SNAP: An Action-Based Ontology for E-commerce Reasoning

Leora Morgenstern and Doug Riecken
IBM T.J. Watson Research Center
Hawthorne, NY 10532
leora@us.ibm.com, leora@steam.stanford.edu; riecken@us.ibm.com

FOMI 2005

Abstract

This paper presents SNAP, an e-commerce ontology developed for automated recommendation systems for the domains of banking, insurance, and telephony. This ontology is distinguished by the fact that its core concepts come from commonsense theories of action; its support of derived relations, which rely on composition of basic relations; and its close connection to a sound (with respect to first-order logic) and efficient reasoning mechanism.

We present the basic concepts and relations of the ontology, and explain how derived relations are built using the construction operators for regular expressions. We describe the knowledge structure and reasoning mechanism that are used on instantiations of the ontology to produce customer recommendations. We then discuss how this ontology has been used in practical applications.

1 Introduction

This paper presents SNAP, an e-commerce ontology developed for an automated system for recommending products and services to consumers. The automated system was originally developed for the domains of financial planning and banking and has since been extended for insurance and telephony applications. The same underlying ontology is used in all applications.

This ontology has three distinguishing features:

1. In contrast to many e-commerce ontologies which are primarily organized around the concepts of product and service, SNAP is based on a commonsense theory of agent interaction. Specifically, it is built on basic concepts of standard AI theories of action, such as situations, needs (or goals), and actions, and on various relations betwen these concepts. (SNAP stands for **S**ituations, **N**eeds, **A**ctions, and **P**roducts. As we discuss below, products and services are considered types of actions.)

2. SNAP has two sorts of relations: basic relations, and derived relations, which are built out of basic relations using the construction operators of regular expressions. These derived relations allow great expressivity. In particular, since composition is allowed in derived relations, SNAP has a distinct expressive advantage over ontologies, such as DOLCE [8], built according to specifications of Description Logics like CLASSIC [12], OWL [2].

3. SNAP is accompanied by an efficient reasoning algorithm that is sound with respect to first-order logic.

This paper is organized as follows: In the following section, we discuss the fundamental elements of our ontology, the concepts and the basic relations between these concepts. We then introduce *derived* relations, which are expressed as compositions of the basic relations. We illustrate how derived relations can be represented as regular expressions composed out of the elements of the ontology,

and show how we can use these derived relations to reason about customer recommendations and other e-commerce related tasks.

2 Background: The Business Problem

The business problem for which this ontology lwas developed is the automated recommendation of products and services to consumers. We aimed to develop an ontology that was independent across different business domains.

In line with the methodology outlined in [15], we began our work by constructing various scenarios for product and service recommendations, and analyzed the reasoning needed to provide suitable recommendations. For the domain of financial planning, the first domain that we examined, our scenarios included a young couple planning to fund their children's education, a middle-aged couple planning to retire, and a wealthy person seeking to reduce his tax burden.

Our analysis of these scenarios, and conversations with domain experts led us to hypothesize that the process of recommending products to consumers can best be thought of as a type of commonsense reasoning. The best salespersons operate not by studying their product catalogues and seeing which products they can offer to their customers, but by getting to know as much as they can about their customers, and subsequently reasoning about their circumstances, the needs that these circumstances engender, and the ways that they can help meet these needs. The salesperson succeeds because meeting these needs usually entails buying some products. The reasoning process, however, is primarily about the customer and his circumstances, rather than a set of products. This suggests that there are some basic ontological constructs that carry across different domains.

Moreover, the reasoning involved is closely tied to reasoning about actions and plans. The salesperson or customer service representative suggests a plan of action that meets his customer's goals. He must engage in hypothetical reasoning, and consider the short-term and long-term consequences of particular actions (such as redistributing one's portfolio).

We have set up our ontology so that it can support this sort of reasoning. To do this, we have chosen as the fundamental concepts of our ontology the commonsense concepts of situations, fluents, needs, and actions, rather than the products and services offered by a company. As we demonstrate below, this choice leads to a flexible ontology and reasoning system.

3 Ontology: Basic Elements

The essential concepts in SNAP are taken from AI theories of action. We borrow from the situation calculus and event calculus [9] [11] [6] the concepts of situations, fluents, and events (or actions), and from theories of planning [4] and of desire and intention [3] the concept of a goal or need.

The concept of an agent is not explicitly addressed in this version of the ontology. Instead, each concept in the ontology refers implicitly to agents: it is an agent who is in a particular situation, who has a need, who performs an action. However, there is no representation of an agent as a distinct concept. This is a suitable representation for our current purposes, since multi-agent planning is absent or minimal.

Situations and fluents A situation [9] is a time slice of the world: it describes the way the world is at a particular moment in time. We are interested in the facts that are true in a particular situation. As in the situation / event / fluent calculus, we speak of a *fluent f* being true in situation *s—Holds(s,f)—* to capture this notion.

Examples of fluents are an agent having minor children, an agent having a car, and an agent owning a home. A single fluent generally matches, or holds in, many situations. (There are many situations, for example, in which a particular person has children.) Indeed, a fluent can

be conflated with the set of situations in which the fluent holds. [1] Moreover, a single situation can match many fluents, since many facts will hold true in any situation.

There are several important types of fluents, enumerated below.

Life Stages Life Stages characterize the fluents that describe some major stage of a person's life. Examples are being a child, being an adult, being a parent, and being a retiree.

Age Life Stages: Child, Teenager, Adult, Middle-Aged

Career Stages: Starting a job, Owning a business, Being a retiree

Family Stages: Parent, Parent of minor children, Empty nester

Demographics: These include such facts as marital status, income, and education.

Life Style: These include a person's habits, such as living expensively, living frugally, or taking frequent vacations. A life style actually holds over intervals of time; however, we note that it is common to refer to a person's life style at a particular moment in time. For our purposes, therefore, we consider life styles to be fluents.

Obligations: Obligations include financial and non-financial commitments. Examples of financial commitments include a person's outstanding loans. Examples of non-financial commitments include a commitment to take care of an aging parent and a guardianship.

Needs A need represents something desirable or useful which an agent does not have. It is quite similar to the standard AI concept of a goal. An agent may not be aware that he has a particular need. (Part of the task for an automated reasoning system is making these implicit needs explicitly known to the customer.) Examples of needs include a need for a car or a need for tax-free investments.

Events: An event is defined as a noteworthy happening or occurrence. Any event can be considered either as an action or a behavior. Actions are those events which are planned; behaviors are those events which are observed. A single event can be both a behavior and an action, depending on the system context. If the event is actively planned by the system, it is considered an action; if it is merely observed by the system, it is considered a behavior. This paper does not focus on behaviors; therefore, we will often use interchangeably the terms *event* and *action*. In addition, we will not discuss relations and reasoning types that apply to behaviors, such as abduction.

Below are some types of important events.

Life events: These are noteworthy events that happen over the course of a person's life.

Major life events: Examples include getting married, having children, buying a house, starting a new job, and retiring.

Minor life events: Examples include buying a car or taking a vacation.

Eventualities: Eventualities are those life events which one must plan for. They are usually negative. Examples include death, illness, disability, and the usual range of disasters, such as fire, robbery, hurricanes, and earthquakes.

E-commerce Events: These are events that are directly connected to e-commerce transactions.

Purchasing a Product or Service: An e-commerce transaction can involve several different sorts of events. The most common of these is buying a product or service. The events of buying a product or service can be organized hierarchically: e.g.,

[1] This explains a certain looseness in notation: one often speaks of an event causing a situation, when one really means that an event causes a fluent. If a fluent is equivalent to a set of situations, however, one can as well speak of an event causing a set of situations.

getting a mortgage is a subtype of getting a loan; getting a loan is a subtype of performing a bank transaction. There is, in fact, an isomorphism between the taxonomy of product purchase events and the product taxonomy.

Receiving a Feature, Making a Selection, and Choosing an Option: A product purchase generally has associated with it actions involving such features, selections, and options. Features, selections, and options are distinguished in the following way. One has no choice about the features that come with a product: they are given away "for free." For example, a mobile phone plan may come with free voice mail; one need not do anything to get this feature. In contrast, one decides whether or not to have an option. For example, one may decide to have windshield coverage as part of one's automobile insurance. Selections are similar to options, but are multi-valued rather than binary. For example, one can decide whether to have a 15, 25, or 30-year term for one's mortgage.

There are events associated with features, options, and selections: one *receives* a feature, *makes* a selection, and *chooses* an option. These events can be arranged hierarchically. In the same way that there is a direct isomorphism between a product taxonomy and the taxonomy of product purchases, there are direct isomorphisms between, respectively, feature and receiving-a-feature taxonomies; option and choosing-an-option taxonomies; and selection and making-a-selection taxonomies.

3.1 Relations

We distinguish between *basic* relations and *derived* relations. Derived relations are composed from concepts and basic relations using regular expression constructors, as described in section 3. Derived relations correspond to reasoning types, as we discuss below .

3.1.1 Basic Relations

Except for the subsumption relation, all basic relations in SNAP have to do, in some way or other, with action, planning, and the passage of time. We group the non-subsumption relations into three types: (1) *planning relations*, which capture how goals or needs interact with situations, fluents, and events; (2) *causal relations*, which capture the way the world changes through action or the passage of time; and (3) *constraint relations*, which capture the static constraints, as in [14], among fluents, among events, and between fluents and events.

A note about the label *planning relations*: It is of course the case that one needs to reason about causation and static constraints while planning. Nevertheless, planning relations are distinguished by the fact that they are used primarily for planning purposes, as opposed to causal and constraint relations which are used, as well, to reason about prediction (what will happen in the future), postdiction (what happened in the past / explanation), and how fluents are related.

We discuss these relations in more detail below.

Planning relations: There are four major types of planning relations.

Triggers(Fluent, Need): A fluent often engenders a particular need or goal which an agent wishes to achieve. For example, the fluent of having minor children triggers the need to provide for one's children.

ServedBy(Need, Event): Intuitively, this relation holds when performing a particular action would result in the goal or need being satisfied. For example, the need to improve one's health is served by the action of exercising; the need to provide for one's children is served by the action of purchasing life insurance.

Generates(Need, Need): This is similar to the planning notion of subgoal generation. One need can generate, or give rise, to another. For example, the need to purchase a house generally gives rise to the need to finance that purchase.

Anticipates(Fluent, Need): If a fluent holds in a particular situation, that can mean that sometime down the line, a need may be triggered. For example, the lifestage of being an adult anticipates the need to retire comfortably. This is different in nature than Triggers in the sense that Triggers talks about a fluent that holds in the current situation and a current actual need; Anticipates talks about a fluent that holds in the current situation and a future likely need.

Causal-type Relations The relations discussed below have the flavor of causation, in the sense that they describe relations that frequently develop along with the passage of time and/or the occurrence of events.

Causes(Event, Fluent): This is the standard notion of causation, in which performing an action causes some fluent to hold in the situation resulting from the action. For example, buying a home causes the fluent of owning a home to be true.

DevelopsAfterTime(Life Stage, Life Stage): This relation holds between life stages. Often lifestages lead to other lifestages simply through the passage of time. For example, a baby becomes a child after the passage of some time; likewise a child becomes an adult through the passage of time.

LeadsTo(Life Event, Life Event): Often one event will frequently lead to another. For example, when one switches jobs to a new locale, one will frequently, but not necessarily, move. Similarly, when one gets married, one frequently moves.

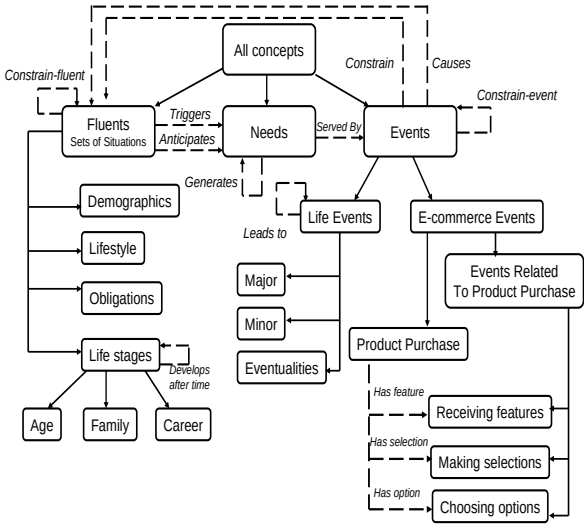

Figure 1: *The schematic ontology diagram for SNAP. Subsumption relations are shown with solid lines; non-subsumption relations are shown with dashed lines.*

Constraint-type Relations: The relations discussed below have the flavor of domain constraints [7] [14] rather than causation. However, these constraints do not necessarily hold only between fluents ; they can also hold between two events, or between events and fluents.

Constrain-fluent(Fluent, Fluent): This is similar to the standard notion of domain constraint. When one fluent holds, another may be constrained to hold as well. For example, in certain circumstances, if the fluent of being above the age of 70 holds in a situation, tht can entail that the fluent of being a retiree holds in the situation as well.

Constrain(Event, Fluent): The occurrence of an event may *constrain* a fluent to be the case, event though the event does not *cause* the fluent to hold. For example, if one takes out a loan to finance one's business that means that one must own a business to start with or have detailed plans to start a business.

Constrain-event(Event, Event): The occurrence of one event may constrain other events to take place (concurrently or otherwise). For instance, if one takes out car insurance, then one must have at one point bought a car, or one must be buying a car now.

Relations having to do with product purchases: An important subrelation of the Constrain-event relation specifies the connection between product purchases and receiving features, making selections, and choosing options. Specifically, when one purchases a product, one is constrained to receive certain features (e.g., for travel insurance, free roadside assistance), to make certain selections (e.g., the size of one's deductible), and to decide on certain options (e.g., coverage for skiing-related accidents).

Because in common discourse we will conflate the hierarchies of product purchase, receiving features, making selections, and choosing options with the hierarchies of respectively, products, features, selections, and options, we will denote these relations as, respectively, *Has Feature*, *Has Selection*, and *Has Option*.

Figure 1 is a schematic ontology diagram showing the basic ontology types and relations between these types.

4 Derived Relations and Reasoning Types

From the point of view of practical commonsense reasoning, the most important relations in the ontology are the derived relations. These are the relations that are defined the basic building blocks of the ontology; they correspond to types or reasoning, in a sense to be made precise below.

We add to the ontology a set of definitions $li = di$, where each li is the name or label of the derived relation and di is a derived-relation expression (DRE). We define DREs as follows:

Let NT and LT be, respectively, the set of concept types and basic relationship types. (NT and LT stand for node types and link types.) A DRE is a regular expression over the alphabet $NT \cup LT$ in *alternate concept-relation (node-link) form*: it begins and ends with a concept type and concept types and relation types alternate. We have the following formal definition:

- if $n \in NT$ and $l \in LT$ then $l \cdot n$ is an LT-pair
- if $x1, x2$ are LT pairs, then $x1 \cdot x2$, $x1|x2$ and (x1)* are LT pairs
- if x is an LT pair and $n \in NT$ then $n \cdot x$ is a DRE

When no ambiguity results, we will drop the $\cdot$ from DREs.

A DRE corresponds to a path through the graph or network induced by an instance of the ontology.

Some examples follow. For ease of reading, we will use the term Product instead of Product Purchase.

Example 1: *R1 = Fluent Triggers Need ServedBy Product .* *R1* is thus the composition of the Triggers and ServedBy relations. *R1* is a very rudimentary Recommendation relation. It specifies

that if some fluent triggers a need, and that need is served by some product (purchase), then it is reasonable to recommend that product (purchase) for an agent in the situation where the fluent holds.

Example 2: *R2 = (LifeStage (DevelopsAfterTime LifeStage)* | Fluent) Triggers Need (Generates Need)* ServedBy Product (Subsumes Product)**. This is useful for reasoning about suitable products to recommend for the needs triggered by the fluents that hold in one's current and future situations. We call this relation *RecommendFromFluent*.

Example 3: *R3 = Product Causes Fluent Triggers Need (Generates Need)* ServedBy Product (Subsumes Product)*. R3* is a type of cross-sell relationship. It corresponds to determining what fluents are caused by a product purchase, and then determining what products might be recommended for that circumstance. For example, purchasing a mortgage results in a situation in which one has large financial obligations each month. Such large obligations may trigger several needs, such as a need for protection to ensure that one will be able to make the payment, and a means for making the payment with as little hassle as possible. These needs, in turn, are served by, respectively, mortgage insurance and automatic check payment services. Thus, one can reason that reasonable cross-sells for mortgage products are mortgage insurance products and automatic check payment services.

The derived relation *R2* or *RecommendFromFluent* is shown in Figure 2. Note that the path *Has minor child - Triggers - Need to fund child's education - ServedBy - Tax Deferred Savings Products - Subsumes - Whole Life* matches the DRE on the right-hand side of *R2*. Intuitively, this means that one can add the link *RecommendFromFluent* between the first and last nodes on the path, that is, between *Has minor child* and *Whole Life*. In other words, when we find a path that matches a DRE, we can add that derived relation as a link between the first and last nodes in a path. Finding this path corresponds to reasoning that the relation *RecommendFromFluent* holds between the fluent *Has minor child* and the product (purchase) *Whole Life*, meaning that Whole life insurance is a suitable recommendation for someone who has a minor child.

In previous work, we have shown that finding paths that correspond to these predefined regular expressions is indeed sound with respect to first-order logic. These results are briefly summarized below.

4.1 Reasoning with SNAP: Enhanced Semantic Networks

In previous work [10], we introduced the *Enhanced Semantic Network* or ESN. The ESN differs from standard semantic networks in its ability to characterize a wide range of sound reasoning. [2] In this section, we define ESNs, define entailment within an ESN, and discuss soundness.

Definition: An Enhanced Semantic Network is a tuple (NT, LT, SN, RT, RR, RGX, CLT, f) where NT is a set of node types; LT is a set of link types; SN is a set of nodes, each associated with a member of NT; $RT \in NT \times LT \times NT$ is a set of relationship types; RR is the set of all nodes and links in the ESN; RGX is a set of regular expressions over the alphabet $NT \cup LT$, in *alternating node-link form*, as defined in the previous section, intuitively comprising the set of reasoning types that are permitted within the ESN; $CLT \subseteq LT$ is a set of conclusion link types, intuitively, representing the conclusions one can draw when performing valid reasoning within the ESN; and f is a function from RGX to CLT, giving the mapping between the regular expressions that define the reasoning types and the conclusions that one can draw.

The correspondence between ESNs and instances of an ontology in SNAP are obvious. NT and LT give the concept (node) and relationship (link) types; SN gives the actual set of nodes in an instantiated ontology; RR gives the links between these nodes; RGX gives the DREs; CLT gives the

[2] Semantic networks can be divided, roughly, into two groups: those that have a formal semantics and those that do not. Those that have a formal semantics, such as standard inheritance networks and description logics [1], limit expressivity and reasoning power, sometimes severely: even relatively expressive DLs like OWL [2], for example, do not permit composition among relations. Those that do not have a formal semantics do not of course in general permit sound reasoning [16].

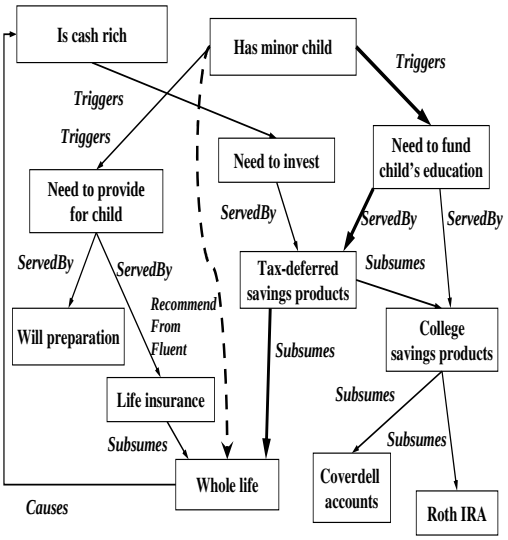

Figure 2: *ESN E1. Since the path between* Has minor child *and* Whole life *(indicated by heavy arrows) is legal, one can conclude that the* RecommendFromFluent *relation holds between the first and last nodes of that path.*

derived relation names.

Figure 2 shows an example of an ESN.

4.2 Entailment in an ESN

What differentiates an ESN from an ordinary SN is its notion of entailment. Entailment within an ESN is defined in terms of *legal paths*. In turn, legal paths are defined with respect to a predetermined set of regular expressions, the *RGX* of the ESN. A path is legal iff it matches some element r of *RGX*. If a path is legal, one can *augment* the ESN by adding a link between the first and last nodes of the path, where the link is *f(r)*. Furthermore, we say that the new triple in the network is a *consequence* of the ESN. Entailment is then defined as closure under this notion of consequence.

The formal definitions follow:

Definition(Paths): An ESN path π is an alternating sequence of nodes and links, beginning and ending with a node.

Definition (Legal paths): π is a *legal path* if π is a path and π matches some element (regular expression) of *RGX*.

We introduce the following notation to express augmenting an ESN by adding links and possibly nodes:

Definition (Augmentation of an ESN): If T is a set of tuples (n_a, l_y, n_b) then $E \cup_{aug} T$ represents the ESN obtained by replacing *RR(E)* with *RR(E)* $\cup$ T.
Note that $E \cup_{aug} T$ is also an ESN.

Definition (One-step entailment): If E is an ESN-I, *Cons-1-step(E)* $= E \cup_{aug} \{(n_i, f(r), n_j) \mid \pi$ is a legal path in E and r is a regex in *RGX* matching $\pi\}$.

Thus, 1-step entailment corresponds to adding to the ESN-I a link of type $f(r)$ between nodes n_i and n_j.

We can now define *Cons(E)*, the consequences of an ESN, as the ESN closed under one-step entailment.

Definition (Closure under one-step entailment):
Cons(E) is the smallest set satisfying the following properties:

1. *Cons-1-step(E)* $\subseteq$ *Cons(E)*
2. *Cons-1-step(Cons(E))* = *Cons(E).*

We then define entailment in an ESN E as follows:

Definition (Entailment): $E \models_{ESNI} (n_a, l_y, n_b) \iff (n_a, l_y, n_b) \in RR(Cons(E))$

That is, a path of length 3 is entailed by an ESN E if the corresponding triple exists in the ESN representing the consequences of E.

Returning to the ESN *E1* in Figure 2: Suppose *RGX* of *E1* contains the regular expressions $R1 \ldots R3$ defined in section 3.1. Furthermore, suppose that f(R2) is equal to the derived relation /reasoning type *RecommendFromFluent*. Since the path *Has minor child - Triggers - Need to fund child's education - ServedBy - Tax Deferred Savings Products - Subsumes - Whole Life* is a legal path matching *RecommendFromFluent*, therefore, $E1 \models_{ESNI}$ *(Has minor child, RecommendFromFluent, Whole Life).* Thus, one can augment the ESN with the link *RecommendFromFluent* between the nodes *Has minor child* and *Whole Life*.

In our previous work, we have shown that reasoning within an ESN using this notion of entailment is sound with respect to first-order logic. The proof is by construction —we provide a translation from an ESN into a subset of FOL— and induction on the connectives in the regexes of the ESN. However, reasoning is much more efficient in the ESN than in FOL, since the core of the reasoning depends on recognition of regexes.

5 SNAP in Practice

5.1 Using Snap in Business Domains

We have used the SNAP ontology in a variety of domains including banking, financial planning, insurance, and telephony. We developed the basic ontology prior to any customer engagement, through an analysis of the financial planning domain, using public, non-proprietary sources, including textbooks, publicly available product and service catalogues of various companies, and various materials on the web. Using these resources, we developed a generic ontology for financial planning.

The resulting ontology contained all the basic constructs (fluents, needs, actions) discusssed in Section 3. The instantiation of the ontology contained two distinct though closely intertwined parts: the domain-independent concepts and relations, and the domain-specific concepts and relations. Examples of the former included the hierarchy of age life stages and family stages. Examples of the latter included the hierarchy of types of life insurance and college savings plans.

Our initial customer engagement was with a multi-national bank, for the development of a prototype recommendation system for mortgages. A significant part of the project involved the development of the ontology. (Other parts included the implementation and the integration with a dialogue system; see [10]). This was accomplished using a combination of knowledge engineering technology (see, e.g., [13]) and the methodology outlined in [15], section 6.1, modified to take account of the fact that we were iteratively expanding the ontology.

In collaboration with domain agents, we constructed a set of motivating scenarios. [3] We expanded

[3]Note that the second step of Figure 5 of [15], the formulating informal competency questions, was much abbreviated as we expanded the ontology, since we had so much formal terminology already in place.

the formal ontology as necessary, characterized the reasoning types necessary to accomplish the desired recommendation tasks, formally specified the relations between the elements of the ontology, and showed that the resulting ontology could indeed support the necessary reasoning tasks.

This process was repeated as we extended the preliminary recommendation system to other domains, specifically insurance (travel, car, income, mortgage) and small business loans. We are now working with a second customer, in the domain of telephony, to design and develop a system for the recommendation of cell phone service and equipment, and have repeated the process in this case as well.

These ontologies lie at the core of the prototype recommendation systems that we have developed. These prototypes are scheduled for deployment in the near future.

5.2 Evaluation

In spite of the plethora of work relating to evaluating ontologies [4], there exists no definitive standard for ontology evaluation. This is due partly to the fact that there is no consensus on what is actually meant by the term "ontology" and partly to the fact that one can evaluate an ontology in many different ways, not all of which are relevant to a particular task or application. [5]

Ontology evaluation may also be limited by the size of one's ontology and the maturity of the projects that use this ontology. In general, one needs a large number of test cases in order to perform evaluations; in our work, which is still in preliminary stages, these do not yet exist.

We believe that for this work, the following evaluation metrics are particularly relevant: Adequacy of capture of domain knowledge; Usefulness of ontology for application; and Reusability of core ontology in iterated expansions of ontology. We can make the following preliminary remarks for each of these categories:

(1) *Adequacy of capture of domain knowledge:* We make two remarks here: First, we have succeeded in using our ontology to represent the knowledge needed for our initial test scenarios, as well as for sets of new scenarios that have since been introduced. Second, the customer, shown the ontology in a natural language form, as well as in graphical form, has commented that we have not only captured their business, but that we have helped them understand it better than before.

(2) *Usefulness of ontology for application:* The ontology is a central part of the recommendation system. Indeed, reasoning about recommendations using this ontology, augmented with the proper reasoning types, has proved to be a rather straightforward task. Reasoning reduces to finding a legal path in the graph induced by the ontology. As discussed in [10], we have also used the ontology in the dialogue portion of the recommendation system: paths in the network are used to guide questions and to explain recommendations.

(3) *Reusability of core ontology in iterated expansions of ontology:* As we have examined new domains and expanded our ontology, we have noticed the following phenomenon: Naturally, we must build new product, feature, selection, and option hierarchies for each new domain. However, there is a core ontology — the portion of the ontology that we labelled as domain-independent — that has been virtually unchanged. The core model — of how agents live, grow, and interact —- has remained substantially the same. Moreover, the core model is relatively small, numbering less than a thousand nodes. Indeed, some portions of the model, such as life stages, major life events, and minor life events are very small, measuring only a few dozen nodes each. We believe that this indicates that there is a relatively small core of commonsense reasoning that can be re-used for a variety of business domains, and that viewing the core in terms of standard AI theories of action has proved to be a

[4] [5] gives a good survey of this area.

[5] For example, the metrics of precision and recall are very important for Information Extraction but not particularly relevant for applications requiring deep reasoning; thus, a method of ontology evaluation that gives great importance to precision and recall would be suitable for our ontology. Similarly, evaluation methods meant to be used to evaluated ontologies generated through text mining are not necessarily suited to evaluating hand-constructed ontologies.

fruitful and robust approach.

Finally, we note that we have throughout followed the methodology of [15], which incorporates evaluative techniques at various points during the ontology creation process.

6 Future Work and Concluding Remarks

We began this work with the hypothesis that much e-commerce reasoning is largely general commonsense reasoning, rather than special purpose reasoning tailored to a particular domain. Our experience with multiple domains has largely borne this out. Our initial application for a customer was in a subdomain of banking, for recommending mortgages. We have since developed new tracks of our recommendation system for income, travel, and auto insurance, small business loans, and cell phone service and equipment. As discussed above, our experience has been that most of the core ontology remains unchanged for each new domain introduced, which gives confiming evidence for our hypothesis.

We are planning to extend our work in the following ways. First, we are currently expanding our ontology so that it can support explicitly representing and reasoning about multiple agents. Second, we are planning to focus in greater detail on issues of abduction—determining potential causes given an observed customer behavior. We believe that our underlying ontology of situations, fluents, needs, and events is sufficiently powerful to handle abduction. In particular, we have constructed regexes that correspond to very simple cases of plan recognition. Further research is needed to expand and validate these results. Third, we would like to explore connections with other foundational ontologies, such as DOLCE.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah McGuinness, Peter Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference, 2004. http://www.w3.org/TR/owl-ref/.

[3] Michael E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[4] Richard Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[5] Jens Hartmann, Peter Spyns, Alain Giboin, Diana Maynard, Roberta Cuel, Mari Carmen Suarez-Figueroa, and York Sure. Methods for ontology evaluation, 2005. KnowledgeWeb Deliverable D1.2.3.

[6] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[7] Fangzhen Lin. Embracing causality in specifying the indirect effects of action. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 1985–1993, 1995.

[8] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, and Luc Schneider. The wonderweb library of foundational ontologies: Preliminary report, 2003. WonderWeb Deliverable D17.

[9] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[10] Leora Morgenstern, Erik Mueller, Doug Riecken, Moninder Singh, and Leiguang Gong. Enhanced semantic networks: Hybrid knowledge structures for reasoning, 2004. IBM Research Report RC23436.

[11] Raymond Reiter. *Knowledge in Action*. MIT Press, Cambridge, Massachusetts, 2001.

[12] Lori Alperin Resnick, Alex Borgida, Ronald Brachman, Charles Isbell, Deborah McGuinness, Peter Patel-Schneider, and Kevin Zalondek. CLASSIC description and reference manual for the COMMON LISP implementation, version 2.3, 1995. http://www.ida.liu.se/ TDDA13/labs/desclogic/manual.pdf.

[13] Mark Stefik. *Introduction to Knowledge Systems*. Morgan Kaufmann, San Francisco, 1995.

[14] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89:317–364, 1997.

[15] Michael Uschold and Michael Gruninger. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, 11(2), 1996.

[16] 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.