An Expert System Using Nonmonotonic Techniques for Benefits Inquiry in the Insurance Industry

Leora Morgenstern
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
leora@watson.ibm.com

Moninder Singh
Department of Computer & Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
msingh@gradient.cis.upenn.edu

Abstract

This paper describes BenInq, an expert system for benefits inquiry in the insurance industry. BenInq is designed to be used by both customer service representatives, who answer customers' questions in real time, and policy modifiers, who update insurance products. The main challenges were the design of the KR structure — in particular, representing a huge knowledge base at varying levels of granularity, and the development of the reasoning methods — in particular, developing a method that determines which insurance regulations apply to an insurance service. This required nonmonotonic reasoning, which we modeled using an extension of inheritance methods. BenInq represents one of the few large scale industrial applications that explicitly uses formal nonmonotonic reasoning techniques.

1 Introduction

Benefits inquiry, the process of querying an insurance company to determine one's benefits, is becoming increasingly complex. Whereas years ago, a medical insurance company would offer only a small number of products, today companies have thousands of different insurance products, each of which contains a myriad of services and regulations. Furthermore, these products are always changing. The amount of information is difficult to keep up with; thus, the need for an expert system.

The expert system described in this paper, BenInq, was created for the application of benefits inquiry in the medical insurance domain. The paper is structured as follows. We first describe the existing technology for benefits inquiry. We next list the desiderata for our expert system. Subsequently, we discuss the knowledge representation and reasoning issues, focusing on the nonmonotonic techniques used. We then present BenInq, evaluate its usefulness, and discuss how the techniques can be generalized.

2 Existing Technology

Benefits inquiry has traditionally been performed by customer service representatives (CSRs) who rely on basic charts, detailed manuals, and implicit knowledge gained through common sense and 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. 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 this 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. But 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 on both screens. Third, there are a small number of screens relative to the num-

ber of types of questions a CSR may have to answer. Thus, the CSR may not get the level of information that she needs. This is not an artifact particular to this application but is due to the logistical difficulties of creating and continually modifying a large number of 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 if a particular service is covered, he is asked to provide the CSR with the *procedural* (CPT) code, which represents the service which is to be performed, and the *diagnostic* (ICD-9) code, which represents the diagnosis that is the reason for the service that 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' question can be answered quickly and unambiguously. The disadvantages are: First, customers must know complete code information before they can call their CSRs; this slows down the inquiry process. Second, the code-based scheme only allows questions 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 in addition to codes 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.[1]

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 and navigation around a large information space
- 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.

[1]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.

3 Knowledge Rep and Reasoning

Insurance employees such as CSRs and PMs must reason about services, benefits, coverage information, and business rules. A *service* represents some class of medical services, such as Surgery or X-rays. A *benefit* is some component of an insurance product which covers or excludes a particular service. For example, the Drug Benefit may generally cover Prescription Drugs, though it may exclude certain classes of prescription drugs, such as Experimental Drugs. *Coverage information* indicates which services are covered or excluded by which benefit, as in the above example. *Business rules* limit or modify the ways in which services are covered by benefits. Rules include *cost-share* rules, which specify the deductible and co-pay of a covered service, *access* rules, specifying whether it is required to go to a network provider, and *administrative* and *medical* rules specifying other restrictions. For example, there is a cost-share rule specifying a 20% co-pay for surgical services, and an administrative rule stating that patients in drug rehabilitation programs lose all rehab benefits for a year if they are non-compliant.

CSRs and PMs must deal with a large amount of information. There are tens of thousands of services of interest. There may be several thousand business rules for an insurance product; many of these rules (such as cost-share) will apply to a great number of services.

In developing BenInq, we needed to focus on the *external* representation—the representation that the CSRs and the PMs see, as well as the *internal* representation, which the inner workings of the system use. Since BenInq would be used primarily to answer questions about service coverage and applicable business rules, the internal representation had to facilitate efficient reasoning about these issues. In addition, to ease use by CSRs and PMs, the external representation had to be perspicuous to employees of the insurance company. Moreover, PMs needed to directly manipulate knowledge in the system in order to modify products. The easiest way to achieve this was to have a direct mapping between the internal and external knowledge representations. A complete direct mapping is not always possible.

The choice of representation resulted from the observation that many services can be organized in a taxonomic manner (see Figure 1). For example, the services of PET Scans, Lab Tests, EKG/EEG, and Genetic Testing are all subtypes of Diagnostic Services. The taxonomy generally is at least several layers deep. Benefits likewise can be arranged taxonomically (see Figure 2).

The structure, however, is not purely taxonomic. Certain services have multiple supertypes. For example, Genetic Testing is both a subtype of Diagnostic Services and of Family Planning Services. Thus, the structure is a dag rather than a tree.

The quasi-taxonomic structure led us to consider a semantic network—in particular, an inheritance network with exceptions—as our KR structure. There are efficient algorithms for reasoning with a semantic network.

Moreover, such networks are natural to laypersons such as CSRs and PMs; this naturalness guarantees a direct mapping between the external and internal knowledge representations. We needed to determine, however, whether a semantic network was capable of encoding all the knowledge that was needed for the benefits inquiry system, specifically, (1) coverage or exclusion of medical services and (2) rules limiting or modifying the ways in which a service was covered.

3.1 Representing Coverage Information

Coverage and exclusion information can easily be encoded in the semantic network. In addition to the subtype link, one introduces covers and excludes link which connect benefit and service nodes. Thus, to represent the information that Diagnostic Services are *covered* by the Diagnostic Benefit, one places a covers link between Diagnostic Benefit and Diagnostic Services; to represent the information that PET Scans are *excluded* by the Diagnostic Benefit, one places an excludes link between Diagnostic Benefit and PET Scans. The covers and excludes links propagate along the taxonomy, so that, e.g., we can reason that Diagnostic Benefit covers Blood Work, since Blood Work is a subtype of Diagnostic Services. The propagation is relative to specificity considerations, so we do not use propagation to conclude that PET Scans are covered by the Diagnostic Benefit: the path from PET Scans to Diagnostic Benefit along the excludes link is more direct than the path from Pet Scans to Diagnostic Benefit along the covers link.

It is obvious that there is a close connection between this structure and a standard inheritance network with exceptions [Horty *et al.*, 1990], and in fact we can make this connection (an example of a direct mapping) 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., the Diagnostic Benefit node would be replaced by the node Services Covered by Diagnostic Benefit). Replace each covers link by a subtype (isa) link, and each excludes link by a cancels link. One then uses a standard traversal algorithm for inheritance networks with exceptions [Horty *et al.*, 1990; Stein, 1992] to determine whether or not services are covered. 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 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 covered or excluded by some benefit; there are almost always exceptions. This nonmonotonicity is hardly surprising; proponents of nonmonotonic logic [Reiter, 1980] 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 3.4.

3.2 Representing Rules

Certain rules lend themselves to representation within a semantic network. Consider the rule:

There is a co-pay 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% co-pay, 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 non-compliant

cannot be so easily represented. One could posit a node that represents the services which have the property that if patients are non-compliant wrt that service, then they lose all rehab 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 well outside of the spirit of a semantic network, where nodes are supposed to represent easily understood and expressible concepts.

Instead, we choose to represent rules as well-formed formulae in some classical logic. For example, the rule above could be translated as

Drug-rehab(p) $\wedge$ enrolled(x,p,i) $\wedge$ $\exists$ j sub(j,i) $\wedge$ non-compliant(x,p,j) $\Rightarrow$ ((Drug-rehab(p2) $\wedge$ enrolled(x,p2,i2) $\wedge$ time(start(i2)) - time(end(j)) $\leq$ 1year) $\Rightarrow$ $\neg$ covered(x,p2)).

The wff translation of business rules exists in the internal representation.

These rules seem to be attached to specific nodes in the net. For example, the rule specifying that the diagnostic co-pay is 20% ought to be attached to the Diagnostic Services node in the network. This observation prompted the development of FANs (formula-augmented semantic networks). A FAN is a structure which allows attaching wffs to nodes in a semantic network. Formally, a FAN is a tuple consisting of a set of nodes, a set of wffs, a set of links (subtypes) on nodes, a (possibly empty) set of partial orderings on these links, allowing the prioritization of links in cases of multiple supertyping, and links connecting nodes and sets of wffs. Details are given in [Morgenstern, 96].

The semantics of a rule attached to a node is roughly: the rule attached to a node usually applies to the services represented by the node. Nonmonotonicity is again present; the rule may not apply to all subtypes of the node. Thus, a natural question is: are rules inherited along taxonomic lines? We discuss this issue in sec. 3.4.

3.3 Categorizing Nodes

Benefit nodes correspond to the types of benefits provided by a particular insurance product. There are about twenty to thirty benefit nodes in a semantic network representing a typical insurance product.

In contrast to benefit nodes, which are few in number, there are many service nodes. Medical care providers speak of tens of thousands of medical services. Even if we do not represent all of these, we need to have some broad classification of services. A first cut at classification is provided by the insurance industry, which categorizes services according to *(professional) procedure*, *condition* or *setting*. Examples of professional procedure services, which are categorized according to the professional who provides the services, are Surgical services, Therapy, and Diagnostic services. Examples of condition services, which are categorized according to a particular class of conditions, include Maternity and Mental Health. Examples of setting services are Hospital Ancillary services or Inpatient Hospice Room and Board.

Each of these classes of services itself contains a few major classifications, roughly corresponding to the way laymen speak of medical services. Formally, a classification is considered major if it is a root node in the service taxonomy. Examples are Surgical, Diagnostic, Preventive, and Hospital Services. There are between 30 and 40 major classifications for a typical insurance product.

If we decide to explicitly represent tens of thousands of services in the semantic network, each major classification will have a very large number of nodes, much too large to display or for a CSR to comprehend. We are faced with a trade-off of ease of display, navigation, and comprehensibility, versus granularity and comprehensiveness. Achieving arbitrarily fine levels of granularity for all types of services will result in a system that may be too unwieldy for a CSR to use.

Our approach is to generate nodes in the (external representation of the) semantic network according to the following principle. A subtype is explicitly given only if there is a reason: for example, a particular subtype may not be covered, or different rules may attach to the subtype. Thus, Endoscopic Surgery is given as a subtype of Surgery because there are different rules for Endoscopic Surgery than for standard Surgery; Routine Endoscopic Surgery is given as a subtype of Endoscopic Surgery because Routine Endoscopic Surgery is not covered. On the other hand, there are no explicit nodes for appendectomy or gall bladder surgery, because these are standard surgeries; one can always just look at the Surgery node to determine coverage and the rules.

Indeed, a fully comprehensive network would have considerably more than tens of thousands of nodes. As indicated above, insurance companies often speak of procedure/condition or procedure/setting pairs or even procedure/condition/setting triples. For example, Lab Tests performed in a hospital are generally paid in a different manner (different schedule of payment and different regulations) than Lab Tests performed in a doctor's office or home. Expanding the network to include these pairs and triples would mean that the network would have millions of nodes. This would not only yield an unwieldy network, but would also be extremely wasteful: most pairs and triples are not of interest. We deal with this problem in several ways. First, sometimes a pair or triple, if common enough, is given its own node. Thus we have Routine Endoscopic and Sickness Endoscopic. [2] Second, each node is associated with a (possibly empty) list of settings which change the coverage or regulations of that node. For example, the node Lab Tests would have associated with it a list of settings including Hospital and Hospice, since payment for Lab Tests is different in those settings. We discuss this further in section 4.1.

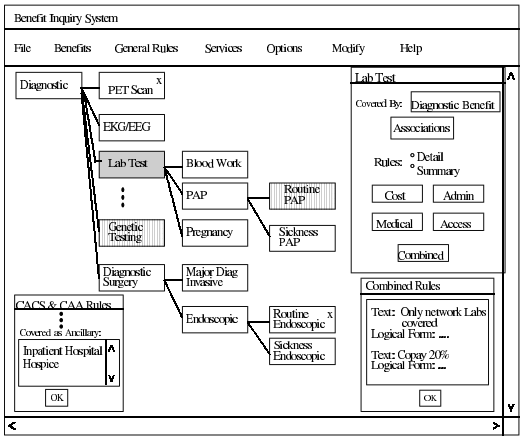

Figure 1: CSR's view of BenInq — Diagnostic Services

3.4 Reasoning Methods

Two types of reasoning are needed for the semantic network:

1. Determining if a node is covered or excluded by some benefit.
2. Determining which rules apply to a node.

The first problem has been studied extensively in the literature [Horty *et al.*, 90; Stein 92]. BenInq implements Stein's (92) algorithm to solve the problem.

The second problem has been studied extensively by [Morgenstern, 96]. A short summary follows. The rules that *apply* to a node are not merely the rules that are *attached* to a node. For example, the rules that are attached to Diagnostic Services should also apply to the node Lab Tests. It might seem that given a node of interest (which we call the *focus node*), we can collect all rules attached to supertypes of that node[3]. But such an approach could lead to rampant inconsistency; it is very possible that one of the rules at Lab Tests is inconsistent with the rules at Diagnostic Services.

The issue of which rules apply to a node in a FAN is known as the *wff-inheritance problem*. The solution is to compute a *maximally consistent subset* (mcs) of all the rules attached to nodes to which there is a positive undefeated path from the focus node. Since there may be

[2] Of course, these are really sets of pairs; Sickness Endoscopic is the set of pairs (Sickness Endoscopic, x) where x stands for one of a large number of conditions for which endoscopic surgery is considered appropriate.

[3] Or more precisely to nodes to which there is a positive undefeated path [Horty *et al.*, 90] from the focus node.

many such maximally consistent subsets, one prefers the *preferred maximally consistent subsets* (pmcs's) based on criteria of specificity and path preference. For example, if there are two mcs's at a focus node, and they differ in that the first has a wff from a more specific node than the second, than the first is preferred to the second.

The algorithm to determine a pmcs at a focus node is given in [Morgenstern, 96]. It calls for first preprocessing the semantic network to remove conflicted and pre-empted edges. Then the network is traversed upward, starting at the focus node. As each node is visited, the node is marked, and a pmcs of the collected wffs along with the new wffs is computed. At each upward branching point, the branches are ordered according to the given prioritization, and the branches are recursively traversed according to that order.

The most computationally intensive part of this process is computing the pmcs, and in particular, determining if a set of wffs is consistent.[4] Our system uses the method of Selman and Kautz (1993) to randomly produce a valid truth assignment to a set of wffs. If it is consistent, this procedure generally works very rapidly. Thus, if after a short time no valid assignment has been produced, the system tries to show inconsistency in a more traditional manner.

4 System Description

BenInq has been implemented in VisualAge Smalltalk on an OS/2 platform. The system incorporates two tools: the *inquiry tool* which is used by CSRs to answer customers' questions, and the *authoring tool* which is used by PMs to modify products. BenInq has two main components, a graphical user interface and a reasoning engine, which are used by both tools. The graphical user interface is used to navigate through the large number of nodes and to display the dags that are rooted at specific benefit or service nodes. It consists mainly of functions to search through the network, draw nodes in the network structure, and display dialog boxes. We exploited the power of VisualAge Smalltalk, which comes with a library of drawing tools (particularly for menus and dialog boxes) and facilitates the creation of graphical interfaces.

The reasoning engine consists of two components: one for performing standard attribute inheritance and one for performing wff inheritance. The attribute-inheritance component uses the algorithm described in [Stein, 1992]. The wff inheritance component uses the algorithm described in [Morgenstern, 1996] and summarized in section 3.4.

The authoring tool uses both components of the reasoning engine as well as the graphical interface. The graphical interface is used to search for the node or portion of the network to be modified. If a node is inserted or deleted, or inheritance information is changed, the attribute-inheritance component recomputes the coverage or exclusion of the relevant nodes. If a wff is added, deleted, or modified at a node, the wff-inheritance component recomputes the wffs that apply to that node and the nodes beneath it in the dag.

[4]Determining consistency is only semi-decidable for the predicate case; it is decidable but intractable for the propositional case.

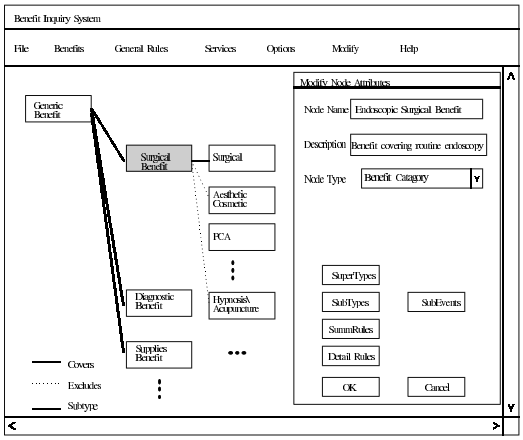

Figure 2: Surgical Benefit before updating

The inquiry tool uses the attribute-inheritance component of the reasoning engine in addition to the graphical interface. Since the wff-inheritance component is by far the most computation-intensive portion of BenInq, we try to restrict its use within the system as much as possible. It is therefore not used by the inquiry tool. Thus, the set of wffs that apply to a node are stored during the authoring process.

Below, we briefly discuss how BenInq is used by its two classes of users, CSRs and PMs.

4.1 A CSR's view of BenInq

BenInq gives a CSR the ability to view various benefits and services by either selecting the appropriate item in one of several pull-down menus (organized by node type) or by using a node search mechanism. This allows the CSR to view all the subtypes of a chosen service or benefit, and to view all top-level services that are either covered or excluded by a chosen benefit. The system also gives the CSR the ability to determine which services are covered or excluded, and by which benefit. Moreover, the CSR can easily determine all the rules that apply to a particular service, including those inherited from its ancestors. The CSR can choose to view rules of a particular sort (e.g., cost-share or access) or all rules. BenInq also allows the CSR to view the *associations* at a particular node. Such associations allow the CSR to see how a procedure's coverage may change depending on the setting in which the procedure is performed.

Figure 1 illustrates how a CSR would use the system to answer questions concerning different Lab Tests. The CSR can view the relevant portion of the network either by first selecting Diagnostic Services using the pulldown menu and then clicking on Lab Tests, or by using the node search mechanism. The screen displayed provides

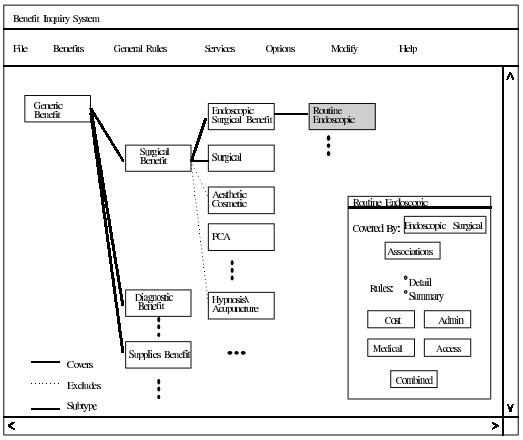

Figure 3: Surgical Benefit after updating

the CSR with much useful information at a single glance. First, it is immediately apparent which nodes are not covered; they have an x in the upper right corner. All other nodes are covered. There are visual cues indicating that a node is not covered by the expected benefit (in this example, Diagnostic Benefit). Thus, for example, the CSR finds out that Routine Pap (which is a different color than other nodes), although a lab test, is not covered by the Diagnostic benefit. To find out which benefit covers a particular node, the CSR clicks on that node. In Figure 1, since Lab Tests is currently selected, a dialog box shows, among other things, that Lab Tests is covered by Diagnostic Benefit. If the CSR clicks on the Routine Pap node, a similar dialog box comes up indicating that it is covered by the Preventive Benefit. Moreover, the CSR is given the option to switch views to a tree rooted at the Preventive benefit so that all information relevant to Routine Pap becomes visible.

In addition, the CSR may view various rules applicable to the node by clicking on the appropriate button. In Fig. 1, the CSR has selected the Combined button, indicating that all rules applying to the service should be displayed. The lower right portion of the screen shows the dialog box that pops us when the Combined button is selected. Note that both the text and logical forms of the rules are given. Similarly, the CSR can view the *associations* at a particular node by clicking on the appropriate button. The lower left portion of Figure 1 displays the dialog box which pops up when the CSR clicks on the associations button.[5] This box lets the CSR know that if lab tests are performed in an inpatient hospital (resp. hospice) setting, the procedure is covered by the Inpatient Hospital (resp. Hospice) Benefit and is subject to those rules. If the CSR wishes to see these rules, she clicks on the desired setting in the dialog box and the appropriate regulations appear.

[5]In reality, only one of the rules/associations dialog boxes can be shown at one time. Both dialog boxes are shown for explanatory purposes.

4.2 A Policy Modifier's view of BenInq

The authoring tool provides the Policy Modifier with several features to effectively modify benefits and services. The PM can add new benefits or services, modify existing services and benefits, and delete existing products. The PM can add a node either before or after an existing node in the network. She can also specify other supertypes, subtypes, and the rules that apply to the new node. For service nodes, the PM can specify whether the service is covered (or excluded), and by which benefit, or whether coverage/exclusion is inherited from another service node. Similarly, the PM can modify various attributes of any existing benefit or service including name, list of supertypes, prioritization of the supertypes, list of subtypes, set of rules that apply to that node, and for service nodes, the coverage/exclusion information and service classification. Finally, the PM can delete services or benefits from the system.

Figures 2 and 3 demonstrate how the authoring tool would be used by the PM to add a new benefit, Endoscopic Surgical Benefit, to explicitly cover the previously excluded Routine Endoscopic Services. Since the new benefit is a form of Surgical Benefit, she adds it as a subtype of Surgical Benefit. Figure 2 shows a portion of the dag displaying the Surgical Benefit. The PM adds a new node as a subtype of Surgical Benefit (by choosing the appropriate option from the Modify menu), and the dialog window pops up. She can then specify the name, type, subtypes, and supertypes as well as the rules that apply to the new node. Figure 3 shows the situation in which the new benefit, Routine Endoscopic Benefit has been added as a subtype of Surgical Benefit. The service node Routine Endoscopic has also been modified to change its coverage/exclusion information so that it is now covered by the new benefit (as opposed to being excluded by Diagnostic Benefit).

5 Discussion

5.1 Evaluating the System

BenInq represents a significant advance in existing technology for benefits inquiry. The departure from a code-based system to a knowledge-based system is a big change for the insurance industry. This is, as far as we know, one of the few large-scale industry applications that explicitly models nonmonotonic reasoning and uses formal nonmonotonic reasoning techniques. This is clearly an advance for the insurance industry, which previously could deal with exceptions only in an ad-hoc manner. It also validates the work of the nonmonotonic reasoning community, which has often been accused of being insular and producing no useful results for the real world.

BenInq allows CSRs to immediately determine whether or not a service is covered and which rules apply to that service. It further provides PMs with an easy method of updating and modifying insurance products. The current version of BenInq was completed after our consulting engagement had ended. The medical in-

surance company received an earlier version of BenInq, which incorporated some but not all of the reasoning methods described in this paper. This version supported standard inheritance with exceptions and attaching wffs to a node, but did not perform wff-inheritance. Thus PMs who modified the rules at a node had to modify the rules at all subtypes of the node as well in order to ensure integrity. Despite this drawback, the earlier version of BenInq received excellent reviews from the CSRs and PMs who used it. They needed virtually no training, surpassing our most optimistic expectations. They were delighted with the ease of finding out information and modifying products.

The desiderata outlined in section 2 were largely satisfied, although the last desideratum, supporting easy updates and modification, was only partly satisfied. In particular, while it is easy for a PM to add, delete, or modify links and nodes in the network, it is not always easy to modify or add wffs. The PM may easily modify the text version of business rules, but cannot modify the logical form of business rules.

This is a direct consequence of the fact that there is no direct mapping between the internal (logical form) and external (text) representations of business rules, as discussed in section 3. This is in direct contrast to all other parts of the semantic network, in which there is a natural, direct mapping between the internal and external representations. Fixing the problem requires the construction of a mapping between the text and logical form representations; this problem is in general difficult and beyond the scope of this work. However, one can identify certain types of rules for which there is a reasonable mapping. We have sketched out mappings for cost-share and access rules, as well as for administrative rules involving temporal and spatial (setting) constraints. The aim is the construction of templates which will allow the PM to directly enter rules in logical form, and will generate a reasonable text equivalent.

What were the alternatives? The success of BenInq was due to two facts: first, the realization that the bulk of the information was taxonomic, but that a significant chunk was not, and second, the construction of a knowledge structure, the FAN, which allowed the representation of both taxonomic and non-taxonomic information, and linkage between the two. Existing benefits inquiry systems had ignored the taxonomic nature of the medical insurance domain. This resulted in systems that were extremely spotty in their coverage (text-based systems) or were unwieldy (codes). Even a more sophisticated rule-based (or production) system that ignored the taxonomic nature of the domain would run into the problem of redundant information and the need for multiple updates. On the other hand, any attempt to force all of the information into a taxonomy would have resulted in highly artificial nodes with no clear semantics.

5.2 Generalizations and Extensions

The techniques described in this paper, while developed specifically for benefits inquiry in medical insurance, can be generalized to other domains as well. The construct of a FAN and its associated algorithms may prove useful in other domains which satisfy the following criteria:

- there exists a large amount of taxonomic information
- there exists a significant amount of non-taxonomic information, conceptually linked to the taxonomic information
- the non-taxonomic information can be mapped into wffs

The construction of an internal representation which closely parallels the external representation is a useful technique for domains where there is a non-technical class of users who must directly manipulate the knowledge structure. This technique is best suited to domains in which there is a natural translation between the user's conceptual model and a knowledge structure which lends itself to efficient reasoning methods.

Examples of such domains include other parts of the insurance industry, such as life, property, and casualty insurance; legal reasoning, especially case law (cases are often organized taxonomically, and different legal rulings are associated with cases); and medical reasoning.

We are currently using these techniques in the development of a system to dynamically configure property and casualty insurance products. In this system, the constructs of composition and subtyping are both important. The methods developed to reason with subtypes are also applicable for reasoning with composition, with appropriate modifications. This indicates that the techniques described in this paper are useful not only in domains in which taxonomic information is central but in domains in which composition is a useful construct. This has the potential to expand the applicability of this work. We hope to report on this research in the future.

Acknowledgements: We are grateful to Lynn Bricking, Kathleen Jamison, Bill Wynne, Tracy Edmonds, Terry Brown, and Kay Krimmer for their assistance in the knowledge engineering phase. An earlier version of BenInq was implemented with the assistance of Mike Ashwell and Jason Springer. Thanks also to Ernie Davis and Benjamin Grosof for helpful suggestions.

References

[Horty *et al.*, 1990] J. Horty, R. Thomason, and D. Touretzky. A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. *Artif. Intell. 42,* 311-349.

[Morgenstern, 1996] L. Morgenstern. Inheriting Well-formed Formulae in a Formula-Augmented Semantic Network. *Proc. KR-96,* 268-279.

[Morgenstern, 1997] L. Morgenstern. Inheritance Comes of Age. Invited Talk, IJCAI-97.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning. *Artif. Intell. 13,* 81-132.

[Selman and Kautz, 1993] B. Selman and H. Kautz. Domain-Independent Extensions to GSAT. *Proc IJCAI.*

[Stein, 1992] L. Stein. Resolving Ambiguity in Nonmonotonic Inheritance Hierarchies, *Artif. Intell.,* 259-310.