Using an OWL Repository for Inductive and Abductive Learning

Andrew CRAPO
GE Global Research, 1 Research Circle
Niskayuna, NY 12309, USA

Amy ARAGONES
GE Global Research, 1 Research Circle
Niskayuna, NY 12309, USA

Jeanette BRUNO
GE Global Research, 1 Research Circle
Niskayuna, NY 12309, USA

Marc GARBIRAS
GE Global Research, 1 Research Circle
Niskayuna, NY 12309, USA

ABSTRACT

The Adaptive Work-Centered User Interface Technology (ACUITy) program and architecture uses semantic models captured in Web Ontology Language (OWL) repositories as the basis for adaptive, extensible human-computer interfaces in open-ended problem solving domains. Central to the adaptive nature of ACUITy is its ability to implement several forms of learning. In this paper we will explore different kinds of learning and describe how the ACUITy implementation has realized some of these learning methods and enables future implementation of others. ACUITy will soon be released to Open Source with the hope that others will find this technology useful and will contribute to its future improvement.

Keywords: learning, adaptive user-interface, ontology, semantic modeling, decision support.

1. INTRODUCTION

Over the past three years, we have developed Adaptive Work-Centered User Interface Technology (ACUITy) using a model-based approach [1,2,3]. More specifically, we have constructed semantic models of the user, the human-computer interface, and of problem solving work domains using the Web Ontology Language (OWL) [4] as the model representation paradigm and Jena as the repository (see http://jena.sourceforge.net/). Our most extensive modeling work to-date has been of the human-computer interface, where the model provides both a shared semantics through which dialogue between the computer and the human can occur and a framework for easy extension to domain-specific applications. The ACUITy framework not only models the human-computer interface, it also "executes" the model, in essence, implementing the modeled application. Bundled in the human-computer interface model is the ability to adapt the content and screen layout to user preferences and the type of problem that the user is attempting to solve. For example, the presentation (rendering) of information via graph and table displays is in the basic model. Bundled in with these presentations is the ability for the user to change the coloring, layout, etc. of any rendered graph or table at runtime. The system accumulates instance data from changes the user does or does not make to the display: for example, changing chart types or colors, resorting or hiding data in a table, or bringing new information into the focus of the display.

During the course of our work, we have explored the use of an OWL model as the basis of learning from instance data using several different algorithms and have hypothesized the usefulness of other kinds of learning. In this paper we report on both what we have done and what we think is feasible and worth doing in the future. We find semantic models to be a natural framework in which to learn in open-ended, decision support environments.

2. A CONCEPTUAL MODEL OF LEARNING

Human learning is based on sensory perception. From these perceptions are derived increasingly complex mental models, including categorizations [5,6,7,8]. Learning includes natural language constructs, and at some point we are able to learn indirectly through communication with other people and not just from our own first-hand observations. Such second-hand learning depends upon shared semantics—upon largely aligned ontologies in the minds of the participants.

Computational models--models externalized in some artifact and processed by a machine to produce an output--are [at least initially] constructed by people and are based upon their mental models. The semantic models underlying a computational model may be more or less explicit. The vision of the Semantic Web is to make the semantics of information on the Web explicit—comprehensible to both people and computers. To this end, OWL supports ontologies that are modular and extensible and in this sense are similar to people's mental ontologies; each has his or her own version that is largely an extension of a shared ontology "owned" by a larger community.

In this context, we discuss several different kinds of learning that can occur over an ontological model. The simplest has to do with analyzing instance data. Given homogenous data (that is, attributes of instances of some meaningful grouping, e.g., the weights of apples in a basket), we can perform various

computations over the data. We might do statistical analysis, e.g., determine the average weight and standard deviation of the apples in a basket or determine the median weight of the apples in the basket. The grouping criteria can of course vary. Rather than apples in a basket, it might be apples from Washington State, or apples sold by Wal-Mart during January 2004. This kind of learning from instance data is a form of logical induction (not to be confused with mathematical induction, which is deductive in nature).

In our analysis of ACUITy instance data regarding human-computer interactions, we have so far found the following to be useful:

1. the average value of a numerical attribute of a group of instances
2. the most frequent value of an attribute of a group of instances
3. the most recently used/specified value of an attribute of a group of instances

We have primarily used these computed values as the learned default values to be assigned to attributes of a new instance of the specified group (class). For example, if this decision maker has most frequently preferred to see trend data regarding mean time between failure as a line graph in the upper-left corner of a particular display, then a reasonable default is to encode a new trend data set as a line graph and display it in the usual location for that type of display.

Another type of inductive learning is the extension of an ontological category to finer-grained subclasses. This is illustrated by the way that children learn [9]. To a child learning to talk, all animals often are initially grouped into a single class. With experience and the prompting of devoted tutors, the animal class (whatever it might be called by the child) gets refined into subclasses; cows and horses, dogs and cats. For that child who pursues a doctorate in zoology, the classifications are extended far beyond those used by the average member of society. Subdivision of a class into a set of subclasses occurs when the observable attributes of the members of the class are found to form clusters and when the clustering is of interest to the observer. As technology increases observational capability, alternate classifications become feasible.

A substantially different type of learning occurs when reasoning over a model fails to produce the expected results. This kind of learning is sometimes called abduction, a process described by Peirce as a way of generating new ideas [10]. Abduction looks for a pattern in a phenomenon, but then it goes on to first suggest and then test a hypothesis or theory. It is in the area of hypothesis generation that people sometimes show the most brilliance over computer-based systems, sometimes formulating astonishingly insightful theories based on as few as one observation. It seems quite likely that this type of learning is based on reasoning by analogy and other mechanisms of exploiting the broader semantic networks of experience and mental models of the human. It is plausible that a semantic network of information over which abduction can occur in an artifactual system makes it more likely that useful patterns in complex information can be used for hypothesis-generation.

Abduction is very similar to reasoning by analogy, and in fact the hypothesis generation seems particularly associative or analogous in nature. Rather than generating a hypothesis from nothing (which poses a problem for a symbolic logic system), Hoffman [11] has theorized that a hypothesis is selected from an infinite set of possible theories. (Fortunately we do not have to generate all members of the set before we can select one to test.) The use of relevant knowledge and experience in the problem domain provides the human with "instinctive power" leading to hunches and leaps of intuition. It has been observed that the inductive step of abduction is not continually active, either in humans or animals, but is triggered by a surprising observation. For an observation to be surprising, it must in some way violate the expectations of the observer, which are derived from his models. It is the resulting search for a theory creating an expectation compatible with both the new and the old observations that generates the hypothesis.

No current theory of hypothesis generation seems wholly satisfying, but it is plausible that the process depends heavily upon the organization of experiences and beliefs (the models) that are represented in the memory of the cognitive system. We make the process more formal as follows. Suppose that a cognitive system believes a theory H, which, in situation S_n described in all ways believed to be relevant by the parameter set X_n, generates an expectation E of observations O_n. This is informally represented as:

$$\text{describes}(S_n, X_n) \text{ and } H(X_n) \Rightarrow E(O_n)$$

Note that a parameter of the set X_n might be a graph segement with depth greater than 1. We might suppose that the theory H was learned from previous situations classified by the cognitive system as being similar to S_n, i.e. $\{S_1, S_2, \ldots, S_{n-1}\}$. (Or more precisely, the cognitive system classified the previous situations $\{S_1, S_2, \ldots, S_{n-1}\}$as being similar and generating H and subsequently, upon being presented with the current situation S_n, identified it as belonging to the same class of situations.) Now suppose further that in the current situation S_n, the actual observations O_n' are surprising, i.e. $O_n' \neq O_n$ in some significant way.

The cognitive system might respond by looking for differences between the current situation S_n and the set of previous situations $\{S_1, S_2, \ldots, S_{n-1}\}$ used to generate H. One form this search might take is the consideration of an expanded parameter set X+Y where Y consists of one or more possible descriptors of situations $\{S_1, S_2, \ldots, S_{n-1}, S_n\}$ and where $Y_1=Y_2=Y_3=\ldots=Y_{n-1}\neq Y_n$. In other words, Y represents a significant difference between this situation and the previous situations. The search for Y would be informed by the cognitive system's ontology and deep understanding of what parameters not previously believed important are most likely to have relevance, i.e. are most likely to be involved in some causal chain leading to O_n'. Our mental models may play a crucial role in this search. Successful identification of a Y would lead to a revised theory H' such that

$$\text{describes}(S_i, X_i+Y_i) \text{ and } H'(X_i+Y_i) \Rightarrow E(O_i) \text{ for } i=1 \text{ to } n-1$$

and

$$\text{describes}(S_n, X_n+Y_n) \text{ and } H'(X_n+Y_n) \Rightarrow E(O_n')$$

Other methods of hypothesis generation can be envisioned that leverage the cognitive system's models. (For an example, see Thagard & Shelley [12].) These might include looking at observations that are similar in some way to the surprising observation, looking at causations (including reasons, purposes, and goals of other cognitive systems, e.g. enemies) that might produce these or similar observations, or looking at causations in other classes of situations that are not too distant by some measure of similarity (e.g., graph distance) from the current situation. The meanings of purpose, reason, goal, and situation

as used here are upper-level ontological concepts and an artificial cognitive system capable of this kind of learning will need to have representations and models for these kinds of abstract entities.

3. USING OWL REPOSITORIES AS A FRAMEWORK FOR LEARNING

OWL, with roots in the Description Logics community [13], stores both generalizations, e.g., class and property definitions, and instance data. The former is often referred to as the tbox or terminology. The latter is referred to as the abox. Instance data stored in an OWL repository is always semantically tagged, and therefore has an explicit semantic context. This characteristic, combined with a powerful query language such as SPARQL [14], provides a very powerful mechanism for data exploration. In other words, the ontological data is stored as a graph (in the mathematical sense—it may or may not be displayed as a visual graph). SPARQL is a graph query language. One can retrieve data of arbitrary semantic complexity. Drawing on an example from maintenance planning and logistics, one can construct queries of varying "reach" into the semantic network.

- Find all assets of a certain type
- Find all assets of a certain type that operate in a particular environment
- Find all assets of a certain type that operate in a particular environment and were serviced in the last year
- Find all assets of a certain type that operate in a particular environment, were serviced in the last year, and received a particular upgrade.

The graph structure of an ontology and the flexible reach of a graph query language provide valuable support for the various forms of learning discussed in the previous section. In the case of computing learned attribute values over the instances of a group, the group can be flexibly defined by a graph query that is grounded in class and property definitions in the application ontology and/or higher-level imported ontologies. In the case of tbox extension through induction of finer-grained subclasses, graph queries retrieve desired sets of data that can be analyzed for clustering or other patterns. In the abductive reasoning scenario, graph queries over the domain ontology allow exploration of modified hypotheses. For example, suppose that we have a group of engineers who use a maintenance and logistics planning application to maintain situational awareness of the status of a fleet. Suppose that an original hypothesis stated that the users of the planning system prefer to see a "red-yellow-green dashboard" display that visually summarizes the fleet performance. Suppose now that a new user, who happens to work for the service contract management department, changes the visualization from the dashboard format to a table that reports the raw availability statistics for that fleet (for example, to verify contract compliance). The system could see this anomaly and pose a modified hypothesis that users from the engineering department prefer to see the dashboard display and users from contract management department prefer to see a data table. Thus, the system proposes two new subclasses of user, distinguished by the department for which they work, and these users will now by default receive information displays tailored per those preferences. Future interactions with each subclass of user are then monitored to verify or reject the new hypothesis.

Note that in the case of abductive learning, the instance data necessary to test a hypothesis may or may not be present at the time of hypothesis generation. (For example, if an engineer from the above example happens to be color blind and changes his dashboard display to Red-Yellow-Purple but color blindness is not a property in the user model.) Pragmatic considerations will often dictate that observations not deemed relevant to the model at hand may not be made or may not be stored. Consequently, generation of an interesting hypothesis may dictate gathering of new observations that include the additional parameters of the extended hypothesis or it may cause observations to be stored which were formerly "forgotten."

4. ACUITY: A CASE STUDY IN LEARNING FROM OWL INSTANCE DATA

ACUITy provides a domain-independent shared semantics for mixed-initiative decision support in open-ended problem domains, both in terms of the user-interface concepts and in terms of general problem-solving concepts. It also provides an architecture that uses this metadata to take some direct action; for example, to implement changes in the composition of a user interface, or to execute some other behavior via scripts. In this application space, there is considerable opportunity to improve the usability and effectiveness of a system by enabling it to learn after it is deployed.

The ACUITy framework views decision support application development as a two-step "finish the design" process. First, the application developer extends the domain-independent ACUITy models of human-computer interaction and work domain information to the specific domain of interest, identifying the kinds of problems to be solved, the kinds of information relevant to those problems, and a basic set of information "vantages" (views) likely to be useful to a decision maker. Then the end-user is encouraged to finish the design by customizing information content, layout, and display parameters, e.g., graph type, colors, and ordering.

Because OWL does not support default values, ACUITy uses the rdf:seeAlso annotation property to associate instances of the class DefaultValue with the class to which the default is to apply (see Figure 1). When a new instance of a class is created, a special-purpose reasoner looks for a seeAlso property on the class with a value of type DefaultValue (or a subclass thereof). If present, the instance of DefaultValue is used to provide an initial value of the specified property for the new instance. What is important to the current discussion is the nature and use of a particular subclass of DefaultValue called LearnedDefaultValue. A LearnedDefaultValue also specifies the value to be used as the default if a learned value cannot be calculated.

In the current implementation of ACUITy, a LearnedDefaultValue has the following properties used by the special-purpose reasoner:

- valueCalculatedBy, with range Script, which identifies the specific algorithm to use to calculate the learned value
- dataSetDefinedBy, with range Script, which identifies the algorithm to determine the set of data from which to learn
- userPreference, which, if true, will collect similar instances of the class created for/by this user as the data set
- minimumSampleSize, which specifies the minimum amount of relevant instance data which must be found before a learned default will be calculated
- thresholdFrequency, which indicates, for applicable algorithms, the preponderance of usage necessary for a learned value to be used. For example, if the threshold is 75% and a given sample for most-frequent value is evenly split between "green" and "purple," the specified default value will be used and not the calculated value.

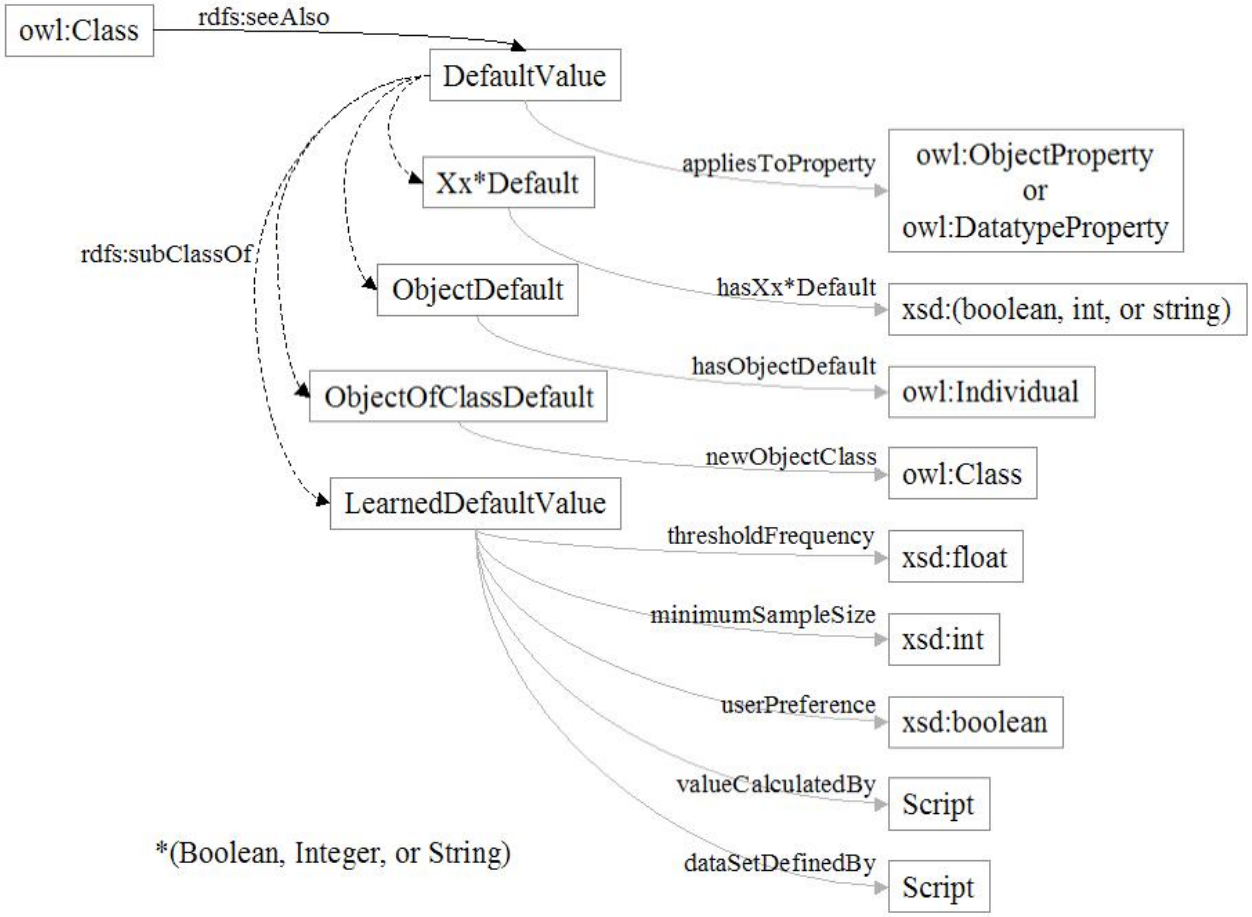

Figure 1: DefaultValue Properties and Ranges

Note that the Boolean "userPreference" is a special-purpose shorthand way of specifying the data set to be all instances of the subject class created by the current user. If a value of the property "dataSetDefinedBy" is given, a much more flexible definition of the data set over which learning can occur is possible. For example, the data set might be instances created by all users in the same role, by users in the same role with the same educational background, or any other expressible data selection criteria. While not currently implemented, there could be multiple selection criteria ordered so that if there is insufficient data at one level the next-level learning algorithm is used. In fact, the user might be allowed to specify the algorithm for data selection, possibly choosing to look at decision criteria and outcomes over a peer group to identify the "best performers," enabling the propagation of "best practices" to the user community.

Next steps in implementing ontology-based learning include supporting induction and abduction in open-ended problem domains. As we gain more experience with the ACUITy environment, we anticipate extending it with various learning modules. With our approach we can develop the ability to modify the standard behavior of an ACUITy-delivered application by recognizing changing patterns in the ontologies and instance data.

For example, we envision implementing a standard learning routine to emulate the behavior previously described in the graph versus table display example. This learning module would look for "large" style or content changes to the presentation such as changing a general presentation nature or including or removing information content. When such changes occur, the learning module would look for discriminating characteristics in other parameters that coincide with the change, such as a difference in the users' communities. It would then "propose" a new hypothesis, possibly by adding new subclasses, and then analyze subsequent user behavior to validate or reject the hypothesis. This validation/rejection phase would, across subsequent uses of the application, analyze the users' behavior to find contradictions to the hypothesis (e.g., users from the new user community using the old display). If sufficient contradictions were found, it would withdraw the hypothesis (possibly removing new subclasses). If usage patterns were consistent with the modified model, the new or modified hypothesis would grow in level of confidence.

As mentioned above, a sufficiently extensive shared semantics allows second-hand learning to occur as the learner is "taught" new concepts that extend existing models. We are working on an ACUITy Editor that will allow an existing application to be "taught" by developers and even end-users. The ACUITy Editor will make it easier and more intuitive to extend the application design by creating new concepts as natural extensions of the existing domain ontology. Once second-hand learning is enabled, we can extend learning analysis to monitor users' usage of and reactions to new model extensions.

5. CONCLUSIONS

The ACUITy environment is in its infancy, but we feel it is a promising start toward achieving the vision of its name: an Adaptive Work-Centered User Interface TechnologY. In this work we have combined:

- a tight coupling of the higher order semantics of domain-independent ontological models of human-computer interaction and problem solving with the implementation of an application's domain-specific models,
- user-based customizations/modifications to the application's model as intrinsic behavior in an ACUITy delivered application,
- the memory of these changes (built into the OWL modeling paradigm), and finally
- explicitly included learning as an extensible capability within our human-computer interface model.

The learning capability we have modeled positions the environment to truly deliver adaptive behavior in ACUITy-based applications. Our human-computer interface model specifically provides for extending it with new learning capabilities. Later this year ACUITy will be released to the Open Source community in the hopes that the technology will be found useful to a wide spectrum of practitioners and that the larger community will be willing to further extend and refine the many aspects of the ACUITy architecture, including its learning capabilities and potential.

6. ACKNOWLEDGEMENT

This work was funded by the Air Force Research Laboratory, Wright Patterson AFB, and by the General Electric Company under a dual-use program, contract number F33615-03-2-6300.

7. REFERENCES

[1] Aragones, Amy, Jeanette Bruno, Andrew Crapo, and Marc Garbiras, "An Ontology-Based Architecture for Adaptive, Work-Centered User Interface Technology," 2006. Available on-line at http://ip.com/pubView/IPCOM000134526D

[2] Aragones, Amy, Jeanette Bruno, Andrew Crapo, and Marc Garbiras, "Using ACUITy to Personalize Content in Semantic Web Applications," Proceedings of the Workshop on Personalization and the Semantic Web, European Semantic Web Conference 2006, *forthcoming*, 2006.

[3] Aragones, Amy, Jeanette Bruno, Andrew Crapo, and Marc Garbiras, "An Ontology-Based Architecture for Adaptive Work-Centered User Interface Technology," Jena Users Conference, 2006, *forthcoming*, 2006.

[4] Miller, Eric, and Jim Hendler, Web Ontology Language (OWL), 2006. Available on-line at http://www.w3.org/2004/OWL/

[5] Craik, Kenneth J., The Nature of Explanation, Cambridge University Press, Cambridge, UK, 1952.

[6] Johnson-Laird, Philip N., Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness, Harvard University Press, Cambridge, MA, 1983.

[7] Johnson-Laird, Philip N., The Computer and the Mind, Harvard University Press, Cambridge, MA, 1988.

[8] Johnson-Laird, Philip N., Human and Machine Thinking, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

[9] Pan, Barbara, and Jean B. Gleason, *Semiotic Development: Learning the Meanings of Words* in The Development of Language, 4th Edition, Gleason, Jean B. (editor), Allyn and Bacon, Boston, MA, 1997.

[10] Sowa, John F., Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole, Pacific Grove, CA, 2000.

[11] Hoffman, Michael, *Is there a "Logic" of Abduction?* Proceedings of the 6th Congress of the IASS-AIS, International Association for Semantic Studies, Guadalajara, Mexico, July 13-18, 1997.

[12] Thagard, P. & Shelley, C., *Abductive reasoning: Logic, visual thinking, and coherence.* In: M.-L. Dalla Chiara et al (eds), Logic and Scientific methods. Dordrecht: Kluwer, 1997, pp.413-427.

[13] Baader, Franz, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (editors), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[14] Prud'hommeaux and Andy Seaborne (editors), SPARQL Query Language for RDF, W3C Candidate Recommendation of 6 April 2006. Available on-line at http://www.w3.org/TR/rdf-sparql-query/