ELSEVIER

Artificial Intelligence 103 (1998) 237–271

Artificial Intelligence

Inheritance comes of age: applying nonmonotonic techniques to problems in industry

Leora Morgenstern [1]

IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532, USA

Abstract

Nonmonotonic reasoning is virtually absent from industry and has been so since its inception; the result is that the field is becoming increasingly marginalized within AI. We argue that this is largely because researchers in the area focus exclusively on commonsense problems which are irrelevant to industry and because few efficient algorithms or tools have been developed. A sensible strategy is thus to focus on industry problems and to develop solutions within tractable subtheories of nonmonotonic logic.

We examine an example of nonmonotonic reasoning in industry—inheritance of business rules in the medical insurance domain—and show how the paradigm of inheritance with exceptions can be extended to a broader and more powerful kind of nonmonotonic reasoning. This is done by introducing *formula-augmented semantic networks* (FANs), semantic networks which attach well-formed formulae to nodes. The problem of inheriting well-formed formulae within this structure is explored, and an algorithm is given and discussed. Finally we discuss the underlying lessons that can be generalized to other industry problems. © 1998 Published by Elsevier Science B.V. All rights reserved.

Keywords: Artificial intelligence; Inheritance; Inheritance networks; Semantic networks; Expert systems; Nonmonotonic reasoning; First-order logic; Rule-based systems; Insurance; Medical insurance; Benefits inquiry; Applications; Commercial applications; Medical AI; Nonmonotonic logic; Nonmonotonic techniques

1. Introduction

Nonmonotonic logic, the formalization of plausible reasoning, is invisible and virtually nonexistent in industry. It is in a worse position, in this respect, than most other areas of

[1] Email: leora@watson.ibm.com.

PII: S0004-3702(98)00073-3

Artificial Intelligence. It is true that AI researchers have long accustomed themselves to the huge gap between AI hype, which promises great things (e.g., housekeeping robots), and AI reality, which delivers much less (robots that have a hard time collecting tennis balls).

Yet AI as a whole is quite visible in industry and the marketplace. Although AI has delivered less than was anticipated one or two decades ago, there is enough going on: expert systems are used in medical diagnosis, circuit configuration, and financial applications; dictation systems for restricted domains are on the market. Unfortunately, such examples do not include applications that are based on nonmonotonic reasoning. [2]

The absence of nonmonotonic reasoning from the commercial world may have been small cause for concern in the early eighties, when AI showed endless promise, research money was plentiful, and the field was very young. As we approach the end of the nineties, however, we have reason to worry. Funding has shrunk, and there is little tolerance for research programs that do not promise—and deliver—practical results in the foreseeable future. There is the real danger—if nonmonotonic reasoning and industry continue to inhabit separate worlds—that nonmonotonic reasoning will become marginalized and isolated; that funding for nonmonotonic research will dry up to the point where we are no better off than researchers in mathematics (or worse, philosophy); that as a result the field will shrink, leaving only a few die-hards talking to one another instead of a vibrant research community which is tackling one of the hardest problems in reasoning.

I do not feel happy about writing that last paragraph. These are the sort of gloomy prognostications one is used to hear from people outside the field of nonmonotonic logic. When I have heard these sentiments in the past, I have usually put them down to some combination of *schadenfreude* and the resentment of practitioners toward theorists. This is not the case here. On the contrary: I am a member of the nonmonotonic reasoning community, and I am concerned about the current state of nonmonotonic research. But this is concern rather than pessimism: I believe that we can stop the field from being marginalized, and strengthen nonmonotonic reasoning as a central part of mainstream AI. In order to do so, we must find some way to make nonmonotonic reasoning useful to industry. We need to understand why nonmonotonic reasoning and industry are so far apart and to figure out how to bridge the gap. We also need to see if there are any examples of nonmonotonic reasoning in industry, and to study these examples for lessons to generalize and mistakes to avoid in the future.

The paper is accordingly structured as follows. Section 2 discusses the reasons underlying the gap between nonmonotonic reasoning and industry and suggests possible strategies to bring these two areas closer together. Sections 3 through 5 examine in detail an example of a nonmonotonic system that I developed for industry—specifically, a benefits inquiry system for the insurance industry. The system extends the standard paradigm

[2] It might be argued that fuzzy logic [43], which also claims to capture plausible reasoning, has been used in industrial applications. Without commenting on the merits of this argument, we merely note that the fields of fuzzy logic and nonmonotonic logic are quite separate in both philosophy and community; thus, the presence of fuzzy logic in industry says little about the field of nonmonotonic reasoning. It is also true that logic programming (such as the logic programming language Prolog [3]) is used in industry, though it is generally not used to capture nonmonotonicity. We discuss logic programming in Section 2.

of nonmonotonic inheritance by adding general well-formed formulae to the nodes in an inheritance network. We explore how one can *inherit* well-formed formulae using this structure, examine the ways in which nonmonotonic techniques provide the benefits inquiry system with the necessary expressiveness and reasoning ability, and discuss how we can avoid the complexity problems that beset nonmonotonic reasoning systems. We subsequently consider generalizations of the system, and finally examine the lessons which can be applied in general to joining nonmonotonic reasoning and the commercial world.

2. Analyzing the gap between nonmonotonic reasoning and industry

2.1. Reasons for the gap

Nonmonotonic reasoning was first introduced in the late 1970s [25,27,34] in order to formally capture plausible or default reasoning.[3] Plausible reasoning includes reasoning with exceptions, reasoning with default rules (rules that talk about a typical member of a class, rather than all members of a class), reasoning with incomplete information, and jumping to conclusions and retracting such conclusions if they are proved to be wrong. The aim in those early days was to construct a logic that is more powerful than classical first-order logic and to aid in developing programs that could reason more flexibly and fluently than the programs then available. Nonmonotonic logic was supposed to make AI easier. As such, it could have been reasonably expected that nonmonotonic logic would become a tool of software engineers (as is the case, for example, with object-oriented programming). In fact, this has not happened: two decades later, open activity in nonmonotonic research is found only in academia and tolerant research labs.

What went wrong? The answer, in a nutshell, is that nonmonotonic reasoning is nowhere near ready to handle industrial-strength problems. Researchers freely admit this, and have been freely admitting it for the last twenty years. After this length of time, the admission is in itself cause for concern. Much of AI and all of industry is about getting things done. Confession will not save us here: we need to determine why nonmonotonic reasoning has not helped get things done. Three reasons come to mind.

(1) *Nonmonotonic research has focussed almost exclusively on toy problems of commonsense reasoning.* The canonical Tweety problem—inferring that Tweety can fly from the facts that Tweety is a bird and that birds typically fly; and retracting that conclusion upon discovering that Tweety is a penguin—is still one of the benchmark problems that researchers seriously tackle when they develop new nonmonotonic logics or modify old ones.[4]

[3] These papers, as well as other classic papers in the field, are collected in [12].

[4] While all existing nonmonotonic logics, as far as I know, can solve the Tweety problem, simple variations on this problem are beyond some well-known nonmonotonic systems. For example, a nonmonotonic reasoning system based on consequence relations (as in [23]) cannot infer that Tweety can fly from the facts that Tweety is a robin, robins are birds, and birds typically fly. (The problem is that chaining is in general not permitted in consequence-relation logics.) A relatively simple fix [10] results in a system that can solve this variant Tweety problem; the point, however, is that it is far from obvious that nonmonotonic systems can solve very simple reasoning problems.

Indeed, nonmonotonic theories have trouble solving a host of other toy problems such as the well-known Yale Shooting problem [17], which involves predicting that if one loads a gun, waits, and then fires the gun at a turkey, the turkey will die. [5]

It can be argued with a good deal of justification that commonsense reasoning is one of the more difficult areas of intelligent behavior for AI to model [5] and that sneering at research on the Tweety and Yale Shooting problems merely reflects a lack of understanding of the difficulties of the underlying issues. It may very well be that toy problems of commonsense reasoning are more difficult than industry problems. Nonetheless, one can hardly be surprised that industry has been reluctant to invest in technology based on research that is stymied by the likes of the Yale Shooting problem.

(2) *In fact, industry is primarily concerned with problems which appear to have very little to do with commonsense reasoning.* Examples include diagnosing bacterial infections, determining where oil is likely to be found, and predicting variations in the stock market. Nonmonotonic researchers typically ignore these problems, preferring instead to work on problems of commonsense reasoning, as discussed above. The result is that these researchers have very little to offer industry. The irony is that one can plausibly argue that nontrivial nonmonotonic reasoning—and indeed much commonsense reasoning—is present in a wide variety of industry problems. For example, virtually any prediction task must be done in the absence of complete information about one's situation, and must use causal rules which have exceptions; this suggests that some form of nonmonotonic reasoning (such as nonmonotonic temporal reasoning) is needed.

Industry seems to offer fertile ground for nonmonotonic researchers. The problem is that industry remains uncharted territory for the nonmonotonic community. The result is that this community has not yet demonstrated that it is capable of solving any industry problems. It is all very well to argue that the seemingly hard problems facing industry are in reality easier to solve than the deceptively simple commonsense problems upon which nonmonotonic research focusses, but this argument, however cogently formulated, must be buttressed with solid solutions to problems in industry.

(3) *Nonmonotonic reasoning techniques have not scaled up to industry.* Even if the nonmonotonic community were to start working on a problem directly relevant to industry, and to come up with a good solution, nonmonotonic reasoning is crippled by decidability and tractability problems, and by a lack of good tools. Specifically, nonmonotonic predicate logic is in general undecidable and even simple classes of propositional nonmonotonic logics are intractable. For example, determining whether a formula is in the extension of a propositional default theory is in general Σ_2^P-complete [13].

There are some bright spots in this otherwise dark picture. There are relatively efficient polynomial algorithms for a particular type of nonmonotonic reasoning known as *inheritance with exceptions* due to Horty et al. [20], and Stein [41], which is the

[5] It is assumed that firing a loaded gun at a turkey always results in the death of the turkey. The difficulty arises in predicting that a gun that was loaded at one moment will remain loaded at the next. More precisely, the Yale Shooting problem is a particular instance of the multiple extension problem, which arises when two or more default rules conflict. In this case, the default rule that guns typically stay loaded from one moment to the next conflicts with the default rule that turkeys typically stay alive from one moment to the next. Early papers on the Yale Shooting problem can be found in [12]; for recent analyses, see [30,39].

foundation upon which the work in this paper is built.[6] Inheritance with exceptions is the nonmonotonic extension of inheritance, a simple form of reasoning with subclasses and superclasses that dates back to Aristotle and the syllogism [22]. This extended form of inheritance allows one to posit exceptions to the general behavior of classes and to reason with those exceptions. It easily handles Tweety-style problems. (It cannot, however, handle the Yale Shooting problem unless that problem is reformulated in a somewhat unnatural manner. The problem is that temporal reasoning cannot generally be represented as a subclass-superclass problem. This highlights the major disadvantage of standard inheritance with exceptions, which will be remedied to a large degree in this paper: it is ill suited for anything but taxonomic reasoning.) In addition to work in this area, there are many promising results in the area of logic programming, which is sometimes viewed as a subfield of nonmonotonic reasoning. For example, computation of solutions for theories that are propositional, nonrecursive, and in Horn form is $O(n)$ [7]; computation of a unique solution set for *courteous logic programs*, a restricted form of prioritized defaults, is $O(n^2)$ [15].

At least in the case of inheritance, however, these positive theoretical results are weakened by the lack of corresponding industrial-strength tools which implement these algorithms. For example, there are no commercially available tools for inheritance with exceptions, despite the fact that efficient algorithms have been known and published for a decade. Anybody who wishes to use the technology in industry must build the code from scratch. In an age and an industry where tools have become a sine qua non, the lack of a good tool can freeze any possibility of using a nonmonotonic technique.

This was, indeed, my experience in developing the inheritance system described in Sections 4 and 5. Despite the fact that it was clear that standard inheritance would not do the job, and that a system as least as powerful as nonmonotonic inheritance was needed, the fact that tools that performed standard inheritance existed, while tools that performed nonmonotonic inheritance did not exist, caused many involved with the project to argue that the existing tool be used. It is a mark of the Dilbertian nature of the software and consulting industry today that using an existing tool to get the job done badly but quickly is considered preferable to building the correct tool and doing the job slowly but well.

The importance of good tools is underscored by the relative success in industry of logic programming. Logic programming offers a basic tool: the language Prolog, which supports simple nonmonotonic reasoning. Logic programming techniques are widely used in practical applications (see, e.g., the papers in [33]). Unfortunately, the success of logic programming does not mean that nonmonotonic reasoning is present in industry. Logic programming is often used because of its elegance and its usefulness as a specification language; its nonmonotonic reasoning abilities are rarely exploited. Perhaps this is because the logic programming community does not always share the nonmonotonic community's interest in modelling and solving nonmonotonic reasoning problems. Indeed, the work on using logic programming's nonmonotonic capabilities for practical applications comes from those researchers at the intersection of logic programming and nonmonotonic

[6] These algorithms are tractable. The tractability of Horty et al.'s algorithms is discussed in [38]; the complexity depends on the kind of chaining involved in path construction. The version presented in [18], Section 2.1, leads to tractable algorithms. Stein's algorithm is $O(n^5)$.

reasoning who also have a particular interest in applications (e.g.,[14,15,36]). It is interesting to note that in these cases, researchers have had to augment standard logic programming techniques with special-purpose nonmonotonic reasoning techniques such as conflict handling.

2.2. Closing the gap

If the field of nonmonotonic logic has remained entirely separate from industry because first, nonmonotonic research and industry focus on very different problems, and second, researchers have not yet developed efficient algorithms and/or tools, the strategy for integrating nonmonotonic reasoning with industry becomes clear.

First, researchers in nonmonotonic logic should familiarize themselves with problems in industry, select a set in which nonmonotonic reasoning appears to be important, and focus on those problems in their research. Second, researchers ought to actively design efficient algorithms for the tractable portions of nonmonotonic theories, and develop industrial-strength tools.

Ideally, these endeavors should be carried out simultaneously. That is, as solutions to nonmonotonic problems in industry are found, tools to implement these solutions should be developed. That is not essential, however; what is important is that both tasks get done.

At this point, relatively few industry problems have been solved using nonmonotonic techniques. The remainder of this paper will discuss one such problem and its solution. We will outline the problem, explain why nonmonotonic reasoning is necessary, present the nonmonotonic techniques used, and suggest how this method could be generalized to other problems in industry.

3. The problem: inheritance and inheriting rules in the medical insurance domain

3.1. The case in point: benefits inquiry

3.1.1. Problem description

Benefits inquiry is the process of querying an insurance company to determine one's benefits. In this paper, as well as in [31], the term benefits inquiry also refers to the process performed by insurance company employees in answering such queries. In the medical insurance industry, customers may wish to know whether a particular procedure is covered, the extent of the coverage, and the specific rules that limit coverage. Examples of questions that customers typically ask are:

> Will my son's tonsillectomy be covered?
> Can it be performed in an inpatient facility?
> How many days can I stay in the hospital after a standard delivery?

Benefits inquiry occurs frequently in the medical insurance industry and has become increasingly complex during the past few years: medical insurance companies today may have thousands of insurance products, each of which contains a myriad of services and regulations; these regulations themselves change frequently. The vast amount of

changing information is difficult to keep up with. In addition, there are many rules that have exceptions, and exceptions can be nested. For example, physical therapy might be generally limited to twenty visits per year, unless more visits are ordered in writing by a physician, but spinal manipulation, a type of physical therapy, might have a more generous limit (around thirty visits). The importance of exceptions suggests that some form of nonmonotonic reasoning would be useful.

Several years ago, I was asked to develop an expert system for benefits inquiry. (This was part of a comprehensive consulting engagement between IBM Research and a large medical insurance corporation to update major portions of their information management system.) The primary goal of the expert system was to aid customer service representatives (CSRs), the insurance company employees who answer customers' questions about their benefits.

3.1.2. What had been done

Customer service representatives have traditionally relied on a variety of sources for information, including basic charts, detailed manuals, and implicit knowledge gained through common sense and on-the-job experience.

As insurance products have proliferated, there have been attempts to harness computing power to help with the benefits inquiry task. Often, such efforts are limited to placing charts and manuals on-line, which at least guarantees that all CSRs are looking at the same materials at the same time, and facilitates quick updating of these materials. Other efforts include text-based and code-based systems, discussed below.

Text-based systems. Text-based systems allow for some search and indexing of subject areas. The medical insurance company for which we consulted had a rudimentary text-based system. Information is divided into chunks or subject areas. A piece of text is associated with each subject area. Subject areas might include preventive care, immunizations, and maternity. The text associated with preventive care lists the different types of preventive care available, such as routine physicals and standard immunizations, as well as coverage rates and allowed frequency of services. Another screen may deal with one of the topics described in one screen; e.g., there may be a screen devoted to immunizations, a topic described in the preventive care screen. The CSR uses the system by pulling down a menu of topics, clicking on a topic, and reading the information on the associated screen which comes up.

The advantages to this system over on-line manuals are first, the system allows rudimentary search, and second, the information is partly organized in a modular fashion. This is helpful both for ensuring that information on a topic can be found in a single location and to ease updating. The disadvantages are: first, there is no reasoning; the system has merely pruned the amount of information to be read; second, the system does not make explicit the many interconnections between subject areas. For example, nothing in the system indicates a connection between the screens on immunization and preventive care. The CSR must reason, e.g., that the schedule rate for preventive care most likely applies to routine immunizations. The lack of connection between related screens makes updating the system difficult. If both the preventive care and immunization screens do have schedule rate information and this schedule rate changes, the individual modifying

the system must make changes to both screens. Third, there are a small number of screens relative to the number of types of questions a CSR may have to answer (there are literally thousands of services which a customer may inquire about).

Thus, the CSR may not get the level of information that she needs. This is not an artifact particular to this application but is rather due to the logistical difficulties of creating and continually modifying a large number of screens, and the unwieldiness of a menu with too many screens.

Making inquiries via codes. Many insurance companies have a code-based scheme for answering customers' questions. Whether automated or not, such schemes operate using a table-lookup methodology. When a customer calls to inquire whether a particular service is covered, he is asked to provide the CSR with a *procedural* (CPT) code, which represents the service which is to be performed, and a *diagnostic* (ICD-9) code, which explains why the service is to be performed. The CSR then feeds this pair of codes into the system, which responds with the information that the service is or is not covered, along with the appropriate cost of the service.

The advantage of code-based inquiry is that often customers' questions can be answered quickly and unambiguously. Moreover, since adjudication of claims is usually done using a code-based scheme, code-based inquiry helps ensure that the answer given by the CSR corresponds exactly to the benefit that the customer receives. However, there are several disadvantages. First, customers must know complete code information before they can call their CSRs; this slows down the inquiry process and is frustrating to customers. Second, the code-based scheme allows questions only at a detailed level of granularity. But customers often wish to ask general questions, such as `Are routine immunizations covered?` Third, code-based systems do not allow questions that rely on information other than the data represented by the codes. For example, code-based systems could not handle a question such as, `If a 44-year-old woman had a mammogram two years ago, will she be covered for a routine mammogram now?`, in which the treatment history of the patient is relevant. Fourth, updating code-based systems is an exceedingly difficult process. There are tens of thousands of CPT codes and tens of thousands of ICD-9 codes, and a large subset of the possible pairings of these codes must be considered.[7] Insurance companies typically employ scores of people whose function is tied solely to the code-based scheme.

3.1.3. Desiderata—toward replacing existing technology

We aimed to develop an expert system that supports benefits inquiry but avoids the problems of both the text-based and code-based systems. In particular, we wished to develop a system that:

- allows questions at varying levels of granularity;
- gives clear, unambiguous answers to commonly asked questions;
- allows representation of very large amounts of material;
- allows navigation around a large information space;

[7] The nightmarish prospect of updating millions of code pairs is somewhat mitigated by the fact that CPT and ICD-9 codes have their own structure and support some rudimentary abstraction.

- facilitates searching by topic;
- supports connections among related topics;
- supports easy updates and modifications.

The ability to modify is important because products change so frequently; an outdated benefits inquiry system is useless. Thus, the system had to be usable not only by CSRs but also by *policy modifiers* (PMs), the insurance company employees responsible for making changes within a particular insurance product, creating new products, and deleting old products.

3.2. Why inheritance with exceptions is useful

Much of the information about medical services is taxonomic in nature. For example, Spinal Manipulation is a type of Physical Therapy; Physical Therapy, Speech Therapy, and Occupational Therapy are all types of Therapy. Likewise, the Adult Primary Care Benefit is a subtype of the Primary Care Benefit. Coverage and accompanying restrictions are to a large extent inherited along taxonomic lines: Physical Therapy is covered, for example, because it is a subtype of Therapy, and Therapy is covered. On the other hand, there are exceptions: even though Drugs are covered by the Drugs Benefit, and OTC (over-the-counter) Drugs are a subclass of Drugs, OTC Drugs are not covered by the Drugs Benefit.

These observations suggested to us that we needed some sort of inheritance network, and indeed, an inheritance network with exceptions (this meant that we could not use a standard inheritance network such as KL-ONE [37] or K-REP [24]). An inheritance network with exceptions is of course not a purely taxonomic (treelike) structure; it is instead a directed acyclic graph, as can be seen in the simple example of the well-known penguin triangle: penguins are birds; birds fly; penguins do not fly. (There are two links emanating from the node representing penguins.) In fact, multiple inheritance frequently arises in this domain apart from the inheritance-with-exceptions issue, because certain services have multiple supertypes. For example, Genetic Testing is a subtype of both Diagnostic Services and of Family Planning Services.

3.3. Representing coverage information

There are two ways to encode coverage and exclusion information within a semantic network, depicted in Figs. 1 and 2. One way (Fig. 1) is to introduce, in addition to the subtype link, *covers* and *excludes* links which connect benefits and service nodes. Thus, for example, to represent the information that Surgical Services are *covered* by the Surgical Benefit, one places a covers link between Surgical Benefit and Surgical Services; to represent the information that Routine Endoscopy is *excluded* by the Surgical Benefit, one places an excludes link between Surgical Benefit and Routine Endoscopy. The covers and excludes links propagate along the taxonomy, so that, e.g., we can reason that Surgical Benefit covers Orthopedic Surgery, since Orthopedic Surgery is a subtype of Surgical Services. The propagation is relative to specificity considerations, so we do not use propagation to conclude that Routine Endoscopy is covered by the Surgical Benefit:

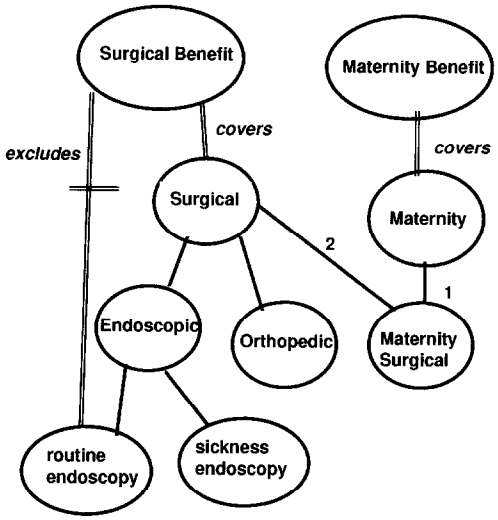

Fig. 1. One method of representing coverage information: covers and excludes links are added to the standard subtype and cancels links of inheritance networks. Solid lines represent is-a links; slashed solid lines (not present here) represent cancels links; double lines represent covers links; slashed double lines represent excludes links. This representation can be mapped to a standard inheritance network.

the path from Routine Endoscopy to Surgical Benefit along the excludes link is more direct than the path from Routine Endoscopy to Surgical Benefit along the covers link.

It is obvious that there is a close connection between this structure and the standard inheritance network with exceptions described in [20]. In fact, the second way of encoding covers and excludes information is simply to use a standard inheritance network with exceptions. One can easily do so by *reifying* each benefit as the set of services which are covered by that benefit (Fig. 2). Thus, to represent the information that Surgical Services are covered by the Surgical Benefit, one would have a node representing Surgical Services, a node representing the services *covered* by Surgical Services, and a subtype (is-a) link between these two nodes. The information that the benefit covers the service has been transferred from the covers link to the nodes and subtype link.

The close connection between these two representations can be made precise in the following way: replace each benefit node by a service node which represents all services covered by that particular benefit (so that, e.g., as described above, the Surgical Benefit node would be replaced by the node Services Covered by Surgical Benefit). Replace each covers link by a subtype (is-a) link, and each excludes link by a cancels link. One then uses a standard traversal algorithm for inheritance networks with exceptions [20,41] to determine whether or not services are covered.

Indeed, both representations are useful: the standard inheritance network with exceptions representation allows us to use well-documented algorithms without modification, while the explicit use of covers and excludes links is more accessible to CSRs and PMs. The implementation of the benefits inquiry system (discussed in detail in [31]) used both representations: the internal representation used the standard inheritance network; while

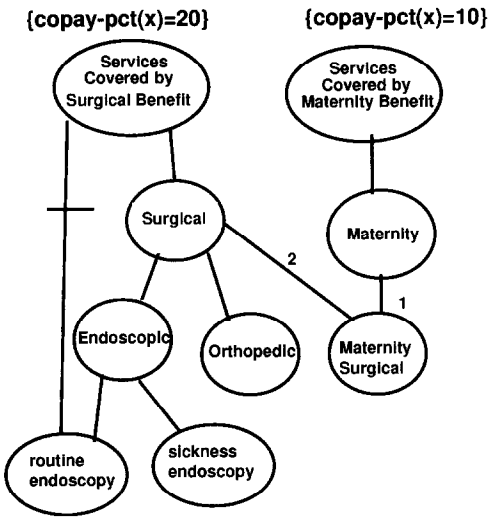

Fig. 2. Representing coverage information in a standard nonmonotonic inheritance network. Lines represent is-a links; slashed lines represent cancels links. Medical benefits are reified as services covered by a particular benefit.

the external representation, the graphical user interface manipulated by CSRs and PMs, used explicit covers and excludes links.

The importance of the cancels links in the standard inheritance representation points out that much of the relevant information for benefits inquiry is nonmonotonic. It is rarely the case that a class of services is entirely covered or excluded by some benefit; there are almost always exceptions. This nonmonotonicity is hardly surprising; proponents of nonmonotonic logic [34] have argued since its inception that nonmonotonic reasoning is common in everyday reasoning (including the business world). What is surprising is how rarely nonmonotonicity has entered commercial applications, even in well-understood forms such as inheritance with exceptions. Nonmonotonicity is also present in this application in the regulations that apply to a node; we discuss this further in Section 5.

3.4. Why inheritance with exceptions is not enough

While much of the information in the medical insurance domain—in particular, the relations among benefits and services—is taxonomic, a large chunk of information, specifically business rules, does not seem to be taxonomic in nature. This information is central to the task of benefits inquiry, since CSRs must determine which regulations apply to a service.

The problem with representing business rules in an inheritance hierarchy can best be appreciated by examining several rules. Some rules lend themselves to representation within a semantic network. Consider the rule:

There is a copay of 20% for diagnostic services.

To represent this rule using the standard inheritance network model, one could have a node representing the services which have a 20% copay, and a subtype link between the Diagnostic Services node and this node.

On the other hand, a more complex rule such as

> Patients in Drug Rehabilitation programs lose all rehab benefits for a year if they are noncompliant

cannot be so easily represented. One could posit a node that represents the services which have the property that if patients are noncompliant with respect to that service, then they lose all benefits for a year, and then have a subtype link between the Drug Rehab Services node and this node. But such a node appears quite artificial and outside the spirit of a semantic network, where nodes are supposed to represent easily understood concepts.

The fact that much of the domain knowledge is not taxonomic in nature means that we must go outside of the standard structure of an inheritance network with exceptions. On the other hand, it is desirable to build on the inheritance network structure: first, because inheritance with exceptions already solves part of the benefits inquiry problem; second, because inheritance with exceptions is one of the few efficient nonmonotonic techniques. Furthermore, there is an obvious connection between nontaxonomic and taxonomic knowledge in this domain. In particular, business rules often apply to particular services—i.e., to nodes in the network. Building on the existing network makes this connection explicit.

The next section suggests an extension to the network structure; Section 5 explores the process of benefits inquiry in this context.

4. The solution: integrating taxonomic and nontaxonomic information

4.1. Definition of a FAN

We wish to introduce a knowledge structure that is capable of representing both taxonomic and nontaxonomic information. The aim is to represent the taxonomic information in a standard inheritance network with exceptions and to attach the nontaxonomic information to the network in some way. For the medical insurance domain, we would like to represent the fact that business rules generally apply to specific medical services and benefits by attaching business rules to nodes in the network. To do this, we introduce the concept of a **F**ormula-**A**ugmented semantic (or inheritance) **N**etwork (FAN), an inheritance network in which sets of logical formulae may be attached to nodes. In the medical insurance domain, the logical formulae usually represent business rules, but they could also represent other sorts of information, including lists or tables (e.g., price lists, lists of preferred providers).

Formally, a FAN is a tuple $(\mathcal{N}, \mathcal{W}, \mathcal{B}, \mathcal{L}1, \mathcal{L}2, \mathcal{O})$, where:

- $\mathcal{N}$ is a set of nodes. In the medical insurance domain, a node represents some set of medical services; e.g., Physical Therapy represents the set of physical therapy services.
- The set of wffs $\mathcal{W}$ consists of well-formed formulae of a sorted first-order logic.

- The background $\mathcal{B}$ is a (possibly empty) set of wffs of first-order logic, intuitively representing the background information that is true. In the medical insurance domain, it includes all rules that are true of all medical services and benefits. It may also include patients' medical records and pay scales. In general, it consists of nontaxonomic information that is too general to attach to a specific node in the network.
- $\mathcal{L}1$ is the set of links on nodes, as described in Section 4.1.2.
- $\mathcal{O}$, the ordering on links, gives a preference on links. This is useful for nonconflicting multiple-path inheritance, since it allows us to prefer one path over another.
- $\mathcal{L}2$ is the set of links connecting nodes and sets of wffs. If N is a node and W is a set of wffs, $N \rightarrow W$ means that the set of wffs W is attached to node N. Intuitively, this means that each wff of W is typically true at N.

Intuitively, in this domain, each instance of a FAN represents a particular product of the medical insurance corporation.

The elements of the tuple are described in detail below. The description of $\mathcal{L}1$, which is part of any standard inheritance network, may be safely skipped by those familiar with the work of Horty et al. [20]. $\mathcal{W}$, $\mathcal{B}$, $\mathcal{L}2$ and $\mathcal{O}$, however, are unique to FANs.

4.1.1. Nodes

In the medical insurance domain, a node represents some set of medical services. In Fig. 2, the node Surgical represents the set of surgical services. A node may represent a set of services that are covered by a particular benefit. For example, in Fig. 2, the node Services Covered by Surgical Benefit represents the set of all medical services that are covered by the Surgical Benefit.

4.1.2. Wffs

The set of wffs $\mathcal{W}$ consists of well-formed formulae of a sorted first-order logic. Wffs typically represent business rules. An example of a wff in the medical insurance domain is

$$\begin{aligned}
&\forall e1, e2, e3, m, b, i1, i2 \\
&\big(\mathit{delivery\text{-}event}(e1, m, b) \wedge \mathit{hospital\text{-}event}(e2, b) \\
&\quad \wedge \mathit{assoc\text{-}hosp\text{-}stay}(e2, e1) \wedge \mathit{newborn\text{-}exam\text{-}ev}(e3, b) \\
&\quad \wedge \mathit{time}(e2, i1) \wedge \mathit{time}(e3, i2) \wedge \neg\, \mathit{subinterval}(i2, i1)\big) \\
&\qquad \supset \neg\, \mathit{covered\text{-}as}(e3, \mathit{newborn\text{-}exam})
\end{aligned}$$

where $e1$, $e2$, and $e3$ are variables ranging over events; $i1$ and $i2$ are variables ranging over time intervals; and b and m are variables ranging over babies and mothers, respectively.

This is the first-order translation of the business rule

> The initial newborn exam must be performed during the mother's hospitalization for delivery.

4.1.3. Background information

The *background* or *background context* $\mathcal{B}$ is a (possibly empty) set of wffs of first-order logic. $\mathcal{B}$ includes all rules that are true of all medical services and benefits. [8] For example, the background context may include a wff which states that all medical services must be provided by licensed medical professionals in order to qualify for reimbursement. $\mathcal{B}$ may also include information about particular individuals, such as a patient's medical records and the name of her primary care physician, as well as pricing charts for drugs and pay scales for medical professionals. As these examples make clear, although the information at $\mathcal{B}$ is considered to be global in the sense that it applies to every node in the network, it may also be subject to frequent change.

4.1.4. Links

Links are relations on objects. There are two sorts of links: links joining nodes and links joining nodes to sets of wffs.

$\mathcal{L}1$: *links on nodes.* The standard positive and negative is-a links relate nodes. (Negative is-a links will also be referred to as cancels links.) Positive and negative is-a links may be *strict* or *defeasible*. If x and y are nodes, we write $x \Rightarrow y$ and $x \not\Rightarrow y$ to represent, respectively, the strict positive and strict negative is-a links; we write $x \rightarrow y$ and $x \not\rightarrow y$ to represent, respectively, the defeasible positive and defeasible negative is-a links. Intuitively, $x \Rightarrow y$ means that all x's are y's; $x \not\Rightarrow y$ means that all x's are not y's (no x's are y's); $x \rightarrow y$ means that typically, x's are y's; $x \not\rightarrow y$ means that typically, x's are not y's. If $x \Rightarrow y$ or $x \rightarrow y$, we say that x is a *child* of y or that y is a *parent* of x.

Our account of paths is based on Horty [18]. A path is a restricted sequence of positive and/or negative links. We may define paths recursively in the following manner:

- There is a path from x to y iff there is a positive path from x to y or there is a negative path from x to y.
- If $x \rightarrow y$ or $x \Rightarrow y$ (respectively, $x \not\rightarrow y$ or $x \not\Rightarrow y$), then there is a positive (respectively, negative) path from x to y.
- If there is a positive path from x to y, and $y \rightarrow z$ or $y \Rightarrow z$ (respectively, $y \not\rightarrow z$ or $y \not\Rightarrow z$), then there is a positive (respectively, negative) path from x to z.
- If there is a negative path from y to z and $x \rightarrow y$ or $x \Rightarrow y$, then there is a negative path from x to z.

Note that positive paths can have only positive is-a links; negative paths can have just one negative link at the very end of the path. Paths that contain only strict links are called *strict paths*; paths that contain only defeasible links are *defeasible paths*; paths that contain both strict and defeasible links are called *mixed paths*. For ease of presentation, this paper will be concerned almost exclusively with defeasible paths; extensions to strict and mixed paths are straightforward. [9] The notation $\pi(x, \sigma, y)$ (respectively, $\pi'(x, \sigma, y)$) represents a positive (respectively, negative) path from x to y through the path σ. We extend this notation so that $\pi(x, \sigma)$ (respectively, $\pi'(x, \sigma)$) represents a positive (respectively,

[8] If we wish, we may think of the wffs in $\mathcal{B}$ as being included in the set of rules at each node.

[9] Note also that in all examples, nodes represent sets of objects, rather than individual objects. This is done to simplify the exposition; the extension to individuals is straightforward.

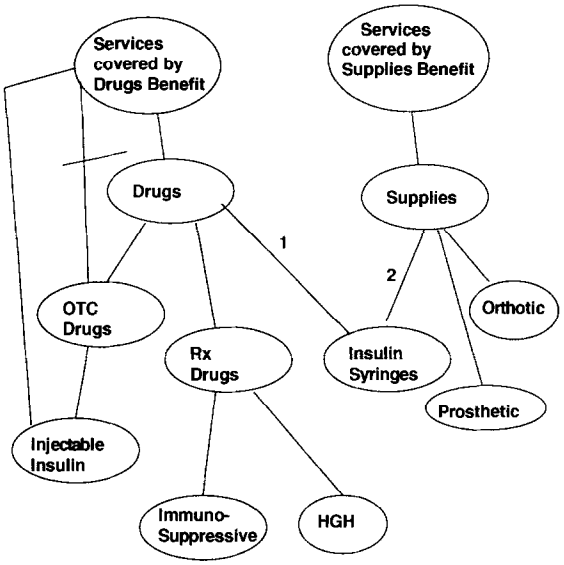

Fig. 3. Parts of the Drugs and Supplies portions of the medical insurance network.

negative) path from x to the last point of σ going through the path consisting of all but the last point of σ. The function *endpoint*(σ) will be used to designate the last point on the path σ.

Often, there will be both positive and negative paths between two nodes. To determine which path to choose, we follow the analysis of Touretzky [42] and Horty [18]. [10] Given a *context*—an inheritance network Γ and a set of paths Φ (intuitively, a set of paths arising out of the network), a path (of length greater than 1) is *inheritable* or *undefeated* if it is *constructible* and neither *preempted* nor *conflicted*. Briefly, a path is *constructible* in a context if it can, recursively, be built out of the paths in the network; a path is *conflicted* in a context if there is a path of opposite sign in the context with the same starting and ending points; a path is *preempted* if there is a conflicting path with more direct information about the path's endpoint (i.e., a direct link from an earlier point in the path). We also say that a path $\pi(x, \sigma, y)$ is preempted or conflicted if $\pi(x, \sigma)$ is preempted or conflicted. Intuitively, the undefeated or inheritable paths are those which "win out" over their rivals. If there is an undefeated positive (respectively, negative) path between x and y, we say $x \rightsquigarrow y$ (respectively, $x \not\rightsquigarrow y$).

Examples of paths can be seen in Fig. 3. There is a positive path from R_x Drugs to Services Covered by Drugs Benefit. There are both positive and negative paths from OTC Drugs to Services Covered by Drugs Benefit; however, according to Horty et al.'s [20] specificity criterion (or Touretzky's [42] inferential distance criterion), the positive path is preempted by the negative path. Thus, the negative path is undefeated.

[10] Horty's [18] exposition of nonmonotonic inheritance is a particularly clear and precise formulation of the ideas in [20].

$\mathcal{O}$: *ordering on links.* Multiple inheritance arises when there is an undefeated path from x to y, an undefeated path from x to z, and $y \neq z$. We call any such point x a *fork point* of the network, and the paths originating from a fork point *multiple paths.*[11] Inheritance networks in the literature have traditionally considered multiple inheritance only when these multiple paths have been initial segments of conflicting paths—as is the case in Fig. 3, where the multiple paths π(Injectable Insulin, OTC Drugs) and π(Injectable Insulin, Services Covered by Drugs Benefit) are initial segments of conflicting paths. We call such cases of multiple inheritance *conflicting-path multiple inheritance.*

In this paper, we will be interested in multiple paths even where they are *not* initial segments of conflicting paths, i.e., *nonconflicting-path multiple inheritance.* Examples of nonconflicting-path multiple inheritance can be seen in Fig. 3, where there are distinct nonconflicting paths between Insulin Syringes and Drugs, and between Insulin Syringes and Supplies.

In such cases, we may allow the prioritization of a particular path. We do this by first specifying a partial order on certain links of $\mathcal{L}1$. Specifically, if $x \rightarrow y_1$, $x \rightarrow y_2$, ..., $x \rightarrow y_n$ are elements of $\mathcal{L}1$ and are not initial segments of conflicting paths, we may place a partial order on the $x \rightarrow y_i$'s; this partial order is an element of $\mathcal{O}$.

The partial order on paths can then be defined recursively as follows:

- $x \rightarrow y$ is preferred to $x \rightarrow z$ if $((x, y), (x, z)) \in \mathcal{O}'$, where $\mathcal{O}'$ is an element of $\mathcal{O}$;
- $\pi(x, \sigma, y)$ is preferred to $\pi(x, \tau, z)$ if $\pi(x, endpoint(\sigma))$ is preferred to $\pi(x, endpoint(\tau))$;
- $\pi(x, \sigma, y)$ is preferred to $\pi(x, \sigma, z)$ if $\pi(endpoint(\sigma), y)$ is preferred to $\pi(endpoint(\sigma), z)$.

The partial order is depicted by placing ordinal numbers on links, as in Figs. 2 and 3. A lower number indicates a preferred element in the partial order. Thus, in Fig. 2, the link between Maternity Surgical and Maternity is preferred to the link between Maternity Surgical and Surgical; in Fig. 3, the link between Insulin Syringes and Drugs in preferred to the link between Insulin Syringes and Supplies.

If there are p fork points in the network, there are at most p partial orders (elements) in $\mathcal{O}$. The number of partial orders in $\mathcal{O}$, and the elements of these partial orders may be further restricted; in particular, we may not wish to specify priorities on links that are initial segments of conflicting paths, or may insist that the priorities be placed in a particular way.

$\mathcal{L}2$: *links between nodes and wff sets.* Let N be a node, and W a set of wffs. $N \rightarrow_w W$ means, intuitively, that each w of W is typically true at node N. $N \Rightarrow_w W$ means that each wff w of W is true at N; $N \not\rightarrow_w W$ means that each wff w of W is typically false at node N; $N \not\Rightarrow_w W$ means that each wff w of W is false at N. In practice, we rarely use the $\Rightarrow_w$ link. Intuitively, this link is used only when a formula is true at all nodes N_i such that $N_i \rightsquigarrow N$; i.e., we assume that there are no exceptions to N. This happens only rarely.

Since W can be a singleton, we allow the overloading of the $\rightarrow_w$, $\Rightarrow_w$, $\not\rightarrow_w$ and $\not\Rightarrow_w$ links so that they can refer to individual wffs as well as sets of wffs. If $N \rightarrow_w W$ or $N \Rightarrow_w W$, we will sometimes refer to W as the set of wffs at N or wffs(N). If w is a member of W, we say $at(w, N)$.

[11] As opposed to a *branch point*, which is downward.

The set of rules at each node in a network is constrained to be consistent with the background information $\mathcal{B}$.

4.2. FANs with added link types

FANs as described in the previous section have only (strict or defeasible) is-a and cancel links between nodes. This is not integral to the definition of a FAN. We can easily augment the definition of a FAN to include links such as covers and excludes (as discussed in Section 3.3), and specify the interaction between these links and the taxonomic links. Although we do not explore this approach further in this paper, we do note here an interesting feature of the FAN with covers and excludes links (called a modified FAN for the purposes of this discussion) for the insurance application described in this paper: the only taxonomic link needed is the standard (strict) is-a link. There is no need for defeasible is-a links, and thus no need for cancel links. Intuitively, the reason for this is that in the insurance application, most of the taxonomic links are genuine and not defeasible subset relations. The only place the defeasible link is needed is between a node and a "services-covered-by-benefit-x" node. But this link is replaced by the covers or excludes links in a modified FAN.

Indeed, this discussion highlights an interesting feature of FANs in this domain: cancel links always go directly up to the root node. Due to this feature, the algorithm to identify undefeated paths could be much simpler than the Horty et al.'s [20] or Stein's [41] algorithm, and in particular, would require only a simple upward traversal from a node until one reaches either a cancels link or a root. Our implementation, however, does not exploit this feature, since we wished to develop a system that was as general as possible. Thus, this paper discusses the more general case of FANs in which general inheritance with exceptions is allowed.

It should also be noted that the problem of wff-inheritance discussed in this paper applies not only to FANs but to modified FANs. This shows that the wff-inheritance problem can arise in a strict taxonomy as well as in an inheritance hierarchy with exceptions.

5. Inheriting well-formed formulae

5.1. The rules-inheritance problem

One of the defining features of FANs (as opposed to standard semantic networks) is that wffs are associated with nodes. This feature prompts the following question: what wffs are *inherited* by a node in a FAN? In a standard inheritance network, we focus primarily on the question: is there an undefeated positive path between A and B? Since most nodes in a network represent sets, and since the top node in an inheritance hierarchy often represents some attribute, the most intuitive interpretation of this question is: does some attribute hold of set A? Such questions are of interest in this inheritance network as well: we wish to know whether a particular medical service is covered by some benefit. In addition, however, we are interested in determining which business rules or wffs *apply* or *pertain* to that service (equivalently, to the node representing that service).

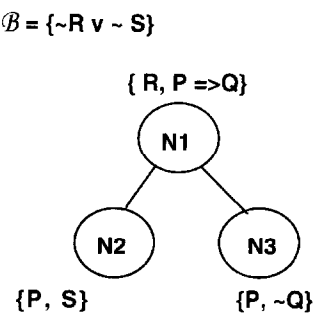

Fig. 4. Taking the union of wffs at nodes yields inconsistency.

Formally, we pose the question as follows. We introduce the following notation: N $\uparrow w$ if wff w pertains or applies to node N. We overload the $\uparrow$ symbol for sets, and say

$$\mathrm{N} \uparrow W \text{ if } \forall w\ w \in W \text{ iff } \mathrm{N} \uparrow w.$$

We define $\Psi(\mathrm{N}) = \{ w \mid \mathrm{N} \uparrow w \}$. That is, $\Psi(\mathrm{N})$ is the set of wffs that apply to node N. Then the question is: given a node N in a FAN, what is the set Ψ such that N $\uparrow \Psi$?

A naive approach might suggest taking the union of the wff set attached to N, together with the wff sets attached to all ancestors of N—or more precisely, of the wff sets attached to all those nodes to which there is an undefeated positive path from N. That is,

$$\Psi(\mathrm{N}) = \bigcup_{N \rightsquigarrow N_i} \mathrm{wffs}(N_i) \cup \mathrm{wffs}(\mathrm{N}).$$

Such an approach, however, is obviously wrong. Consider, for example, Fig. 4. There is an undefeated positive path from N3 to N1. Thus, according to the naive approach,

$$\Psi(\mathrm{N3}) = \mathrm{wffs(N3)} \cup \mathrm{wffs(N1)} = \{ \mathrm{P}, \neg \mathrm{Q} \} \cup \{ \mathrm{R}, \mathrm{P} \supset \mathrm{Q} \}.$$

But this set of wffs is obviously inconsistent. Note that the naive approach also fails to correctly compute Ψ(N2). Although wffs(N1) $\cup$ wffs(N2) is consistent, it is not consistent with respect to the background information $\mathcal{B} = \{ \neg \mathrm{R} \vee \neg \mathrm{S} \}$. Clearly, we do not want to blindly take unions of wff sets.

The problem arises because the wffs at a node are usually *typically* true at a node rather than always true at a node. In particular, a set of wffs W may be typically true at a node N, but may not be typically true at all subclasses of N. We discuss ways of dealing with this inconsistency in the next section.

5.2. *Effecting inheritance of wffs*

The previous section demonstrated that a naive approach to wff inheritance—namely, taking the union of wff sets at all nodes to which there is a positive undefeated path—leads to inconsistency. A proper approach to wff-inheritance must *recognize* potential inconsistency in inheriting wffs and *resolve* these inconsistencies.

Consider the set $S = \Psi(\mathrm{N})$, computed by the following process:

(1) Calculate all the nodes N_i such that $\mathrm{N} \rightsquigarrow \mathrm{N}_i$.

(2) Take the union of the wffs at N, the wffs at all N_i calculated above, and the background information, and determine if this set is consistent.
(3) If this union is not consistent, choose a maximally consistent subset of

$$\text{wffs(N)} \cup \bigcup_{N \rightsquigarrow N_i} \text{wffs}(N_i).$$

More precisely, choose a subset S that satisfies the following conditions:
(a) $S \subset (\text{wffs(N)} \cup \bigcup_{N \rightsquigarrow N_i} \text{wffs}(N_i))$;
(b) $S \cup \mathcal{B}$ is consistent;
(c) S is the largest such subset; that is, there does not exist S' satisfying conditions (a) and (b) such that $S' \subset S$.

We say that S is a maximally consistent subset of

$$\text{wffs(N)} \cup \bigcup_{N \rightsquigarrow N_i} \text{wffs}(N_i)$$

with respect to $\mathcal{B}$. If the meaning is clear, we may omit the reference to $\mathcal{B}$.

We are not suggesting that the above process is the one that should be used in computing S; we are rather using it to help *characterize* the set of rules that apply to a node.

The characterization of S explicitly refers to:
(1) the consistency of rule sets;
(2) the concept of a maximally consistent subset.

We make two brief remarks about the first issue. First, deciding whether a set of rules is consistent is only semi-decidable in general; however, we can restrict our attention to certain subsets of sentences such as those without existential quantifiers or self-embedding function symbols. Such sets are decidable. Second, even in decidable cases, deciding whether or not a set of sentences is consistent is intractable; we discuss ways of dealing with this problem in Section 5.3.

The second issue—choosing a maximally consistent subset—gives rise to some difficulty. In general, there is more than one maximally consistent subset of rules. The question is: which maximally consistent set should we choose? Since each maximally consistent subset is formed by deleting some of the wffs in an inconsistent set, an alternate phrasing of this question is: which wffs should we delete? That is, how do we decide that some wffs are more important than others? The general strategy is to articulate some *preference principles* for sets of wffs and to choose maximally consistent subsets in accordance with these principles.

The particular strategy developed in this paper is to examine preference principles that are based on the structure of the inheritance network. That is, we exploit as much as possible the structure of the inheritance network while we are constructing and choosing maximally consistent subsets. We focus on specificity and multiple paths, both nonconflicting and conflicting. We turn to several examples to illustrate this approach.

Some remarks on these examples: first, the examples deal with paths of length 1; however, we generalize to paths of arbitrary length in the discussion and specification of the wff-inheritance procedure. Second, to simplify and shorten the exposition, we have chosen very simple wffs; the inconsistencies are typically arithmetic in nature. Third, we show only a few wffs at each node. As noted in Sections 3 and 4, rules are typically much more complex and numerous, and contradictions between these complex wffs are rampant.

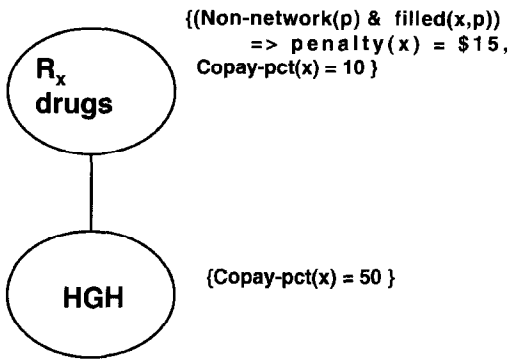

Fig. 5. The wffs at HGH are more specific than the wffs at R_x Drugs.

5.2.1. Specificity

Consider the example in Fig. 5. HGH (Human Growth Hormone) $\rightarrow$ R_x Drugs. Suppose we have the following wffs at these nodes (all variables are assumed to be universally quantified unless otherwise specified):

R_x Drugs:

{ copay-pct(x) = 10

(There is a 10% copay for all drugs)

(Nonnetwork(p) $\wedge$ filled(x,p)) $\supset$ penalty(x) = $15

(there is a $15 penalty for prescriptions if they are filled at nonnetwork pharmacies.) }

HGH:

{ copay-pct(x) = 50 }

(The copay for HGH drugs is 50%).

If we assume that the background context $\mathcal{B}$ entails some basic arithmetic facts such as

$$(v1 \neq v2) \supset (\neg \text{copay-pct}(x) = v1 \wedge \text{copay-pct}(x) = v2)$$

that is, a service cannot have two copay percentages, then the union of these two sets is inconsistent with respect to $\mathcal{B}$. [12]

Obviously, there are two maximally consistent subsets:

[12] The actual network contains the following cost-share rule at the R_x Drugs node: copay(x) = $3 (there is a $3 copay for all prescription drugs). The union of the two sets is then not, strictly speaking, inconsistent, as long as HGH drugs cost $6. In other words, the union of these sets entail that Human Growth Hormone drugs cost $6. Detecting unreasonable consequences such as this is an interesting problem in its own right, but beyond the scope of this paper. Presumably, there are several ways to deal with this problem: one can include price tables in the background information, or include general principles in the background information forbidding ridiculous consequences. The difficulty in the latter strategy then is a matter of expressing such principles and ensuring that we have got them all. (I am grateful to Ernie Davis for pointing out this problem.)

(1) The set of wffs at the R_x Drugs node, namely:

{ copay-pct(x) = 10
(Nonnetwork(p) ∧ filled(x,p)) ⊃ penalty(x) = $15 }

and

(2) the set

{ copay-pct(x) = 50
(Nonnetwork(p) ∧ filled(x,p)) ⊃ penalty(x) = $15 }

The choice of which maximally consistent subset to prefer is clear. The HGH node is more specific than the R_x Drugs node; thus we prefer the subset that has the wff from the HGH node to the subset that has the wff from the R_x Drugs node. That is, we prefer subset (2) to subset (1).

We say that subset (2) is a *preferred maximally consistent subset* (*pmcs*) relative to the sets of wffs at HGH and wffs at R_x Drugs, with the set of wffs at HGH preferred over the set of wffs at R_x Drugs. We define this concept formally below:

Definition 1 (Preferred maximally consistent subset). Assume that S_1, S_2 are sets such that $S_1 \cup S_2$ is inconsistent with respect to $\mathcal{B}$. Let X_1 and X_2 be maximally consistent subsets of $S_1 \cup S_2$ with respect to $\mathcal{B}$. Let R_1 be a subset of S_1; let R_2 be a subset of S_2, such that $R_1 = X_1 - X_2$; $R_2 = X_2 - X_1$. That is, R_1 and R_2 are all that distinguish X_1 from X_2. If this is the case, we say that $X_1 \cong X_2$ with respect to R_1, R_2. Then if S_1 is preferred to S_2, X_1 is a preferred maximally consistent subset of $S_1 \cup S_2$. Otherwise, if $S_1 \cup S_2$ is consistent with respect to $\mathcal{B}$, $S_1 \cup S_2$ is a (in this case *the*) preferred maximally consistent subset of S_1 and S_2.

Note that there is not necessarily a unique preferred maximally consistent subset. We introduce the function PMCS where PMCS(S_1, S_2) returns the *set* of all preferred maximally consistent subsets of S_1 and S_2, with S_1 preferred to S_2. By a slight abuse of notation, we will use the notation pmcs(S_1, S_2) to mean the (random or nondeterministic) function that returns one of the elements of PMCS(S_1, S_2). In addition, we extend pmcs to n sets, so that

$$\text{pmcs}(S_1, \ldots, S_n) = \text{pmcs}(\text{pmcs}(S_1, \ldots, S_{n-1}), S_n).$$

5.2.2. Multiple paths (nonconflicting)

Specificity is clearly not the only criterion one can use in determining a preferred maximally consistent subset. Consider Fig. 2. In this figure, the node Maternity Surgical has links to both Maternity and Surgical and thus inherits wffs from both nodes. The union of the wffs at these nodes { copay-pct(x) = 10, copay-pct(x) = 20} is obviously inconsistent with respect to the background constraint mentioned above. There are obviously two maximally consistent subsets:

{copay-pct(x) = 10}

{copay-pct(x) = 20}.

Although specificity does not help here—Maternity is neither more nor less specific than Surgical—it is clear which subset we should prefer. Since the link between Maternity Surgical and Maternity has preference over the link between Maternity Surgical and Surgical, it is reasonable to prefer the first subset, since this keeps the wff of the Maternity node. Thus, one can compute the rules that apply to Maternity by computing pmcs(wffs(Maternity), wffs(Surgical)). In fact, if there were wffs at Maternity Surgical as well, these wffs might contradict either the wffs at Maternity, the wffs at Surgical, or both. Since the wffs at more specific nodes are preferred, the computation of the rules that apply to Maternity Surgical would then be

$$\text{pmcs}\big(\text{wffs(Maternity Surgical)}, \text{pmcs}(\text{wffs(Maternity)}, \text{wffs(Surgical)})\big).$$

As this example shows, the order in which one takes preferred maximally consistent subsets is crucial. We explore this issue below.

5.2.3. A procedure for general paths

We turn to the issue of computing $\Psi(\mathrm{N})$ for paths of any length. Let us first consider a simple network in which there are no forking points—that is, for any node N, there is exactly one path from N to the root node (no multiple inheritance). It is clear that we do *not* want to compute $\Psi(\mathrm{N})$ by using the *characterizing* process described in Section 5.2. That is, we do not want to first take the union of all sets of wffs at the nodes on the path from N to the root, then take maximally consistent subsets of this large set, and finally choose preferred maximally consistent subsets relative to the specificity criterion. It is hard to beat this method for inefficiency. At the very least, the procedure to compute $\Psi(\mathrm{N})$ ought to iteratively traverse the path, computing $\Psi(\mathrm{N})$ as one goes along.

An iterative traversal of the path immediately gives rise to the question: in which order does one traverse the path?

Upward versus downward traversal. A downward traversal might seem natural for the following reasons. First, a downward traversal emphasizes the natural analogy between wff-inheritance and (iterated) belief revision [4,9]. Specifically, one might think of inheriting wffs as the process of revising the beliefs at a more general node by the wffs at a more specific node; this translates to the downward traversal of a path in the network. Second, one might expect downward traversal to have the side benefit of computing the sets of wffs that apply to intermediate nodes on the path.

However, straightforward downward traversal—a simple recursive procedure in which one sets $\Psi(\text{root}) = \text{wffs(root)}$, and for each nonroot node x, $\Psi(\mathrm{x}) = \text{pmcs}(\text{wffs(x)}, \Psi(\text{parent(x)})$—turns out to be incorrect. Consider computing $\Psi(\mathrm{N3})$ in Fig. 6. It is clear that the expected answer is $\{\mathrm{P}, \neg\, \mathrm{Q}\}$; the union of all wff sets is inconsistent, and one discards the wff at N1 since that is the least specific. However, if one traverses the path downward, one gets

$$\Psi(\mathrm{N1}) = \{\mathrm{P} \supset \mathrm{Q}\},$$

$$\Psi(\mathrm{N2}) = \{\mathrm{P} \supset \mathrm{Q}, \mathrm{P}\}.$$

When one reaches N3, one gets

$$\Psi(\mathrm{N3}) = \text{pmcs}(\{\neg \mathrm{Q}\}, \{\mathrm{P} \supset \mathrm{Q}, \mathrm{P}\}.$$

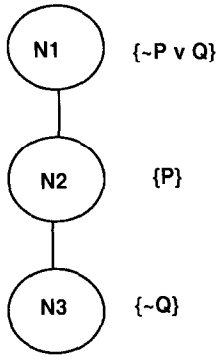

Fig. 6. Doing a simple downward traversal will give the wrong answer for wff-inheritance.

But there are two preferred maximally consistent subsets: $\{P, \neg Q\}$ and $\{P \supset Q, \neg Q\}$—contrary to expectations. The problem is that during the downward traversal, we have lost the information that P comes from a node that is more specific than the node from which $P \supset Q$ comes.

There are obvious ways to fix this problem: we can, for example, keep track of the source node of each of the wffs that has been collected so far.

If we wish to avoid this sort of bookkeeping, however, the simplest method for computing $\Psi(N)$ that is consistent with the specificity constraint is based on an upward traversal of the network. Specifically, one begins at the focus node N, taking wffs(N) as the starting set. One then proceeds up the path, at each node taking a preferred maximally consistent subset of the set computed so far and the wffs at the current node. This process will ensure that the specificity constraint is obeyed.

The algorithm will thus have an upward traversal of the network at its core. We now consider networks with forking points. In order to get the correct result, we must ensure that path ordering is respected in case of forking paths. This is done by examining all links at each point in the path, ordering them, and recursively proceeding up the more preferred links before the less preferred links.

In addition, we must ensure that we do not collect rules from nodes that are only on conflicted or preempted paths. The simplest way to avoid this problem is to preprocess the FAN to remove the preempted and conflicted links. We may do this by using an extension/modification of the procedure given in [41], which computes the *specificity extension* at a focus node.

We thus have the following procedure to compute $\Psi(N)$ in a FAN:

Algorithm COMPUTE $\Psi(N)$.

Preprocess the FAN to remove
preempted, conflicted links
$\Psi(N) := \text{wffs}(N)$
FAN-traverse(N)

Algorithm FAN-TRAVERSE(x).

if x has not been visited
  mark x as visited
  $\Psi(\mathrm{N}) := \mathrm{pmcs}(\Psi(\mathrm{N}), \mathrm{wffs}(x))$
  If x is a root
    then return
    else
      Determine all nodes y_i *such that* $x \rightarrow y_i$
      Do topological sort of all k links $x \rightarrow y_i$
      for $i = 1$ *to k do*
        FAN-traverse(y_i)

Note that since we are traversing a dag, as opposed to a tree, some nodes may be visited twice. To avoid recomputing preferred maximally consistent subsets (an expensive operation), we mark nodes as we visit them.

It will be noted that the computation of $\Psi(\mathrm{N})$ does not necessarily produce a unique set. This is because, as noted, the function pmcs does not yield a unique preferred maximally consistent subset. The existence of multiple maximally consistent subsets has always been something of a problem for nonmonotonic theories. In practice, however, the existence of multiple preferred maximally consistent subsets is not at all problematic for this domain. Indeed, it is perfectly legitimate to reason with any preferred maximally consistent subset of wffs that obeys the criteria of specificity and path preference. (In particular, members who are insured by some product often "win a case" by demonstrating that there is a coherent argument—that, is a preferred maximally consistent set of business rules—that supports their claim.) This phenomenon is at odds with the distaste that AI researchers usually have for credulous reasoning, but is worthy of further investigation.

5.2.4. An open question: the case of preempting paths

Consider now—in contrast to the previous examples—inheriting wffs in a network with conflicting multiple paths. Fig. 7 shows a portion of the drugs network. Although OTC Drugs (over-the-counter drugs) is a subtype of Drugs, it is not a subtype of the class of services covered by the Drugs benefit. On the other hand, a subtype of OTC Drugs, Insulin Syringes, is covered by the Drugs benefit. This is a classic case of preemption. In fact, this case is known as positive preemption, since a positive path—the direct link from Insulin Syringes to Services Covered by Drugs Benefit—preempts a negative path—the path from Insulin Syringes to OTC Drugs to Services Covered by Drugs Benefit.

Clearly, there are positive undefeated paths from Insulin Syringes to Services Covered by Drugs Benefit and from Insulin Syringes to OTC Drugs. Thus, Insulin Syringes stands to inherit wffs from both nodes. The question is: suppose the set of wffs at OTC Drugs is inconsistent with the set of wffs at Services Covered by Drugs Benefit (either absolutely, or with respect to the background information and/or the set of wffs at Insulin Syringes).

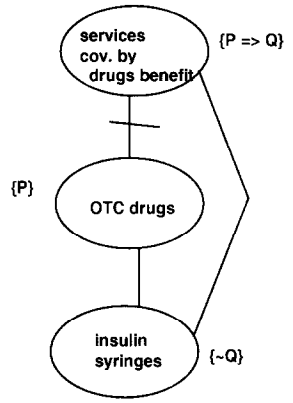

Fig. 7. Which node gets preference? Does preemption make a difference?

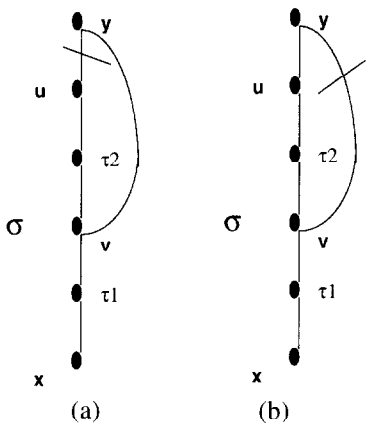

Fig. 8. (a) Positive path preempts negative path; (b) negative path preempts positive path.

Which set of wffs do we prefer? Do we prefer the wffs from Services Covered by Drugs Benefit because that node is on the preempting path? Or do we refrain from preferring either of the paths?

There are arguments both for and against preferring nodes from the preempting path. To formally state the arguments, we use Horty's [18] notation for preempting paths. A positive path $\pi(x, \sigma, u) \rightarrow y$ is *preempted* (in the context (Γ, Φ)) iff there is a node v such that

(i) either $v = x$ or there is a path of the form $\pi(x, \tau 1, v, \tau 2, u) \in \Phi$, and

(ii) $v \not\rightarrow y \in \Gamma$.

A negative path $\pi(x, \sigma, u) \not\rightarrow y$ is preempted (in the context (Γ, Φ)) iff there is a node v such that

(i) either $v = x$ or there is a path of the form $\pi(x, \tau 1, v, \tau 2, u) \in \Phi$, and

(ii) $v \rightarrow y \in \Gamma$. (See Fig. 8.)

The question is: Is the path $\pi(v, y)$ preferred to the path $\pi(v, \tau 2, u)$?

The argument for preferring a positive preempting path over the path it preempts runs as follows. Clearly, the positive path $\pi(v, y)$ is in some sense preferred to the negative path $\pi'(v, \tau 2, u, y)$ (that is the reason that we conclude y). Thus, preemption seems to offer some evidence that the preempting path is stronger than the preempted path. Then presumably, one of the links in the negative path $\pi'(v, \tau 2, u, y)$ is not as strong as the positive path between v and y. It is possible that the weak link is the negative link between u and y—but this is by no means definite, because we do, after all, conclude that u's are not y's. Thus, it seems as likely that one of the links in the path between v and u is weak. (Indeed, we know that at least in one respect v's are not typical u's—unlike typical u's, they are y's.) Thus, we ought to prefer $\pi(v, y)$ to $\pi(v, \tau 2, u)$.

On the other hand, the argument for *not* preferring a positive preempting path over the path it preempts rests on the fact that there is a positive undefeated path between v and u in the same way that there is a positive undefeated path between v and y. Thus, they both should have the same status; neither should be preferred.

If one does decide to prefer preempting over preempted paths, another question arises. Consider paths forking off from nodes in preempted paths (e.g., consider another link from u to some node z). Ought they also to have lower priority than preempting paths? The rationale given for preferring preempting to preempted paths would seem to hold here as well, but the intuition becomes increasingly weak.

Indeed, the lack of examples makes honing intuitions particularly difficult. There are few examples in this domain of positive preemption[13] and fewer still where there are conflicting wff sets in these portions of the network; thus, it is difficult to guess what the correct behavior ought to be.

The procedure to compute the wffs that apply to a node, outlined in the previous section, takes a neutral stand on the issue. It does not state a preference for preempting paths over the paths that are preempted, but it allows that preference to be incorporated. One can incorporate such a preference in two ways. First, one can place a condition on $\mathcal{O}$ stating that preempting paths are always strictly greater than their associated preempted paths. (One may in fact wish to insist that no paths come between the preempting and preempted path.) Second, one can modify the procedure so that, when one identifies the forking paths at a fork point, one groups preempting paths with the paths they preempt, and subsequently collects wffs along preempting paths before collecting wffs along the associated preempted paths.

The difficulty is not a technical one. The hard part here is getting the intuition right. The careful examination of other domains should shed some light on the question.

5.2.5. *Other preference criteria*

The computation of $\Psi(\mathrm{N})$ has thus far focussed on using information present in the structure of the network. Other standard preference principles may also be needed here. One may wish to assign some wffs a higher priority than others (as in [26]), regardless of

[13] The problem does not arise at all in cases of negative preemption—when a negative path preempts a positive path. That is because one never inherits from the last node of a negative path. In Fig. 8(b), one would never wish to inherit from node y—and inheriting along the path from node x to node u is simply standard wff-inheritance using the specificity criterion.

the rule's position in the network. For example, it might always be the case that medical rules have higher priorities than administrative rules. Likewise, it is also reasonable to prefer a particular subset of rules based on the results that this subset entails. This is equivalent to preferring one extension, or model, over the other (as in [40]). For example, we may prefer extensions in which a claim gets paid to one in which the claim does not get paid.

5.3. Computational issues

Inheriting rules immediately transforms the problem of inheritance from a tractable problem (at least in the case of upwards inheritance: see [38]) to one that is badly intractable. Wff-inheritance in this domain will often be done in a static setting as opposed to a dynamic one,[14] and this takes some of the sting out of the fact that computing preferred maximally consistent subsets is NP-hard. Nevertheless, intractability is clearly an unpleasant issue that we must handle in some form.

In practice, we have discovered that we can deal with the complexity issue by using a divide-and-conquer strategy. The trick is to divide the set of wffs into k types, subject to the following constraint:

If $Wffs(i, N_1) \cup Wffs(j, N_2)$ *is inconsistent with respect to* $\mathcal{B}$, *then* $i = j$.[15]

That is, wffs are constrained so that sets can contradict one another only within their own type. This cuts down on much consistency checking (since often when N1 $\rightsquigarrow$ N2, the wff sets at N1 are of a different type than the wff sets at N2) and greatly reduces the time needed for consistency checking and maximal subset construction and choice. Obviously, the greater k is, the more this strategy helps.

Finding a division of the wffs into sets that satisfy this constraint is not a trivial task. The wffs in this domain are taken from business rules that were originally categorized by employees of the insurance corporation into cost-share, access, administrative, and medical rules. Certain of these groups had the desired property—e.g., cost-share rules did not contradict rules of any other type—but others did not: medical rules often contradict administrative rules. On the other hand, it was possible to identify some types of administrative rules that were consistent with all rules outside of their own type. In any case, it is hardly surprising that a categorization of rules developed from a business viewpoint is not optimal in terms of efficiency.

Finding the proper knowledge representation is clearly very important. Doing so is no longer only of academic interest; it can greatly affect the tractability of the system.

5.4. Back to attribute inheritance

We first noted the need to do conflict resolution and recognition for the purposes of wff inheritance. Further reflection, however, suggests that similar problems can arise even

[14] Since the rules that apply to a medical service can be computed off-line. On the other hand, employees of the insurance corporation who modify the network may wish to see in real time how adding or changing rules at points in the network changes the rules that apply to a service.

[15] The notation Wffs(i,N) should be read: the set of wffs at node N of type i.

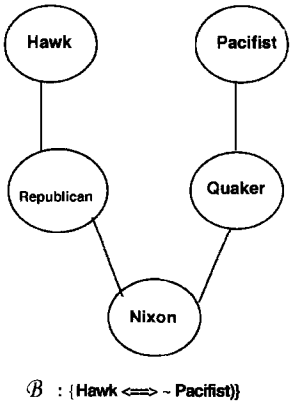

Fig. 9. The modified Nixon diamond is inconsistent, but only implicitly.

when performing standard attribute inheritance. Consider again Fig. 2. Maternity Surgical $\rightsquigarrow$ Covered by Surgical Benefit; Maternity Surgical $\rightsquigarrow$ Covered by Maternity Benefit.

In fact, however, Maternity Surgical cannot be covered by both benefits: there is a constraint that services are covered by at most one benefit. This constraint is not explicit in the structure of the network; it is entailed by background domain knowledge; i.e., the background $\mathcal{B}$. The point of this example is not in enforcing this constraint—implementation is quite simple—but in recognizing the inconsistency. In general, when some of the knowledge of the inheritance network is present as background knowledge, inheriting attributes from multiple parents has the potential for leading to inconsistency, even if this is not explicit.

Oddly, this problem has not been discussed in the inheritance literature, perhaps because inconsistencies in the examples used have always been explicit. The Nixon diamond example [35] is a classic example of such an explicit inconsistency. Suppose, however, we modify Touretzky's [42] modification of this example. (See Fig. 9.) (Nixon $\rightarrow$Republican; Nixon $\rightarrow$Quaker; Quaker $\rightarrow$Pacifist; Republican $\rightarrow$Hawk.) There is no explicit contradiction between Hawk and Pacifist, but if we add a statement to the background theory stating that the two concepts are contradictory, $Hawk \equiv \neg Pacifist$, then there is an inconsistency that must be resolved. Depending on the amount and form of the background knowledge, detecting and resolving this inconsistency can be arbitrarily difficult (that is, as hard as the problem of wff-inheritance).

6. Relating FANs to previous work

6.1. Comparing FANs to other knowledge structures

FANs are standard inheritance networks with wffs attached to nodes and in the background. The trend away from ontological promiscuity (the wanton creation of new knowledge structures) prompts the following questions. In what way are FANs different from existing knowledge structures? Are they truly necessary or can we get by with more

familiar mechanisms? This section argues that they are particularly natural for the domain at hand, and that existing structures have their own pitfalls.

FANs are extremely natural for any commercial domain in which general taxonomic information is best viewed separately from the nontaxonomic information—the domain's business rules. This is precisely the situation in the medical insurance domain. The taxonomic information represents the structure of medical services by the medical profession (as well as the broad benefit categories that are standard in the insurance industry). This is a relatively static structure across products and corporations. The wffs represent the industry's business rules which change rapidly, across products and corporations. In fact, the network has proved to be especially easy for customer service representatives to master. At the same time, representing the business rules as wffs at nodes has kept the size of the network manageable (250–300 nodes).

We now examine alternatives to FANs. There are two obvious choices: representing everything in an inheritance network, or doing this entirely in a nonmonotonic logic.

Alternative I: *putting everything in the inheritance network.* As mentioned in the introduction, only some wffs can be easily represented as links between classes and attributes. However, we can capture all wffs simply by reifying each wff as a node. For example, a node $\forall x \phi(x)$ could be reified as the node representing the property of satisfying ϕ. There are several objections to this alternative.

First, many such nodes would be very unnatural (consider the sample wffs in Sections 3 and 4), and quite outside the spirit of a semantic network. That is, nodes in the network, instead of representing concrete entities such as services and benefits, would represent ontologically dubious creatures.

Second, the problem of wff-inheritance has now been recast as the problem of determining whether two or more paths conflict. In particular, assume wffs $\phi_1, \ldots, \phi_n$ such that $\phi_1 \wedge \cdots \wedge \phi_n$ is inconsistent. Let $y_1, \ldots, y_n$ be the nodes associated with these wffs. It is clear that the paths $x \rightsquigarrow y_1, \ldots, x \rightsquigarrow y_n$ conflict in the intuitive sense of the term. But this conflict cannot even be expressed directly in a standard Hortian or Touretzkian framework. Even if we increase expressivity, there is no obvious way to recognize and resolve this conflict. In contrast, determining maximally consistent subsets of wffs is a well-understood problem. [16]

Alternative II: *doing everything in a nonmonotonic theory.* That is, translate the inheritance network into a nonmonotonic theory, such as autoepistemic logic [32] or circumscription [25]. [17] However, see also Horty [18], who argues that specifying a translational semantics for a path-based inheritance network is a problem that is not yet completely solved. In any case, it is certainly possible, given a *particular* network, to give an equivalent nonmonotonic theory.

[16] As mentioned in the previous section, the general problem of determining whether paths conflict already exists in many networks, such as the modified Nixon diamond of Fig. 9. While we see that these problems can beset any network, we nevertheless believe that flooding networks with this problem, which would happen if wffs were reified as nodes, is not desirable. This is particularly true since the problem of determining whether arbitrary paths in a network are inconsistent with respect to one another has not been addressed.

[17] There are a variety of methods for doing this, such as [11].

There are several objections to this strategy. First, the distinction between static taxonomic rules and more volatile business rules has been erased. Second, this representation is much less accessible to users of the expert system, who find browsing through an inheritance network easy to learn and understand. Either they must live with this hardship or we must keep the FAN *as well as* the nonmonotonic theory. We must then either construct a mapping between the two—a nontrivial task—or be sure to always update the two structures simultaneously, an unrealistic constraint in the business world. Third, such a strategy may be computationally wasteful in that in translational theories, one must generally recompute priorities and/or specificities in a more tedious manner than the algorithms available for determining specificity in an inheritance network.

6.2. *Related work*

There is much work in the literature on determining maximally consistent sets of defaults ([2,6,10,16] among others). Such work is obviously very close in spirit to ours. In particular, there is an emphasis in these works on considerations of specificity. This paper differs from these works in a number of respects (such as the consideration of nonconflicting multiple-path inheritance). Most importantly, what distinguishes this paper from these works is the emphasis on the semantic network structure which we take as our starting point. The other works take some nonmonotonic system as their starting point. The difference is analogous to the distinction that Horty [18] points out between path-based and translational theories of inheritance; the latter specify the meaning of a network in terms of a nonmonotonic formalism; the former specify the meaning in terms of the paths themselves. Just as Horty argued for the naturalness and intuitiveness of path-based approaches for standard inheritance, we argue for the ease, naturalness, and intuitiveness of the path-based approach to wff-inheritance. In particular, note the discussion in [18], which argues that researchers using the translational approaches have not yet satisfactorily formalized the concept of specificity, which is easy to formalize within a path-based approach.

We do not, however, believe that these systems should be seen as competing. Rather, we believe that the following is the case: a major task in knowledge representation is the articulation of the structure that is most natural and useful for a particular application. Another major task is demonstrating, if possible, that this structure is equivalent to more familiar structures, or if that is not possible, demonstrating the inequivalence. In this spirit, we suggest that formula-augmented semantic networks have a useful role to play in domains in which taxonomic knowledge is a major, but not sole, component of the domain knowledge. (For example, we have used FANs in life insurance and property and casualty applications.) At the same time, we hope that future research will shed light on the equivalences between FANs and the systems of Geffner [10], Grosof [14,15], and others.

7. Generalizing wff-inheritance

Can the techniques of wff-inheritance, which were developed for the particular problem of benefits inquiry in the medical insurance domain, be generalized to other problems in industry?

Some generalizations are obvious. Wff-inheritance would clearly be useful for benefits inquiry in other parts of the insurance industry, such as life insurance and property and casualty insurance. In these industries, it is also the case that services are best organized taxonomically, and that business rules apply to services. Wff-inheritance may also be useful for other tasks in insurance, such as adjudication, which would use a taxonomy of services very close to the structure used in benefits inquiry, and administration, which would use a taxonomic structure of products as well as services.

The nonmonotonic techniques discussed here are applicable to a wide range of other problems as well. Indeed, as suggested in the previous section, it can be argued that the construct of a FAN and its associated algorithms may prove useful in other domains which satisfy the following criteria:

(1) there exists a large amount of taxonomic information;
(2) there exists a significant amount of nontaxonomic information;
(3) the taxonomic information is conceptually linked to the nontaxonomic information;
(4) the nontaxonomic information can be mapped into wffs.

There are a number of potential domains:

- *Legal reasoning*, especially case law: legal cases are often organized taxonomically, and different legal rulings are associated with cases; it is reasonable to suppose that these legal rulings can be mapped into wffs. Most automated legal reasoning has been case-based [1], or analogical; adding wff-inheritance may significantly enhance the power of legal reasoning systems. Indeed, one can view FANs as providing an organizing framework for the cases in a case-based reasoning framework, with formal characterizations of some aspects of the similarity relation on cases.
- *Medical reasoning and treatment*: medical conditions are often organized taxonomically, and protocols are associated with these conditions. The protocols are often rigorous sequences of steps which can be mapped into wffs.
- *Reasoning in business organizations*: the organization chart in many businesses is a perfect taxonomy, and there are many rules associated with different positions in the org chart.

These extensions are not straightforward; in particular, mapping business or legal rules into wffs is nontrivial. However, these examples indicate that the usefulness of FANs extends far beyond the domain for which they were invented.

8. Into the future

The detailed examination of an application of nonmonotonic reasoning to industry has taught us some valuable lessons and has suggested several directions for future research.

(1) We need to constantly keep our eyes open for problems in industry that could benefit from nonmonotonic reasoning. There are many such problems; the trick is to identify them. Certainly, we should look out for problems that could benefit from FANs, as suggested above. In general, we should look for domains where exceptions are relatively common.
(2) Basic research is still crucial. We need serious research on theoretical aspects of nonmonotonic reasoning. It would be best if such research were guided by specific

issues highlighted by the study of particular problems in industry. In fact, one of the unexpected dividends of intensively studying a problem in industry is that it often results in the discovery of theoretical problems that were not previously considered. For example, while I was developing the benefits inquiry system described here, I discovered a number of issues that theories of inheritance had not previously examined. Such problems include the interaction of composition and subtyping and non-unary inheritance [29].

The importance of basic research cannot be overstated. Some of the most heartening news about the current state of nonmonotonic research is the recent spate of exciting results regarding the complexity of some restricted nonmonotonic theories. Examples include Grosof's result that computation of the answer set for courteous logic programs, a restricted form of prioritized defaults, is $O(n^2)$ [15]. These results are being translated into commercial products. In particular, courteous logic programs are used in RAISE, a system for building intelligent agents, now commercially released [14].

(3) The proper balance between basic research and serious involvement in industry is important, but difficult to maintain. One meaty industry problem can easily give a theoretical researcher enough material for a decade; on the other hand, we need to work on many industrial problems to get a fair idea of the problems that need to be solved, and to convince industry of the relevance of nonmonotonic reasoning.

The ideal relationship between academe and industry is one that is mutually beneficial: industry benefits because academe offers a better way of solving the problem; academe benefits because examining the problem in industry opens up an interesting area of inquiry and leads to new and exciting results. There is no point in choosing boring problems in industry (though we are not always in a position to do the choosing).

(4) We must develop tools to perform nonmonotonic reasoning. We need to develop general tools for inheritance with exceptions; we also need to develop a tool for general inheritance with wffs. Thus far, inheritance with wffs has been developed for only one application and modified for another. A general tool will facilitate the extension of FANs to other problems in industry, as suggested above.

(5) Solving a business problem often requires more than providing a solution to a problem in nonmonotonic reasoning. For example, one of the biggest problems we faced in the development of the benefits inquiry system was the formalization of the many business rules in the insurance industry: we ran up against the knowledge representation bottleneck. The representation of a large amount of business knowledge—in particular, the development of the proper ontology—is often more difficult than the computational aspects of reasoning with that knowledge.

Finally, we must keep in mind that researchers in nonmonotonic reasoning do not always face a friendly landscape. Some things we ought to watch out for:

(1) Shortsightedness. It always takes longer to solve a problem well, especially the first time. Using nonmonotonic reasoning takes a lot more time, and the advantages may not always be obvious to anyone but AI researchers. AI researchers should be prepared for the possibility of an uphill battle, both with one's management chain

and with the customer. This is not a problem unique to the field of nonmonotonic reasoning, of course.

(2) Refusing to accept the importance of plausible reasoning. This comes in many guises:

- The 80–20 rule. This line of argument runs as follows: even if we ignore exceptions, we will still get things right most (around 80%) of the time, and with very little effort. Is it not worth taking that route?
  The 80–20 rule is particularly pernicious if one is willing to accept wrong answers 20% of the time (one can only hope that this rule is not invoked by the FAA), but is quite troublesome even if one is merely willing to accept admissions of ignorance (answers of "I don't know") 20% of the time. Even in the relatively benign domain of benefits inquiry, a system that cannot answer questions 20% of the time is not very useful: its performance would scarcely be better than the desktop systems that are designed to answer the most frequently asked questions, or CSRs without any aid of technology who can generally answer frequently asked questions right off the bat.
- The back-to-if-then-else movement. This argument recognizes the importance of exceptions, but insists that any branching statement is all that is needed. People who use this argument are convinced that all that nonmonotonic reasoning is trying to achieve has been present since the days of Algol 60 or earlier.
- The protection-of-basic-researchers strategy. Despite constantly urging nonmonotonic researchers to do something practical, management often tries to keep a buffer between researchers and industry. The trouble with this is that if researchers cannot get close enough to industry, they cannot find the problems that are most suitable; if they only hear about a problem second hand, they do not have an accurate picture of the situation, and they cannot determine whether and how nonmonotonic reasoning is useful.

The best way to counteract these obstacles is to demonstrate that nonmonotonic reasoning is capable of yielding practical results. We will achieve recognition when we affect the outside world.

Acknowledgement

I am grateful to Ernie Davis, Benjamin Grosof, Moninder Singh, and Rich Thomason for helpful discussions and suggestions.

References

[1] K. Ashley, Modeling Legal Argument: Reasoning with Cases and Hypotheses, MIT Press, Cambridge, MA, 1991.

[2] G. Brewka, Preferred subtheories: an extended logical framework for default reasoning, in: Proc. IJCAI-1989, Detroit, MI, 1989, pp. 1043–1048.

[3] W.F. Clocksin, C.S. Mellish, Programming in Prolog, 3rd ed., Springer, Boston, MA, 1987.

[4] A. Darwiche, J. Pearl, On the logic of iterated belief revision, in: R. Fagin (Ed.), Proc. 5th Conference on Theoretical Aspects of Reasoning about Knowledge, Morgan Kaufmann, San Mateo, CA, 1994, pp. 5–23.

[5] E. Davis, Representations of Commonsense Knowledge, Morgan Kaufmann, San Mateo, CA, 1990.

[6] J. Delgrande, T. Schaub, A general approach to specificity in default reasoning, in: Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94), Bonn, Germany, Morgan Kaufmann, San Mateo, CA, 1994, pp. 146–157.
[7] W.H. Dowling, J.H. Gallier, Linear time algorithms for testing the satisfiability of propositional Horn formulae, Journal of Logic Programming 3 (1984) 267–284.
[8] D. Etherington, Reasoning with Incomplete Information, Morgan Kaufmann, Los Altos, CA, 1988.
[9] P. Gardenfors, Knowledge in Flux, MIT Press, Cambridge, MA, 1988.
[10] H. Geffner, Default Reasoning: Causal and Conditional Theories, MIT Press, Cambridge, MA, 1990.
[11] M. Gelfond, H. Przymusinska, Formalization of inheritance reasoning in autoepistemic logic, Fundamenta Informaticae 13 (1990) 403–443.
[12] M. Ginsberg (Ed.), Readings in Nonmonotonic Reasoning, Morgan Kaufmann, San Mateo, CA, 1987.
[13] G. Gottlob, Complexity results for nonmonotonic logics, Journal of Logic and Computation 2 (3) (1992) 397–425.
[14] B. Grosof, Building commercial agents: an IBM research perspective, in: Proc. 2nd International Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM-97), 1997. Available as IBM Research Report RC20835 and at http://www.research.ibm.com/people/g/grosof.
[15] B. Grosof, Proritized conflict handling for logic programs, IBM Research Report RC20836, 1997. Also at http://www.research.ibm.com/people/g/grosof.
[16] B. Grosof, Generalizing prioritization, in: J. Allen, R. Fikes, E. Sandewall (Eds.), Proc. 2nd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, Morgan Kaufmann, San Mateo, CA, 1991, pp. 289–300.
[17] S. Hanks, D. McDermott, Nonmonotonic logic and temporal projection, Artificial Intelligence 33 (3) (1987) 379–412.
[18] J. Horty, Some direct theories of nonmonotonic inheritance, in: D. Gabbay, C. Hogger, J. Robinson (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3: Nonmonotonic Reasoning and Uncertain Reasoning, Oxford University Press, Oxford, 1994, pp. 111–187.
[19] J. Horty, R. Thomason, Mixing strict and defeasible inheritance, in: Proc. AAAI-1988, St. Paul, MN, Morgan Kaufmann, San Mateo, CA, 1988, pp. 427–432.
[20] J. Horty, R. Thomason, D. Touretzky, A skeptical theory of inheritance in nonmonotonic semantic networks, Artificial Intelligence 42 (1990) 311–349.
[21] H. Kautz, B. Selman, Hard problems for simple default logics, in: R. Brachman, H. Levesque, R. Reiter (Eds.), Proc. 1st International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Ontario, Morgan Kaufmann, San Mateo, CA, 1989, pp. 189–197.
[22] W. Kneale, M. Kneale, The Development of Logic, Clarendon Press, Oxford, 1962.
[23] S. Kraus, D. Lehman, M. Magidor, Nonmonotonic reasoning, preferential models, and cumulative logics, Artificial Intelligence 44 (1990) 167–207.
[24] E. Mays, R. Dionne, R. Weida, K-REP system overview, SIGART Bulletin 2 (3) (1991).
[25] J. McCarthy, Circumscription—a form of nonmonotonic reasoning, Artificial Intelligence 13 (1980) 27–39.
[26] J. McCarthy, Applications of circumscription to formalizing common-sense knowledge, Artificial Intelligence 28 (1986) 86 116.
[27] D. McDermott, J. Doyle, Nonmonotonic logic I, Artificial Intelligence 13 (1980) 41–72.
[28] L. Morgenstern, Inheriting well-formed formulae in a formula-augmented semantic network, in: Principles of Knowledge Representation and Reasoning: Proc. 5th International Conference (KR-96), 1996, pp. 268–279.
[29] L. Morgenstern, New problems for inheritance theories, in: Proc. 3rd Symposium on Formal Theories of Commonsense Reasoning, Stanford, January 1996. Available at http://www-formal.stanford.edu/leora.
[30] L. Morgenstern, The problem with solutions to the frame problem, in: K. Ford, Z. Pylyshyn (Eds.), The Robot's Dilemma Revisited, Ablex, Norwood, NJ, 1996, pp. 99–133.
[31] L. Morgenstern, M. Singh, An expert system using nonmonotonic techniques for benefits inquiry in the insurance industry, in: Proc. IJCAI-97, Nagoya, Japan, 1997.
[32] R. Moore, Semantical considerations on nonmonotonic logic, Artificial Intelligence 25 (1) (1985) 75–94.
[33] Proceedings of the Conference on Practical Applications of Prolog, 1997.
[34] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.
[35] R. Reiter, G. Criscuolo, On interacting defaults, in: Proc. IJCAI-1981, Vancouver, BC, 1981, pp. 270–276.

[36] T. Schaub, P. Nicolas, An implementation platform for query-answering in default logics: the XRay system, its implementation and evaluation, in: J. Dix, U. Furbach, A. Nerode (Eds.), Logic Programming and Nonmonotonic Reasoning, Springer, Berlin, 1997.

[37] J. Schmolze, T. Lipkis, Classification in the KL-ONE Representation System, in: Proc. IJCAI-1983, Karlsruhe, Germany, 1983.

[38] B. Selman, H. Levesque, The complexity of path-based defeasible inheritance, Artificial Intelligence 62 (2) (1993) 303–340.

[39] M. Shanahan, Solving the Frame Problem, MIT Press, Cambridge, MA, 1997.

[40] Y. Shoham, Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence, MIT Press, Cambridge, MA, 1988.

[41] L. Stein, Resolving ambiguity in nonmonotonic inheritance hierarchies, Artificial Intelligence 55 (1992) 259–310.

[42] D. Touretzky, The Mathematics of Inheritance Systems, Morgan Kaufmann, Los Altos, CA, 1986.

[43] L. Zadeh, Fuzzy sets, in: D. Dubois, H. Prade, R. Yager (Eds.), Readings in Fuzzy Sets and Intelligent Systems, Morgan Kaufmann, San Mateo, CA, 1992.