Approaches to Authoring of Rules for Intelligent Agents

(extended abstract, submission version 11/21/95)

Benjamin N. Grosof
IBM T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
(914) 784-7100
grosof@watson.ibm.com
WWW: http://www.research.ibm.com

Introduction

The problem we address is practical design of methods for users, especially relatively non-technical end users, to author rules that drive rule-based intelligent agents. We are concerned especially with networked applications of intelligent agents in the realm of information access, e.g., news, mail, Web pages, and the like. For these applications, an important value of intelligent agents is their ability to customize or personalize the behavior of an application.

A key challenge is how to enable an end-user to specify the behavior that is desired of that application agent. In other words, how shall the user instruct the agent? Generally, one can imagine a spectrum of approaches, ranging in complexity from full-blown programming languages, such as C++, at one end of the spectrum, to menus and direct manipulation at the other end. Scripting and macro languages, which recently have been receiving a great deal of attention and excitement in both industry and academe, occupy a place in this spectrum intermediate between full-blown programming languages and menus. This spectrum can be viewed as a trade-off frontier: lower skill and effort required vs. greater complexity and power of specified procedures/processing.

Rules and inferential reasoning occupy another place in this spectrum. We place them intermediate between scripting and menus: lower skill and effort required than scripting, but more than menus; greater complexity and power of processing than menus, but less than scripting.

A dream of many (we share it) is to warp out of the above trade-off frontier of explicit instruction: by relying on machine learning to perform powerful implicit instruction. Pragmatically, however, we believe it is important as an incremental first step to facilitate explicit instruction, for several reasons. First, explicit instruction is important as a target for learning by induction or advice. Experience in the field of knowledge-based systems teaches us that it is usually important to first build the performance element and understand how it would exploit knowledge supplied by a learning element, before actually hooking up a learning element. Second, explicit instruction is important to augment learning: to edit the results of learning, and to complement it when only some areas of knowledge can be learned. Third, explicit instruction is important to trust learning: e.g., to confirm or inspect suggestions made by a learning element.

Within explicit instruction, we believe that yes/no rules are an important first step. By yes/no rules, we mean rules that have a truth value of either true or false, as opposed to probabilistic-flavored rules (we would include fuzzy rules as in this category) that may have intermediate degrees of truth, e.g., 0.7. Yes/no rules have two advantages over probabilistic-flavored rules. First, their fundamentals are well understood, notably in terms of practical experience with knowledge-based systems over especially the last fifteen years. Second, yes/no rules enable the control of tasks with predictability. (Software today is mostly written using non-randomized control constructs.)

The RAISE Project and Experience at IBM

IBM has built a number of intelligent agents in different application areas, including information access, support of collaborative work, system management, workflow, adaptive user interfaces, customer service support, e-mail, desktop applications integration, and more. [1] gives an overview of these. The goal of our RAISE project at IBM Research is to create embeddable technology to support a variety of intelligent agent applications, including ultimately as many as possible of those within IBM. RAISE stands for Reusable Agent Intelligence Software Environment. A major part of RAISE is devoted to facilitating user authoring of rules. [2] gives an overview of RAISE. RAISE has been used to embed rule-based intelligent agents within two applications to date: the Globenet system for networked newsgroup-like information and customer service support using such newsgroups, and Lotus Notes collaborative work using newsgroup-like information. [3] describes the Globenet application of

RAISE. More generally, RAISE's design is informed by IBM's overall experiences in the area of intelligent agents, including rule authoring in particular.

Open Questions

In our RAISE design work, we grapple with a number of open questions, in the area of rule authoring, that are immediately and practically important in near-term commercial settings. Next, we list them.

Overall: What does it take to make creating rule bases accessible to end users that lack skills in AI knowledge engineering, and also lack deep understanding of rule-based inferencing techniques?

What kinds of graphical representations of rules, and graphical interfaces for creating one rule, are useful? Which are appropriate for different kinds of users? What is a helpful set of categories for users in this regard?

What kinds of textual representations of rules, and textual interfaces for creating one rule, are useful? How should these complement or substitute for graphical representations, e.g., perhaps as a summary in reviewing a whole set of rules?

For quite non-technical users, what technical concepts in inferencing is it possible, necessary, and/or helpful for them to grasp? E.g., chaining, forwards vs. backwards direction of inferencing, unification, quantification, logical connectives. What fundamental expressiveness, in the sense of logical knowledge representation, is possible, necessary, and/or helpful?

How application-specific vs. generic should the end users interface be for rule authoring?

How to integrate with scripting and macro languages?

What kinds of mechanisms are needed for testing and debugging rules, including to deal with conflicts between rules?

How to reduce the burden of creating, understanding, and reconciling ontologies?

Solutions and Lessons

For the questions above, we have some partial answers. These are based on our earlier-described experience, but have yet to be fully validated in widely deployed applications. Next, we list these approaches, observations, and ideas.

Visual Wiring: One approach to graphical interfaces is what we will call here: visual wiring. In this approach, a rule is created by "wiring" together antecedent conditions and consequent conditions (e.g., actions) that are pulled down from a menu of available conditions. This approach was taken, for example, by IBM's first intelligent agent product, IntelliAgent, an office desktop applications integrator. Wiring means indicating the logical connectives (and, or, implication) between the multiple conditions. E.g., left-to-right sequencing is used to indicate and`ing within the antecedent; branching within the antecedent is used to indicate or'ing. Some users like visual wiring. However, many do not; they find it too unstructured and too difficult to map to the application in which the agent is embedded.

Canned Rules, Forms, Schemas: We have found that many of the less technical users find it helpful to be able to select and parametrize from a collection of "canned" (i.e., predefined, schema) rules and canned rule sets. Forms (in the sense of forms that one fills out) are then an appropriate mode of graphical interface for authoring rules. This approach is being taken in IBM`s work on the Alter Ego intelligent agent for e-mail and personal messaging, for example. The first version of Globenet also takes this approach, but makes various conditions optional to include. This obviates the need for the user to specify logical connectives, but enables a rich number of combinations of conditions: in effect, a large number of rule schemas.

Textual Summarization: Many users find a textual summary of each rule to be a helpful accompaniment to graphical interfaces (e.g., forms). It is more concise, and is closer to natural language; these help especially when reviewing an entire set of rules.

Rule Set (not just individual rule) as a Primary Concept: A set of rules is employed together. This view is important and helpful for users to understand, especially when rules chain or when rules conflict. It is useful to manifest the concept of a rule set in a graphical interface, e.g., by a metaphor of a "book" of rules (e.g., in IntelliAgent or Alter Ego), or in a textual interface, e.g., with a summary title such as "Handling E-Mail While on Vacation".

Keep the Interface Close to the Application or Tool: Users, especially when non-technical, find it helpful when the authoring interface is close to the overall application in terms of concepts and look- and-feel. This principle applies not just at the level of whole rules, but also at the level of individual conditions. E.g., when specifying a free-text condition such as a keyword combination, it is helpful to bring up the interface of an underlying free-text search tool. IntelliAgent takes this approach, for example. The RAISE architecture has been designed to enable this pattern of interfacing.

Non-Horn Connective Structure: in several applications, including IntelliAgent and Globenet, a helpful form of rule is more complex than Horn form. In this form, there may be at least one level of nesting of and's and or's in the antecedent; in addition, there may be and'ing within the consequent. To date, we have found users not to have much trouble grasping this logical complexity; rather, they find its conciseness to be an aid.

Chaining: We have found nontechnical users indeed to be able to exploit at least a small degree of chaining, e.g., to chain through "Importance" (high or low) as an intermediate condition in the Alter Ego agent for e-mail handling.

Declarativeness, Merging, Porting: In RAISE, we have developed an approach to representing rules that is highly declarative (in the AI Knowledge Representation sense of declarative vs. procedural). Rules that are logically monotonic are distinguished from rules that are logically non-monotonic. Moreover, rules purely about belief are distinguished from procedural attachments, which are called "reflexes". A reflex associates a logical atom (typically having free variables) with an attached procedure call. One kind of reflex is an "effector": the attached procedure is invoked to perform some action, e.g., to forward a piece of mail. A second kind of reflex is a "sensor": the attached procedure is invoked to perform a special test (e.g., an analysis or query). A highly declarative underlying rule representation has several advantages. One is that it makes it simpler for the user to anticipate and understand the results of adding or deleting rules to a set of rules. Another is that it makes it simpler to separate the meaning of a set of rules from the particular (e.g., local to a PC) set of available procedures: this aids portability and platform-independence.

Sharing, Advice, Communication: RAISE aims to capitalize on the idea that the easiest way to obtain rules is to "copy" them from some other user or user's agent. That is, we believe that reusability of knowledge, i.e.., rules, is at least as important as reusability of code. RAISE supports emerging standards in the area of knowledge interchange, e.g., the ARPA Knowledge Sharing Effort. But more than that, RAISE facilitates some of the deeper aspects of knowledge sharing, via its high degree of declarativeness. For example, it facilitates the assimilation of advice (rules received from another user or agent) by simplifying the semantics of merging two rule sets into one new rule set.

Acknowledgements

Thanks to Manny Aparicio, Joshua S. Auerbach, Stephen Brady, Hoi Y. Chan, Tad Davis, Davis Foulger, Colin G. Harrison, Terry Heath, Ed Kunzinger, David W. Levine, Henry Lieberman, Frank McKiel, Colin J. Parris, Ted Selker, and Rick Spagna for useful discussions.

References

[1] "IBM Applications of Intelligent Agents". Manny Aparicio *et al.* http://activist.gpl.ibm.com:81. Soon available as an IBM Research Report: http://www.research.ibm.com, click on Cyberjournal.

[2] "Reusable Architecture for Embedding Rule-based Intelligence in Information Agents". Benjamin N. Grosof, David W. Levine, Hoi Y. Chan, Colin J. Parris, Joshua A. Auerbach. In Proceedings of the ACM CIKM-95 Workshop on Intelligent Information Agents. Held in conjunction with the Conference on Information and Knowledge Management, Baltimore, MD, Dec. 1–2, 1995. Workshop Editors: Tim Finin and James Mayfield. http://www.cs.umbc.edu/iia/ . Soon available as an IBM Research Report RC 20305: http://www.research.ibm.com, click on Cyberjournal.

[3] "Globenet and RAISE: Intelligent Agents for Networked Newsgroups and Customer Service Support". Benjamin N. Grosof and Davis A. Foulger. In Proceedings of the 1995 AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval, held Cambridge, MA, Nov. 10˚12, 1995. Editor: Robin Burke. http://www.aaai.org . Available as IBM Research Report RC 20226: http://www.research.ibm.com, click on Cyberjournal.