RC23436 (W0411-150) November 16, 2004
Computer Science

IBM Research Report

Enhanced Semantic Networks: Hybrid Knowledge Structures for Reasoning

Leora Morgenstern, Erik T. Mueller, Doug Riecken,
Moninder Singh, Leiguang Gong
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Enhanced Semantic Networks: Hybrid Knowledge Structures for Reasoning

Leora Morgenstern, Erik T. Mueller, Doug Riecken,
Moninder Singh, and Leiguang Gong
IBM T.J. Watson Research Center
Hawthorne, NY 10532
leora@steam.stanford.edu; leora, etm, riecken, moninder, leiguang@us.ibm.com

Fall 2004

Abstract

Most formal work on semantic networks has concentrated on inheritance hierarchies, which focus on representing taxonomic information. Although semantic networks can be used to represent general binary relations, their use has been limited because they do not offer a method for determining when a path through the network corresponds to valid reasoning. We introduce a new type of semantic network, called the Enhanced Semantic Network (ESN), that uses regular expressions to formally characterize valid reasoning within a network. We show how one can translate ESNs to theories of first-order logic.

The work described in this paper was motivated by a commercial application: designing an automated recommendation system for banking and insurance sales. We discuss how enhanced semantic networks are used in this application.

1 Introduction

The impetus for this work comes from a problem in the commercial world: recommending products and services to consumers in the domains of banking and insurance. The recommendation task entails various types of reasoning, including determining from a customer's situation what his needs are and what set of actions (usually purchases) might best fit his needs; and reasoning about the hypothetical consequences that his actions would result in.

This paper describes new knowledge structures that were created in order to facilitate this sort of reasoning. In the next section, we describe the business task in more detail. We discuss why semantic networks, though a seemingly reasonable choice for reasoning and representation, are not suitable for this task. We then introduce the *Enhanced Semantic Network* (ESN), which supports

valid reasoning. We provide a translation from ESNs to first-order logic. Finally, we discuss how ESNs are used in the automated recommendation system.

2 The Business Problem

Many banks have transferred much of their operating business from branch offices to online banking web sites. This transformation is straightforward for tasks such as writing checks, but not nearly so simple for buying bank products.

Our goal was to create a system that could simulate the behavior of a human salesperson. Such behavior includes engaging the customer in dialogue to find out what his situation and needs are; making recommendations to the customer for his current needs; discussing hypothetical plans with the customer; and explaining recommendations. The system currently performs all tasks. We focus on the second and third tasks in this paper; we discuss the first and fourth tasks in a forthcoming paper.

For example, a salesperson (SP) might find out that a customer is expecting his first child. This means that he will need to support his child in case he dies, and fund his child's education. The SP could reason about products that serve these needs, such as term life insurance and college savings plans. He would also need to reason about the suitability of products for the customer's situation. For example, if someone earns above $80,000, he cannot contribute to a Roth IRA. The SP may need to reason about future scenarios. If the customer buys an overfunded whole life insurance policy, and makes all payments, he will have a large sum of cash in twenty years' time, and there may be tax consequences. This may trigger new needs.

3 Considering Semantic Networks

A preliminary analysis of the application domain showed that a good part of the knowledge used for the recommendation task could be represented by a limited set of binary relations. These relations on concepts such as situations, needs, and products included *triggering* (a situation triggers a particular need) and *serving* (a product serves a need). In addition, a sizeable chunk of the knowledge, product hierarchies, was taxonomically organized. This suggested that some form of semantic network (SN) might be a good candidate representation.

SNs were also appealing because their graphical structure facilitates the visualization of reasoning. (This is a potential advantage both for customers and domain experts updating the system.) Intuitively, SNs would seem to be useful because paths in the SN correspond to reasoning chains: thus, one can perform reasoning by following paths in the SN.

This intuition is unfounded, however. Arbitrary paths in a SN do not generally correspond to valid reasoning [8]. Consider Figure 1. There are many "meaningful" paths, such as the path from

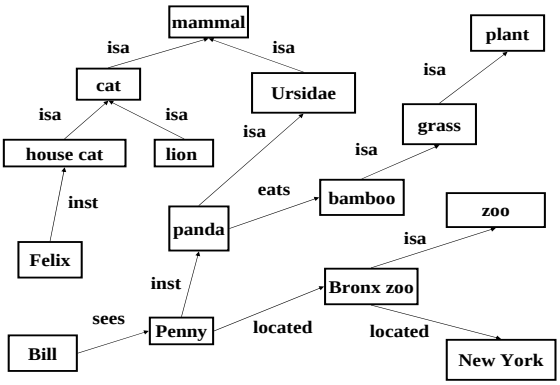

Figure 1: *In a semantic network, only some paths correspond to valid reasoning.*

Penny to mammal. However, many paths don't seem meaningful: there is a path between Bill and plants, but it is unclear what connection there is. In other cases, it is not clear how we draw the correct conclusion: When traversing the path from Bill to New York, how do we know to conclude that Bill is located is in New York? In general there is no real semantics and no clear definition of valid reasoning associated with general SNs.

A restricted class of SNs, inheritance networks (INs), however, has proved very useful. INs(e.g., [2]) allow two links, a set membership link (*inst*) and a subset link (*isa*). This permits correct taxonomic reasoning. An important extension of this work is inheritance networks with exceptions (INEs) [4], which allow cancels links that permit nonmonotonic reasoning.

Indeed, INs, in contrast to other SNs, have been so well studied in AI that they are often what people refer to when they use the term "semantic network" (see, e.g, [7]. They have been widely used in a variety of applications. Nevertheless, because they permit reasoning only about subsumption — that is, whether something is a member of a class, or one class is a subset of another — they are intrinsically limited.

The basic idea of our approach is the following: We note that valid reasoning within a SN generally corresponds to a fixed pattern. For example, taxonomic reasoning is valid (in non-defeasible networks): this means that a path from A to B consisting of isa links (or an inst link followed by isa links) entails that A isa B (or that A inst B). In general, it is possible to identify many valid reasoning paths within a SN by determining whether these paths fit any of a predetermined set of reasoning patterns. We define valid reasoning within a SN by means of these fixed patterns, which we encode as regular expressions. This allows us to formalize multiple types of valid general reasoning within SNs.

4 Developing a Knowledge Structure: Enhanced Semantic Networks

We give a formal definition of an Enhanced Semantic Network (ESN), a structure that allows formal reasoning within a semantic network. We define two types of ESN: the second is augmented by formulas.

Formal definition of ESN-I

An Enhanced Semantic Network I is a tuple (*NT*, *LT*, *SN*, *RT*, *R*, *RGX*, *CLT*, *f*), where

- *NT* is a set of node types $NT_1 \ldots NT_m$
- *LT* is a set of link types $LT_1 \ldots LT_n$
- *SN* is a set of nodes $N_1 \ldots N_p$, each associated with a member of *NT*. Let $\mu(NT_i)$ be the set of all nodes associated with node type NT_i.
- *RT* is a sequence of relationship types, each of which is a triple $(NT_i, LT_j, NT_k) \in NT \times$

$LT \times NT$. Intuitively, a triple (NT_i, LT_j, NT_k) is in *RT* if a link of type LT_j can link two nodes of type NT_i and NT_k.

- *R* is a sequence of relationships $R_1 \ldots R_g$, where the ith element of *R* is associated with the ith element of *RT* in the following way: If the ith element of *RT* is the triple (NT_x, LT_y, NT_z), then the ith element of $R \subseteq \mu(NT_x) \times \mu(NT_z)$. If $(N_a, N_b) \in R_i$, and $RT_i = (NT_x, LT_y, NT_z)$, we say that (N_a, LT_y, N_b) is in the ESN and that R_i is associated with LT_y. We let $RR = \bigcup_{i=1\ldots g} R_i$.
- *RGX* is a set of regular expressions $r_1 \ldots r_s$ over the alphabet $NT \cup LT$. [1] Intuitively, *RGX* comprises the set of reasoning types that are permitted within the ESN.
- *CLT* is a set of conclusion link types $CLT_1 \ldots CLT_t$. *CLT* need not be disjoint from *LT*. Intuitively, this represents the sort of conclusions one can draw when performing valid reasoning with the ESN.
- *f* is a function from RGX to CLT. f gives the mapping between the regular expressions that define the reasoning types and the conclusions that one can draw.

We explain this definition with the help of examples drawn from Figure 2. A special feature of

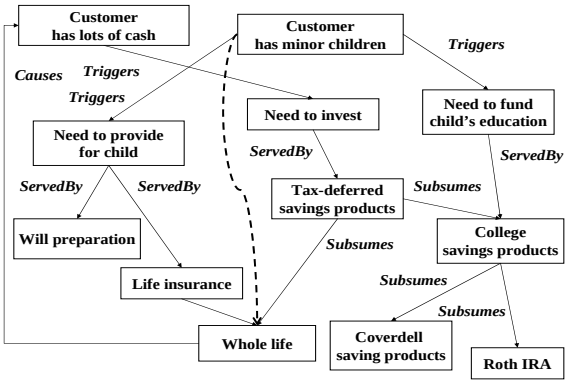

Figure 2: *An example ESN-I E1. The dashed curved link shows the entailed conclusion. Since the path between Customer has minor children and Whole life is legal (matches a regex), one can conclude that the RecommendFromSituation relation holds between the first and last nodes of that path.*

ESNs is that they allows arbitrarily many different types of nodes and links. In Figure 2, there are 3

[1] Strictly speaking, all regexes in *RGX* should begin and end with a node type and alternate node and link types. See p.4 for further details.

types of nodes: *Situation*, *Need*, and *Product (purchase)*. *SN*, the set of nodes, includes: *Has minor children*, *need to invest*, and *life insurance*.
Link types include *Triggers*, *ServedBy*, *Subsumes* (the inverse of *isa*), *isa*, *Generates*, and *Causes*.
RT, which characterizes the permitted relationships in the ESN, includes *(Situation, Triggers, Need)*, and *(Need, Generates, Need)*.

R specifies the actual links in the ESN. For example, R_1, which characterizes the link type *Triggers*, includes the ordered pair *(Has minor child, Need to fund child's education)*. Thus we have the link *(Has minor child, Triggers, Need to fund child's education)*.
RGX consists of the regular expressions characterizing acceptable reasoning within the ESN. For ESN $E1$, RGX includes the regex *(Situation Isa Situation)*Triggers Need ServedBy Product (Subsumes Product)**. The set of concluding link types *CLT* includes the link type *RecommendFromSituation(situation, product)*. The function f maps the regex above to the concluding link type above.

We now give a formal characterization of *legal reasoning* within an ESN. A path in an ESN is an alternating sequence of nodes and links, beginning and ending with a node. [2]
Definition of Legal Paths: Let π be a path in an ESN. We say that π is a *legal path* if π matches some element (regular expression) of *RGX*.
Entailment in an ESN-I: Let E be an ESN and π be a legal path as defined above. Let n_i and n_j be π's initial and final nodes, respectively. For each r in *RGX* that π matches,
$E \models_{ESN-I} (n_i, f(r), n_j)$.

This notion of entailment corresponds to being able to augment the ESN-I by adding a link of type $f(r)$ between nodes n_i and n_j.
Example 1: In the ESN $E1$ of Figure 2, the path *Has minor children - Triggers - Need to fund children's education - ServedBy - Tax Deferred Savings Products - Subsumes - Whole Life (Insurance)* is a legal path. Therefore, $E1 \models_{ESN-I}$ *(Has minor children, RecommendFromSituation, Whole Life (Insurance))*. That is, one can augment the ESN with the link *RecommendFromSituation* between the nodes *Has minor children* and *Whole Life (Insurance)*. The intended meaning of this link is that if a customer has children, one can recommend Whole Life (Insurance) to him.
Example 2: In that same ESN, the path *Has minor children - Triggers - Need to fund children's education - ServedBy - Tax Deferred Savings Products - Subsumes - Whole Life (Insurance) - Causes - Cash rich* is not a legal path.
Example 3: For a standard monotonic inheritance network, in which there are two link types, *inst* and *isa*, there is one node type and two regular expressions: r_1*: node (isa node)** and r_2*: node inst node (isa node)**. For a legal path matching the first regex, one can add an *isa* link between the first

[2] Formally, we define a path as follows:
• If *n1* and *n2* are nodes and *l* is a link such that the triple (*n1*, *l*, *n2*) is in *RR*, then $n1ln2$ is a path. *n2* is the final node of π.
• If π is a path, the final node of π is n_i, and (n_i, l, n_j) is in *RR*, then $\pi l n_j$ is a path.

and final nodes of the path. For a legal path matching the second regex, one can add an *inst* link between the first and final nodes of the path. In other words, $f(r_1) = isa$ and $f(r_2) = inst$. That is, a monotonic inheritance network is just a simple case of an ESN.

4.1 Adding Wffs to the ESN

The ESN-I is useful for reasoning about general classes of customers and recommendations, but is of less value for reasoning about specific circumstances. It would be useful to explicitly reason about a customer's particular context, and to say that some reasoning chains apply only in certain circumstances.

To enable such reasoning, we developed the ESN-II, which allows attaching well-formed formulas (wffs) to nodes and links in the network in restricted ways. We used wffs for two purposes:
1. Defining nodes (*definitional wffs*). E.g., the node *Has minor children* can be defined by the wff *Has-minor-children(Customer)* $\Leftrightarrow \exists x$ *HasChild(Customer, x)* $\wedge$ *Age(x)* $\leq$ *17*.
2. Restricting the set of allowable reasoning paths (*conditional wffs*). This can be done by placing the wff on a link or a node. For example, in Figure 2, one could place a wff on the node *Coverdell savings products* specifying that the customer's income must be below $220,000 to qualify.

Definition of ESN-II The Enhanced Semantic Network-II is a tuple (*NT*, *LT*, *SN*, *RT*, *R*, *RGX*, *CLT*, *f*, *CX*, W_d, W_c, ν, κ, λ), where

- *NT*, *LT*, *SN*, *RT*, *R*, *RGX*, *CLT*, and *f* are as above
- *CX* is a set of wffs, the *context*
- W_d is a set of wffs, the *definitional wffs*
- W_c is a set of wffs, the *conditional wffs*
- ν is a function from nodes to definitional wffs
- κ is a function from nodes to conditional wffs
- λ is a function from links to conditional wffs

Note from the use of functions that we are assuming that there is at most one definitional wff attached to each node, and at most one conditional wff attached to each link or node. (Multiple wffs can always be combined in a conjunction.) We discuss these elements in more detail below.
The Context: *CX*, the context, gives a partial description of the customer who is seeking advice. An example context contains the wffs: $\exists$ *x HasChild(Customer, x)* $\wedge$ *Age(x) = 3* and *Annual-Salary(Customer) = $84,000*. The context may include relevant background information that is not customer specific (e.g., interest rates).

The definitional wffs: W_d, the definitional wffs, have two purposes. First, they give a semantics to the nodes in the network: A node's meaning is specified by its definitional wff. Second, the definitional wffs are used, together with the context, to activate reasoning within the ESN-II. We say that a context *matches* the definitional wff attached to a node if the context entails, in the standard sense, the right-hand-side of the definitional wff. A context always matches a (possibly empty) subset of the nodes in an ESN. The matched nodes are called the *activated* nodes. These nodes are the beginning nodes of the allowed reasoning paths.

The conditional wffs: W_c, the conditional wffs, are used to constrain the set of reasoning paths. Formally, we define the concept of a verified path:

Verified paths: A path π in an ESN-II is a verified path iff for any conditional wff ϕ attached to a link or node of π, CX $\models \phi$.

The paths of interest in an ESN-II are paths that are legal, verified, and begin with an activated node.

Entailment in an ESN-II: Let E be an ESN-II as defined above. Let π be a path with n_i as π's initial node and n_j as the final node of π. If $CX \models n_i$ (π begins with an activated node), π is legal and verified, then for each r in *RGX* that π matches, $E \models_{ESN-II} (n_i, f(r), n_j)$. This corresponds to being able to augment the ESN-II by adding a link of type $f(RGX_i)$ between nodes n_i and n_j.

An example of an ESN-II is shown in Figure 3. This figure depicts a network similar to the one

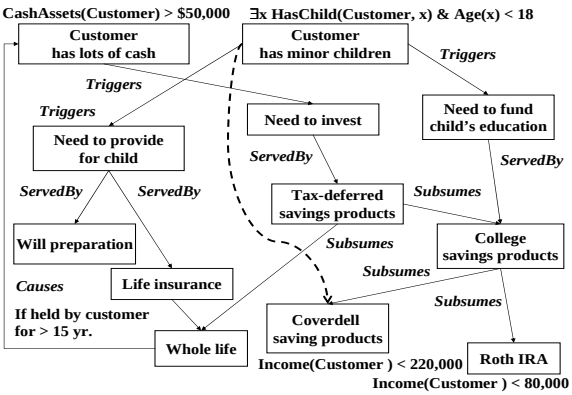

Figure 3: *E2, an ESN-II banking network with wffs. The context is ∃ x HasChild(Customer, x) ∧ Age(x) = 3 and AnnualSalary(Customer) = $84,000.*

in Figure 2, with added context, definitional wffs and conditional wffs. The context is given in the caption. This context activates the node *Has minor children* but not the node *Cash rich*, which is defined as having more than $50,000 in cash. Consider two legal paths beginning with the activated node *Has minor children*: *Has minor children - Triggers - Need to fund children's education - Servedby - College savings products - Coverdell accounts* and *Has minor children - Triggers - Need*

to fund children's education - Servedby - College savings products - Roth IRAs

Both paths are legal, since they match r_1. However, it is not the case that both paths are verified. The first path is verified because the wff attached to *Coverdell accounts* (having an income less than $220,000) is entailed by the context (having an income of $84,000). The second path is not verified because the wff attached to *Roth IRAs* (having an income less than $80,000) is not entailed by the context.

5 Semantics: Translation into First-Order Logic

The semantics for ESNs is given in terms of first-order logic. We can show that reasoning within an ESN is equivalent to deduction in first-order logic. Below, we give a method for translating ESN-I and ESN-IIs into first-order logic.

We assume that each regex is in disjunctive normal form. We further assume that it is in *alternating node-link form* (*ANLF*), that is, it begins and ends with a nodetype, and nodetypes and linktypes alternate. [3]

We define the following mapping between elements of the ESN and elements of a sorted first-order logic $\mathcal{L}$:

• To each node type NT_i in NT, we assign a variable of type x_{NT_i}.

• To each node in N_i in SN, we assign a constant C_{N_i}.

• To each link type LT_i in LT, we assign a predicate P_{LT_i}.

• To each link type CLT_i in CLT, we assign a predicate P_{CLT_i}.

Translating an ESN-I into First-Order Logic

We construct a theory $\mathcal{T} \subset \mathcal{L}$ by placing in $\mathcal{T}$ wffs corresponding to the network (the actual nodes and links in the ESN-I) and to the regular expressions.[4]

For each triple of the form (n_i, l_j, n_k), where l_j is of type LT_j, we add the wff $P_{LT_j}(C_{n_i}, C_{n_k})$.

We can best grasp the intuition behind translating a regex *r* into first-order logic as follows: Consider, first, the simple case: a disjunct of *r* that does not contain the * operator. (Thus, the only operation is concatenation.) This disjunct can be parsed into overlapping substrings of length 3 of the form $n_i \cdot l_j \cdot n_k$. Each such substring can be associated with a literal of the form $P_{l_j}(x_{n_i}, x_{n_j})$. We can then add to $\mathcal{T}$ a wff whose left-hand side is composed of all these literals and whose right-hand side is the predicate corresponding to the concluding link associated with the regex *r*.

Conceptually, the * operator corresponds to recursion. Thus, for each * operator in the regex, we must construct a recursive predicate. In practice, we introduce two predicates for each * operator,

[3] Formally, let N be a letter in the alphabet *NT*; let L be a letter in the alphabet *LT*. *PF* (Pair Format) is defined as follows:

• $L \cdot N$ is in *PF*.

• For any $P_1, P_2 \in PF, P1 \cdot P2$ and $(P_1)^* \in PF$.

• Then a regex *r* is in *ANLF* if $r = N \cdot P$ for any $P \in PF$.

[4] In everything that follows, it is assumed that all variables of wffs added to $\mathcal{T}$ are universally quantified.

one which characterizes the recursive nature of the reasoning, and a second to provide a level of indirection necessary because a *-expression may match a string 0 times. We work from the inside out: that is, we first examine *-expressions of depth 0 (i.e., that do not contain other * operators), then *-expressions of depth 1, and so on.

Formally, we proceed as follows. Consider a regex r_i with at least one occurrence of the * operator. We can rewrite r_i as $s_a p^* s_b$ where p contains no occurrences of the * operator. (r_i must contain at least one *-expression of depth 0.) By construction, s_a cannot be empty. s_b may be empty.

Case I: s_b is not empty: By construction, *p* is of the form $l_2 n_3 l_4 \ldots l_{m-2} n_{m-1} l_m$. Thus, we can rewrite r_i as $s_x n_1 l_2 n_3 l_4 \ldots n_{m-1} l_m n_{m+1} s_y$.

I. We add to $\mathcal{T}$ the following wffs:

(i) $P_{l_2}(x_{n_1}, x_{n_3}) \wedge P_{l_4}(x_{n_3}, x_{n_5}) \wedge \ldots \wedge P_{l_m}(x_{n_{m-1}}, x_{n_{m+1}}) \Rightarrow Q_{i_1}(x_{n_1}, x_{n_{m+1}})$

(ii) $Q_{i_1}(x_a, x_{n+1}) \wedge P_{l_2}(x_{n_1}, x_{n_3}) \wedge P_{l_4}(x_{n_3}, x_{n_5}) \wedge \ldots \wedge P_{l_m}(x_{n_{m-1}}, x_{n_{m+1}}) \Rightarrow Q_{i_1}(x_a, x_{n_{m+1}})$

(iii) $P_{l_m}(x_{n_1}, x_{n_{m+1}}) \Rightarrow Q_{i_2}(x_{n_1}, x_{n_{m+1}})$

(iv) $Q_{i_1}(x_a, x_{n_1}) \wedge Q_{i_2}(x_{n_1}, x_{n_{m+1}}) \Rightarrow Q_{i_2}(x_a, x_{n_{m+1}})$

II. we substitute for the string above the string $s_x n_1 l_{Q_{i_2}} n_{m+1} s_y$.

Case II: s_b is empty: In this case, * operator occurs at the right-hand edge of the regex disjunct. This is easier than the first case. We have fewer wffs to add to $\mathcal{T}$, and we need not add new predicates: we can use the predicate that is associated with the regex. We add to $\mathcal{T}$ the following wffs:

(i) $P_{l_2}(x_{n_1}, x_{n_3}) \wedge P_{l_4}(x_{n_3}, x_{n_5}) \wedge \ldots \wedge P_{l_m}(x_{n_{m-1}}, x_{n_{m+1}}) \Rightarrow f(r_i)(x_{n_1}, x_{n_{m+1}})$

(ii) $f(r_i)(x_a, x_{n+1}) \wedge P_{l_2}(x_{n_1}, x_{n_3}) \wedge P_{l_4}(x_{n_3}, x_{n_5}) \wedge \ldots \wedge P_{l_m}(x_{n_{m-1}}, x_{n_{m+1}}) \Rightarrow f(r_i)(x_a, x_{n_{m+1}})$

We perform this transformation for every *-expression of depth 0, then depth 1, and so on. The substitution above ensures that at any level we need never deal with a nested * operator. After performing these transformations, we can parse the transformed string into overlapping substrings of length 3, and construct a wff as described above.

Example: Assume the regex *Situation Triggers Need (Generates Need) * ServedBy Product* is associated with the predicate *Recommend*. It is translated into the following wffs: (For simplicity, we assume the lowercase spelling of a node type to indicate its associated variable, and use the spelling of a link type as its associated predicate.)

$Generates(need_1, need_2) \Rightarrow Q1(need_1, need_2)$

$Q1(need_1, need_2) \wedge Generates(need_2, need_3) \Rightarrow Q1(need_1, need_3)$

$ServedBy(need_1, product_1) \Rightarrow Q2(need_1, product_1)$

$Q1(need_1, need_2) \wedge Q2(need_2, product) \Rightarrow Q2(need_2, product)$

$Triggers(situation, need) \wedge Q2(need, product) \Rightarrow Recommend(situation, product)$.

Translating an ESN-II into First-Order Logic

Because ESN-IIs have wffs attached to nodes and links, the translation is a bit more complex. Consider a triple of the form (n_i, l_j, n_k). Let l_j be of type LT_j. Let ϕ be a wff attached to l_j, and ψ

be a wff attached to n_k. Then we add to $\mathcal{T}$ the wff $\phi \wedge \psi \Rightarrow P_{LT_j}(C_{n_i}, C_{n_k})$. For each node n_i that has a definitional wff ϕ attached to it, we add to $\mathcal{T}$ the wff $\phi \Leftrightarrow C_{n_i}$.

The translation of regular expressions remains the same.

Soundness and completeness:

Let E be an ESN-I or an ESN-II, and let $\mathcal{T}$ be the first-order theory obtained by performing the translation described above. Then

$E \models_{ESN} (n_i, clt_j, n_k) \Longleftrightarrow \mathcal{T} \models P_{CLT_j}(C_{n_i}, C_{n_k})$.

We have sketched out a proof of this theorem. The core of the proof proceeds by induction on the number of connectives in the regex. The complete proof will appear in a longer version of this paper.

6 Implementation

Since reasoning within an ESN-II subsumes reasoning within an ESN-I, we discuss below only reasoning within an ESN-II.

Reasoning within an ESN-II consists of determining which nodes are activated; for each activated node, finding the legal and verified paths that begin with the activated node; and for each legal and verified path, determining the conclusions that correspond to the relevant regex.

The following algorithm performs the reasoning task within an ESN-II. We assume a deterministic finite automaton $M = (Q, A, \delta, q_0, F)$ defined in the usual way, where $A = NT \cup LT$. In addition to our previously defined notation (p.3), let $D \subseteq Q$ be the dead states of M, G be a function from SN to $\wp(RR)$ (outgoing links), T be a function from RR to SN (destination node), *nodetype* be a function from SN to NT, *linktype* be a function from RR to LT. Then the top-level reasoning algorithm is as follows:

Algorithm 1 *(Reason).*
$\mathcal{O} \leftarrow \emptyset$
for all *node* **in** SN **do**
  if $C \models \nu(node)$ **then**
    $\mathcal{O} \leftarrow \mathcal{O} \cup$ *Reason1*$(M, q_0, node)$
return $\mathcal{O}$

Algorithm 1 invokes the following recursive reasoning algorithm, which takes as input the finite automaton M as defined above, a state q, and node *node*, and returns a set of output nodes $\mathcal{O} \subseteq N$:

Algorithm 2 *(Reason1).*
$q \leftarrow \delta(q,$ *nodetype*$(node))$
if $q \in D$ **then return** $\emptyset$
if $CX \not\models \kappa(node)$ **then return** $\emptyset$
$\mathcal{O} \leftarrow \emptyset$

if $q \in F$ **then** $\mathcal{O} \leftarrow \mathcal{O} \cup \{node\}$
for all *link* **in** $G(node)$ **do**
  if $CX \models \lambda(link)$ **then**
    $q' \leftarrow \delta(q, linktype(link))$
    if $q' \notin D$ **then**
      $\mathcal{O} \leftarrow \mathcal{O} \cup Reason1(M, q', T(link))$
return $\mathcal{O}$

The reasoning system is one component of the implemented system. Our current system engages a customer in dialogue to obtain information, reasons with this information, presents the results of its reasoning to the customer (e.g., its recommendations), and explains its reasoning. Below, we briefly discuss the role of the ESN-II in the system implementation. The system begins with limited information about the customer (harvested from CRM databases), and asks canned questions to obtain a rudimentary profile. This suffices to activate some nodes in the network. The system then traverses potential paths to determine what information it will need. For example, if a wff on a node in a potential path refers to information that the system does not have (e.g., the customer's annual income), the system will ask the customer for this information. (Since network traversal is needed for both obtaining information and reasoning, the implementation executes both tasks concurrently.) The system also uses the ESN to construct an explanation of its recommendation. Consider, for example, Figure 3. The system explains the recommendation of *Coverdell accounts* by following the legal and verified path: noting that the customer has minor children, which means that he has a need to fund his children's education; that this need is met by college savings plans; that a type of college savings plan is the Coverdell savings account, which is suitable for the customer because he earns less than $120,000 a year. The system also explains why certain products are not explained by keeping track of the paths that are not verified. The system generates natural-language (NL) recommendations using the names of the nodes and links, NL glosses of literals of the wffs, and various transformation rules to smooth the whole into NL utterances. A complete description of the explanation generator is given in a forthcoming paper.

We have completed a prototype system that makes recommendations and hypothetical predictions for the domains of mortgages; life, income, auto, and travel insurance; and commercial finance.

7 Related Work

[6] considered using regexes while reasoning in a SN, for the application of story understanding. He constructed an ad-hoc SN representing information in a story and identified "path shapes" corresponding to NL operations; these were candidates for potential inference. Techniques which did not refer to the SN were used to determine which inferences could be made safely. There was no notion of using regexes to characterize valid inference, and little formal development of the theory.

[3] have explored the formal characterization within a SN of partonomic inference, reasoning about parts and wholes. The authors also have some rudimentary analysis of simple compositions of relations within SNs, and the inferences which these compositions permit.

[5] has augmented INEs by allowing one to attach wffs to selected nodes in the network. The attachment has a different semantics than in the ESN-II. A limited amount of non-taxonomic reasoning is permitted, but is restricted locally to only the set of wffs that apply to a node. General SNs are not considered.

Wffs are used to define nodes in description logics (DLs) [1], but differently from ESN-IIs. In DLs, the wffs that define nodes are used to reason about subsumption. In ESN-IIs, defining wffs are used to determine which nodes are activated in a particular context.

8 Ongoing and Future Work

Integration with probabilistic reasoning: The ESN's reasoning is binary: it either does or does not recommend a product. In large domains,in which there are many suitable products, it is often more useful to the customer to order the recommendations and return only the top few.

We have implemented an integration of probabilistic reasoning into ESN-IIs that allows recommendations to be ordered. We modify the ESN-II by attaching weights to a subset of nodes, links, and wffs in the network. Weights may be assigned from the customer's viewpoint (e.g., a product may have greater or less utility for a customer) and/or from the business's viewpoint (a product may be more or less profitable for a business). Methods based on Bayesian reasoning are then used to traverse the network and calculate the weights on recommended product nodes.

Performance issues/ Complexity: Determining the legality of paths in an ESN is computationally efficient: equivalent to recognizing a regex. However, because the ESN-II includes wffs in first-order logic, reasoning within an ESN-II inherits all the problems of first-order theorem proving, including undecidability and intractability. We believe that reasoning within an ESN-II will prove more efficient than standard first-order theorem proving for two reasons: First, only small sets of wffs are reasoned with at any time; second, we use very restricted forms of wffs (not described in this paper) that have excellent computational properties. These restricted forms, however, limit expressivity. We are currently investigating the trade-offs between complexity and expressivity in order to determine which forms are best suited for particular business domains.

References

[1] Ronald J. Brachman, Alexander Borgida, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lori Alperin Resnick. The CLASSIC knowledge representation system or KL-ONE: The next generation. In *Proceedings FGCS-1992*, pages 1036–1043. 1992.

[2] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[3] Udo Hahn, Stefan Schulz, and Martin Romacker. Partonomic reasoning as taxonomic reasoning in medicine. In *Proc. AAAI-1999*, pages 271–276. AAAI/MIT Press, 1999.

[4] John F. Horty, Richmond H. Thomason, and David S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42(2–3):311–348, 1990.

[5] Leora Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103(1–2):237–271, 1998.

[6] Peter Norvig. Marker passing as a weak method for text inferencing. *Cognitive Science*, 13(4):569–620, 1989.

[7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[8] William Woods. What's in a link: Foundations for semantic networks. In Daniel G. Bobrow and Alan Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, New York, 1975.