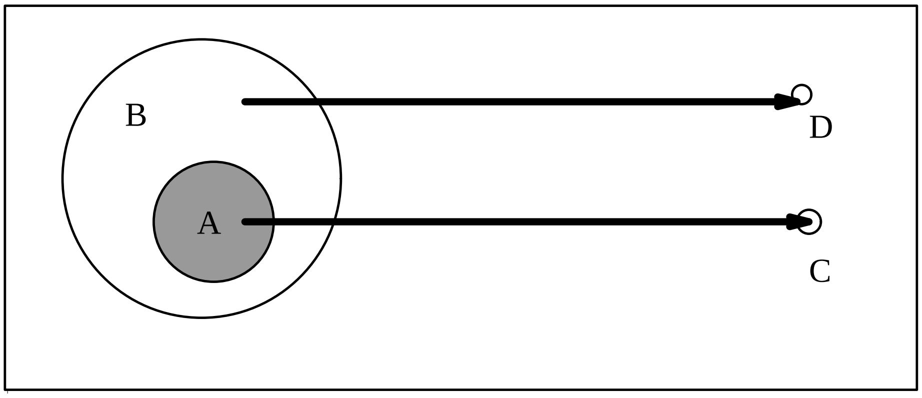

Figure 2: Cancellation of Inheritance Case I

$A \supset B$. $F_A \subset F_B$. Only the feature sets of the respective concepts are shown here. One arrow is drawn in place of pairwise connections.

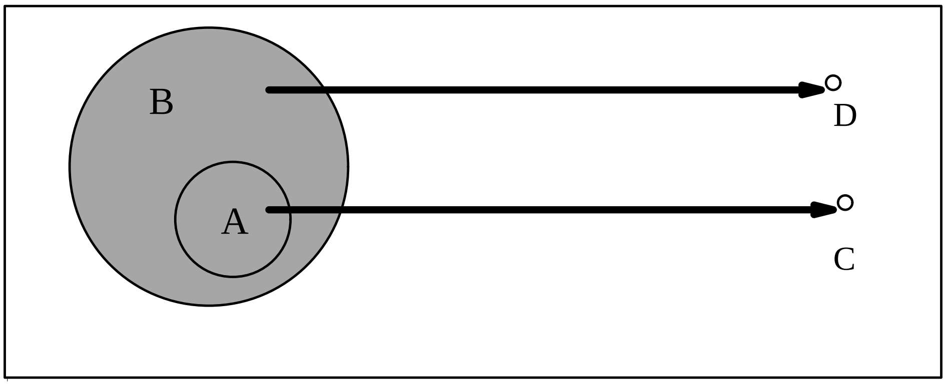

Figure 3: Cancellation of Inheritance Case II

$A \supset B$. $F_A \subset F_B$. Only the feature sets of the respective concepts are shown here. One arrow is drawn in place of pairwise connections.

-

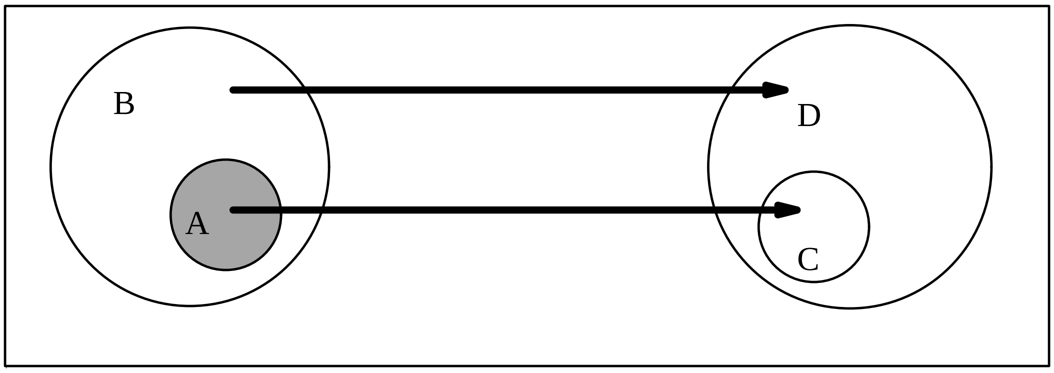

Figure 4: Cancellation of Inheritance Case III

$A \supset B$. $F_A \subset F_B$. $C \supset D$. $F_C \subset F_D$. Only the feature sets of the respective concepts are shown here. One arrow is drawn in place of pairwise connections.

-

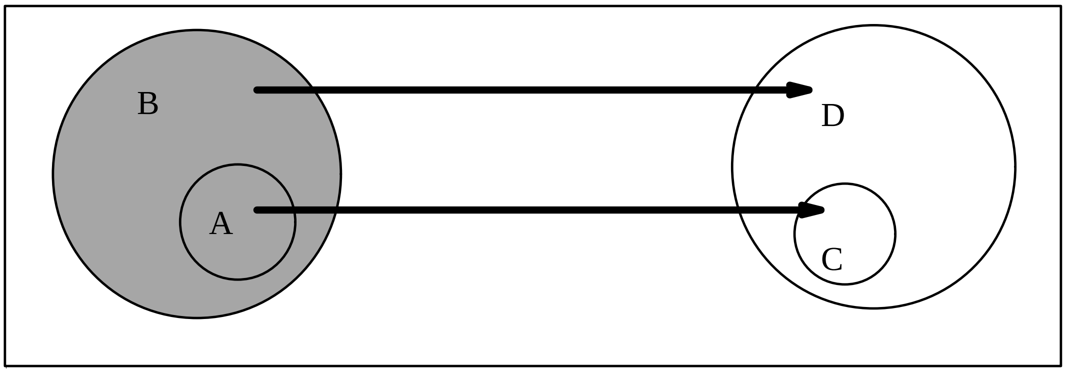

Figure 5: Cancellation of Inheritance Case IV

$A \supset B$. $F_A \subset F_B$. $C \supset D$. $F_C \subset F_D$. Only the feature sets of the respective concepts are shown here. One arrow is drawn in place of pairwise connections.

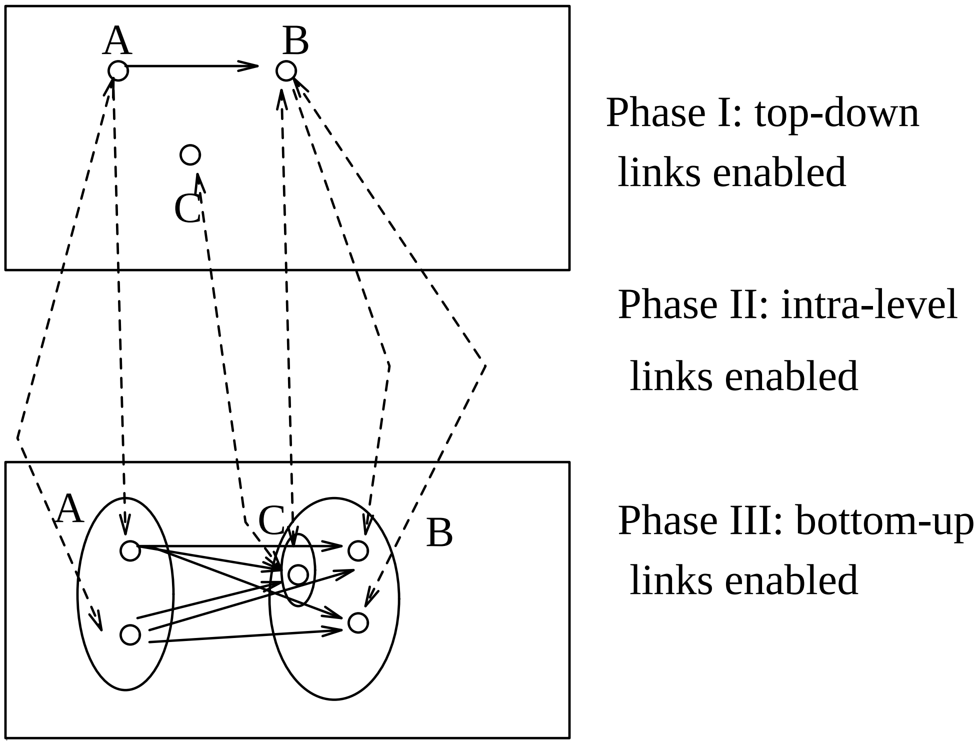

Figure 6: The CONSYDERR Architecture

The top box is CL and the bottom box is CD. In CL, a concept is represented by one node. In CD, a concept is represented by a cluster of nodes. Corresponding nodes are connected via top-down and bottom-up links.

-

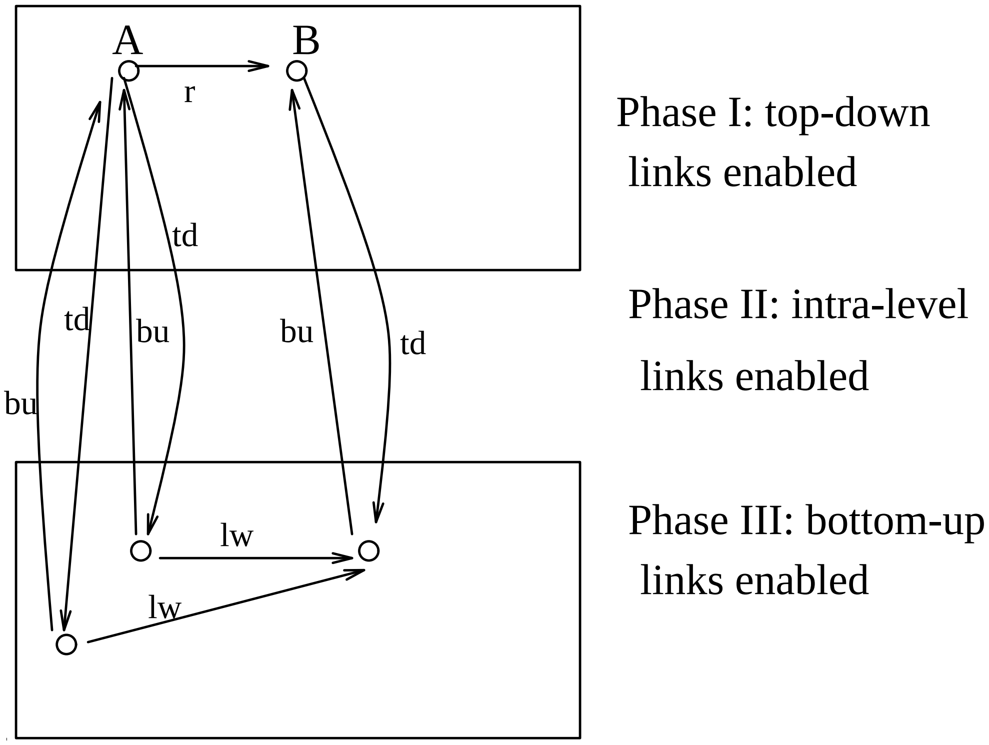

Figure 7: A Generic Model

The top box is CL and the bottom box is CD. r is the link weight (the strength of a rule) in CL. lw is the link weight in CD, which diffusely replicates the CL link. Across CL and CD, bu is the bottom-up weight and td is the top-down weight. The operation of the model is divided into three phases.

-

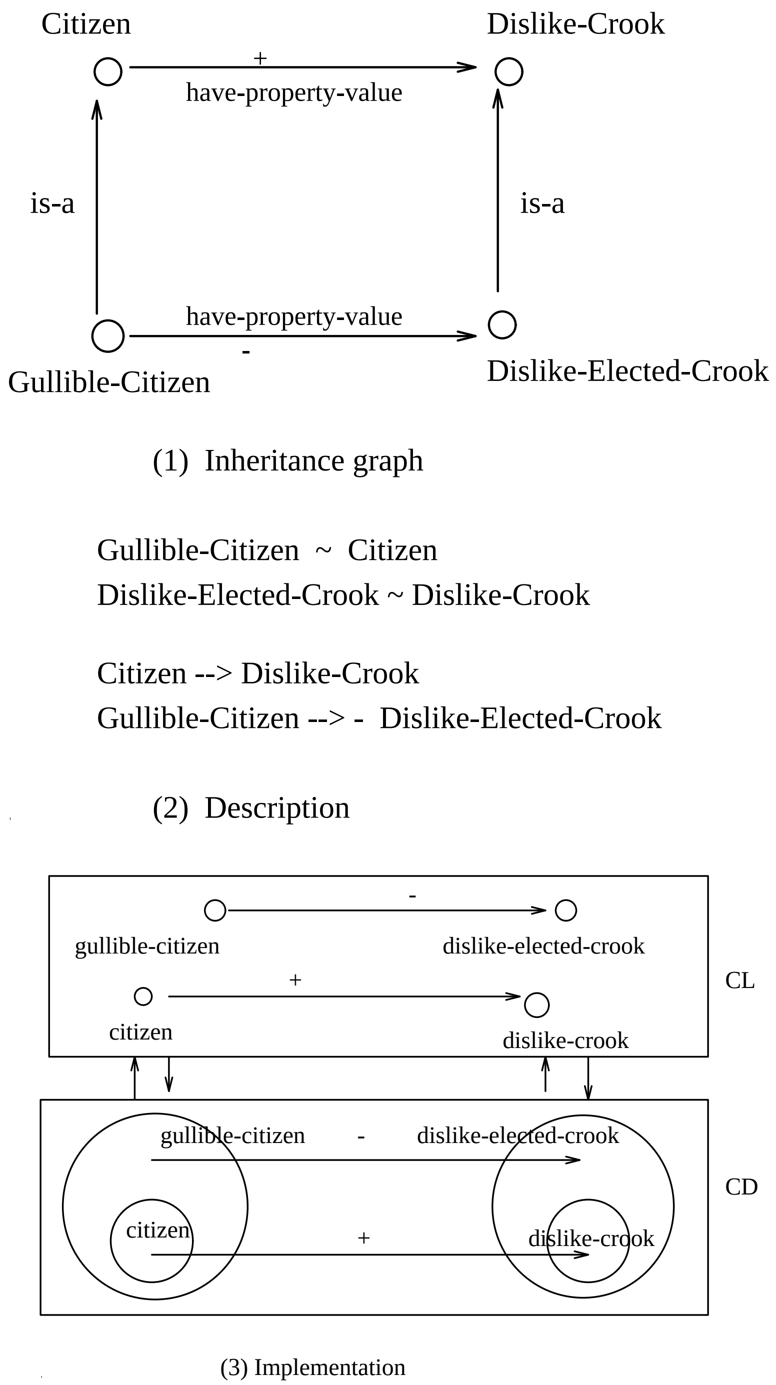

Figure 8: An Inheritance Graph, Its Description and Its Implementation with Two Levels

In the figure, *gullible-citizen* $\subset$ *citizen* and $F_{gullible\text{-}citzen} \supset F_{citizen}$. *dislike-elected-crook* $\subset$ *dislike-crook* and $F_{dislike\text{-}elected\text{-}crook} \supset F_{dislike\text{-}crook}$. "+" represents a positive connection (with a weight 1). "-" represents a negative connection (with a weight -1). In CD, one arrow is drawn in place of pairwise connections.

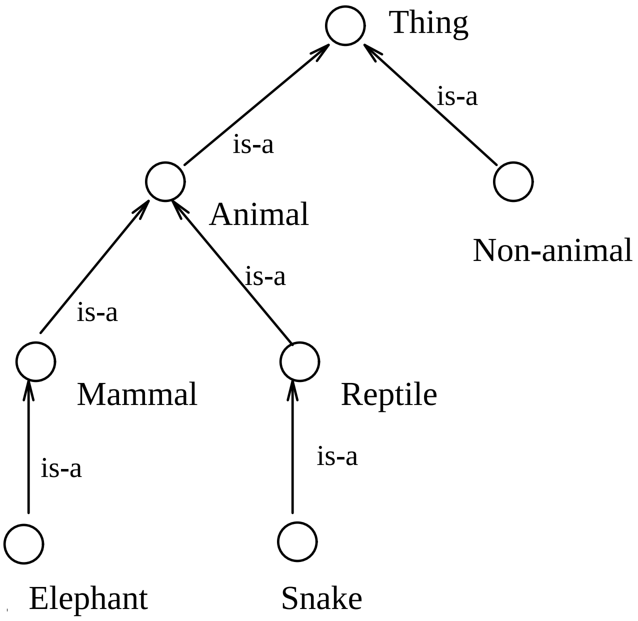

Figure 9: An Inheritance Graph with Long Chains of *is-a* Links

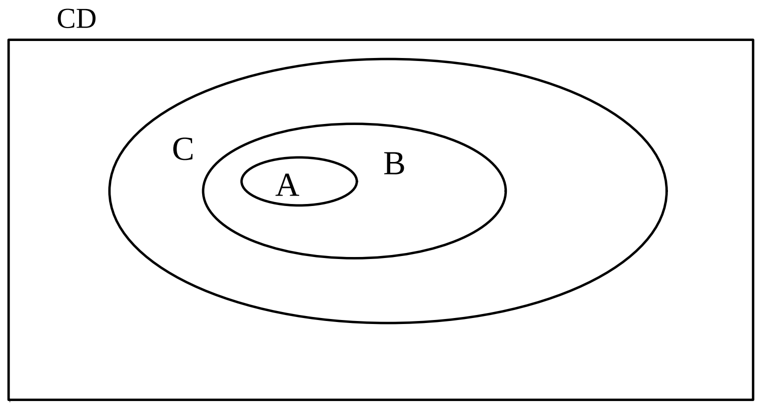

Figure 10: Implementing Chains of *is-a* Links: C is-a B and B is-a A

($A \supset B \supset C$ and $F_A \subset F_B \subset F_C$)

-

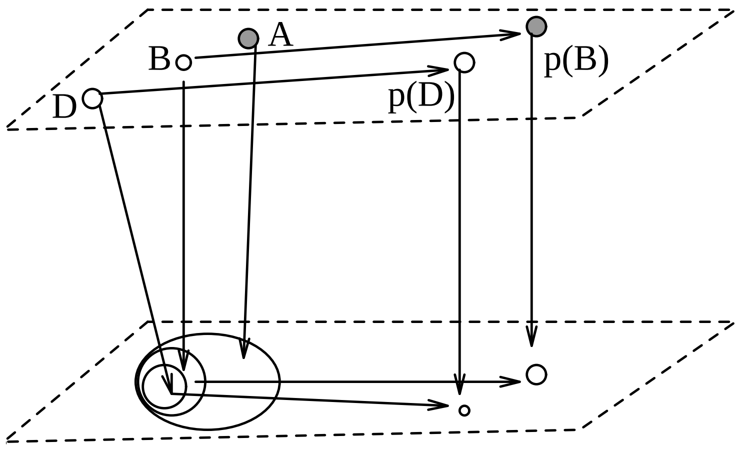

Figure 11: Inheritance Case 1

$A \subset B \subset D$. $F_A \supset F_B \supset F_D$. $p(x)$ denotes a property value of x. Arrows in CD represent pairwise links across two sets.

-

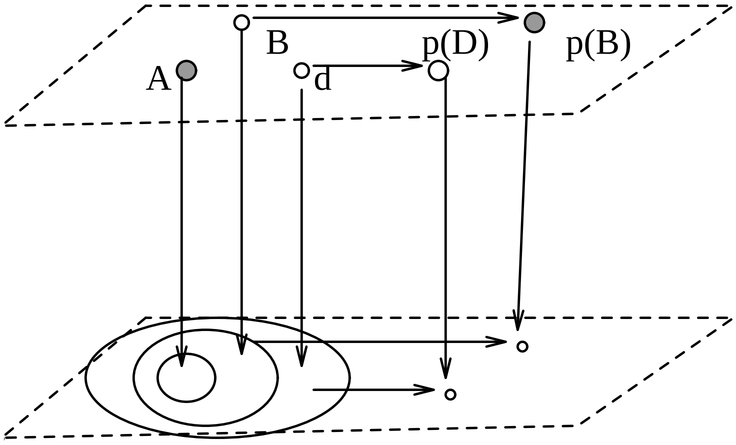

Figure 12: Inheritance Case 2

$A \supset B \supset D$. $F_A \subset F_B \subset F_D$.

-

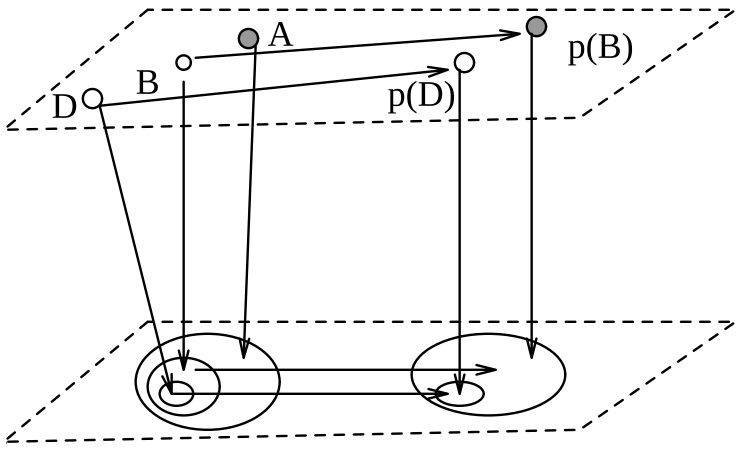

Figure 13: Inheritance Case 3

$A \subset B \subset D$. $F_A \supset F_B \supset F_D$. $p(B) \subset p(D)$. $F_{p(B)} \supset F_{p(D)}$.

-

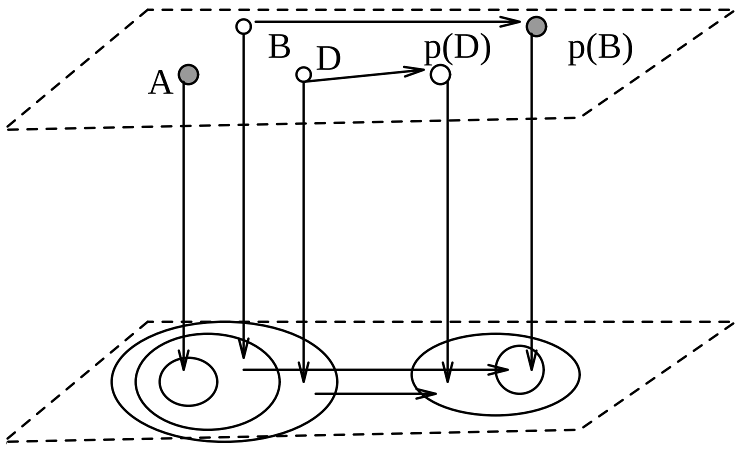

Figure 14: Inheritance Case 4

$A \supset B \supset D$. $F_A \subset F_B \subset F_D$. $p(B) \supset p(D)$. $F_{p(B)} \subset F_{p(D)}$.

-

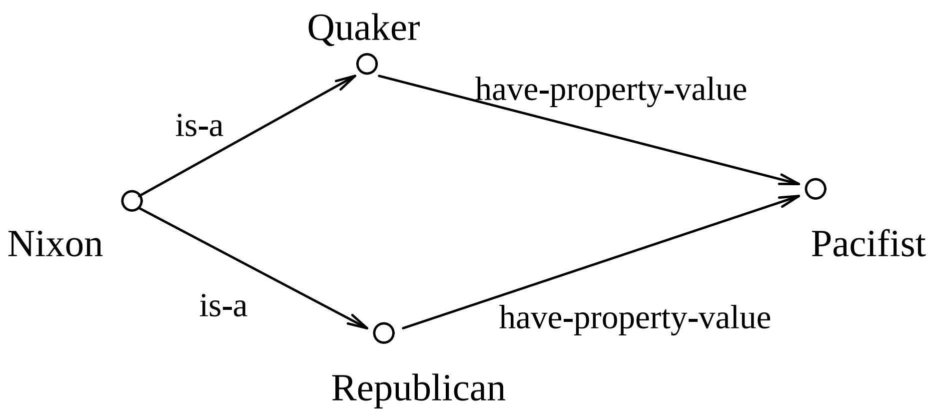

Figure 15: The Nixon Diamond: A Typical Multiple Inheritance Problem