Expert Systems as Overlapping Logical Theories

David Roach, Hal Berghel

University of Arkansas
National Center for Toxicological Research

Abstract

We discuss the model-theoretic characterization of the components of logic-based expert systems which employ pure meta-level inference regimes. The roles of the basic components of such systems in the underlying first order theories are specified. This includes the domain specific rules, meta-level inference engine, and intrinsic interpreter. Fragments of an actual expert system are used to motivate and illustrate the analysis.

Introduction

Although expert systems have been widely implemented using logic-based approaches, not enough has been done to relate the systems to the logical theories which they instance. A deeper characterization of such systems is needed in terms of their underlying first order theories. We undertake to provide such a characterization by examining fragments of a pure meta-level expert system and describing its components in model-theoretic terms. Interesting features of the underlying formal structure of such expert systems are made explicit by this type of analysis - for instance, the existence of multiple overlapping theories.

Pure meta-level inference systems are the focus of the analysis and are described in [5]. Meta-level systems are those in which there is a clear separation between control knowledge (meta-knowledge) and domain knowledge (object-knowledge). The utility of this separation and the implementation of it in logic-based systems has been explored in many places [2,4,9]. Of particular interest are the pure meta-level inference systems in which the *locus of action* [5,12] is primarily at the meta-level. Harmelen gives the following analysis of such systems.

> The behavior of the object-level is fully specified at the meta-level. Using this description of the object-level, the meta-level can completely simulate the object-level inference process. This means that there is no longer a need for an explicit object-level interpreter. As a result, the object-level interpreter is no longer present in the system, and its behavior is completely simulated by the execution of its specification at the meta-level [5, p. 23].

This analysis is procedurally oriented - as evidenced by the discussion of the meta and object-level *interpreters*. A different and deeper characterization of pure meta-level architectures is possible with a more declaratively oriented analysis. We seek to provide such an analysis in this paper.

An informal description of a pure meta-level expert system that was developed by the authors can be found in [10,11]. Our declarative (model-theoretic) analysis is based on it. The system is called RAP (for Relocation Allowance Planner). It's purpose is to determine travel allowances for Government employees who are transferred to a new post of duty. The three components of the system that are relevant to the current discussion are: the problem domain rules, the inference engine clauses, and the built-in inferencing mechanism. Since the system is written in Prolog the three components should be viewed in that context. That is, the problem domain rules are English-like Prolog constructs, the inference engine is a set of Prolog clauses, and the built-in inferencing mechanism is the resolution/refutation facility intrinsic to Prolog.

First Order Theories

First order theories are discussed in [3,7,8]. The components of such theories are a first order *language*, a set of *axioms*, and a set of *inference rules*. Since RAP is written in Prolog, we adopt as our language the Horn clause subset of first order logic. See [3,7]. The formal specification of the well-formed formulas of first order logic is as follows:

A first order logic term is defined as follows:
1) A variable is a term.
2) An object constant is a term.
3) If Φ is an n-ary function constant and $\tau_1,...,\tau_n$ are terms, then $\Phi(\tau_1,...,\tau_n)$ is a term (called a *functional expression* in [3]).

A first order logic wff is defined as follows:
1) If Ψ is an n-ary relation constant and $\tau_1,...,\tau_n$ are terms, then $\Psi(\tau_1,...,\tau_n)$ is a wff (called an *atom* or *atomic sentence*).
2) If P and Q are wffs, then so are $(\sim P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$ or equivalently $(Q \leftarrow P)$, and $(P \leftrightarrow Q)$.
3) If P is a wff and x is an apparent variable, then $(\forall x P)$ and $(\exists x P)$ are wffs.

We wish however to restrict our language to Horn clauses, i.e., clauses with at most one positive literal. Therefore, the wffs of our language are confined to those first order wffs which can be transformed into an expression (or a set of such expressions) of the form

$P_1 \vee ... \vee P_m \vee \neg Q_1 \vee ... \vee \neg Q_n$, where $m \leq 1, n \geq 0$

by some complete and consistent set of rules of inference [6]. We assume the usual model-theoretic semantics for first order logic expressions.

The axioms and rules of inference for our theories will be components of the expert system being analyzed. We include the axioms necessary to have theories with equality.

Expert Systems

As stated in the introduction, there are three distinct but interrelated components of RAP that must be accounted for in any analysis (declarative or procedural): a rule base in which the domain specific information is encoded (we refer to these as the Level 1 (L1) clauses), a set of programmer written inference engine clauses for controlling the chaining process and capturing the trace (the L2 clauses), and an inference engine native to the logic programming system being used (the L3 clauses). We need to provide a description of each of these components.

The L1 clauses encode the expert knowledge. In RAP, this knowledge is represented by a set of rules whose syntax is described in terms of a hierarchy of expressions as follows:

Let the vocabulary of an arithmetic expression be
1. a finite set of real-valued variables, $V=\{v_1,v_2,...,v_k\}$,
2. the arithmetic operators '*', '/', '+', '-', and
3. the punctuation symbols '(' and ')'.
We define arithmetic expressions inductively as follows:
1. Real-valued variables and real numbers are arithmetic expressions.
2. If A and B are arithmetic expressions, then so are (A * B), (A / B), (A + B), and (A - B).

If we let the vocabulary of a relational expression be
1. a finite set of real-valued variables, $V=\{v_1,v_2,...,v_k\}$, and
2. the set of relational operators, $O=\{=,<,>,\leq,\geq\}$,
then a relational expression is an expression of the form v_i o v_j or v_i o μ, where $1\leq i,j\leq k$, $v_i,v_j \in V$, o $\in$ O, and μ is a real number.

The vocabulary of a sentence evaluation consists of
1. a finite set of sentence variables, $S=\{s_1,s_2,...,s_k\}$,
2. the identity operator '=', and
3. the set of sentence values, E={yes,no}.
The resulting sentence evaluations are expressions of the form s_i = e, $1\leq i\leq k$, where e $\in$ E.

Let the vocabulary of a logical expression be
1. the vocabularies of relational expressions and sentence evaluations,
2. the set of logical operators, O={and,or}, and
3. the punctuation symbols '(' and ')'.
A logical expression is defined inductively as follows
1. Relational expressions and sentence evaluations are logical expressions.
2. If A and B are logical expressions, then so are (A and B) and (A or B).

We can now define assignment and conditional rules. The vocabulary of an assignment rule is composed of
1. the vocabulary of arithmetic expressions,
2. the assignment operator '==',
3. a finite set of sentence variables, $S=\{s_1,s_2,...,s_k\}$, and
4. the set of sentence values, E={yes,no}.
There are two forms of the **assignment rule.**
1. If A is a real-valued variable and B is an arithmetic expression, then A == B is an assignment rule.
2. Any expression of the form s_i == e, $1\leq i\leq k$, where e $\in$ E is an assignment rule.

By letting the vocabulary of a conditional rule be
1. the vocabularies of logical expressions and assignment rules plus
2. the conditional operator 'if then',
we can define a **conditional rule** as follows:
If A is a logical expression and B is an assignment rule, then if A then B is a conditional rule.

The following rules are instances of the rule schemata. The first is a conditional rule and the second is a typical assignment rule.

```
if
   (($you$ will $sign a 12 month service
   agreement$ = yes
or
   ($you$ are $a Department of Defense overseas
   dependents school system teacher as determined
   under Chapter 25 of title 20 of the United
   States Code$ = yes and
   $you$ will $sign a service agreement for 1
   school year$ = yes))
or
   ($you$ are $separated for reasons beyond your
   control$ = yes and
   $you$ will $transfer with approval of the
   concerned agency$ = yes))
then
   $you$ meet $the service agreement requirements
   in 2-1.5a(1)(a-b)$ == yes.

$your subsistence/transportation allowance$ ==
   ((($your subsistence allowance per day$ +
   $the subsistence allowance for your immediate
   family per day$) *
   $the number of days required to complete the
   move$) +
   $your transportation allowance for
   relocating$).
```

The specification of the syntax of the L1 clauses makes explicit the fact that these clauses contain variables which range over a particular universe of discourse. Specifically, the variables in the L1 clauses range over the objects (events, costs, times, places, people, statements, etc.) that are relevant to the particular problem domain being addressed.

The L2 clauses make up the programmer defined expert system inference engine. Those clauses in RAP's inference engine that are necessary to the current analysis are discussed. The complete engine is described in [11]. Interpreted procedurally, the L2 clauses direct the chaining process and construct traces of the paths followed. (The sample system is backward-chaining; however, a forward-chaining engine has the same model-theoretic interpretation.) Informally, the inference engine clauses are of the form

```
pursue(Goal,Why,How) :- subgoals
```

where Goal is either a complex goal that is to be parsed or a simple goal for which a matching rule is sought, Why is a list that contains the path from the top-level goal to the current goal (the 'why-trace'), and How is a tree structure of all the paths from the top-level goal to all lower-level goals that have been satisfied (the 'how-trace').

The following L2 clause parses the complex goals which result from 'chaining in' an L1 clause with conjoined antecedent conditions.

```
pursue((Goal1 and Goal2),Why,Reason) :- !,
   pursue(Goal1,Why,Reason1),!,
   ifthenelse(positive(Reason1),
      (!,pursue(Goal2,Why,Reason2),
         ifthenelse(positive(Reason2),
            Reason = (Reason1 and Reason2),
            Reason = Reason2)),
      Reason = Reason1).
```

The important point is what the variables Goal1 and Goal2 in the head of this clause range over. Obviously, it is not over the same set of objects as was the case with the L1 variables. Instead, Goal1 and Goal2 range over portions of L1 clauses. They each range over one of some number of conjoined antecedent conditions of an IF..THEN rule. Thus, they are metavariables with respect to the variables of the L1 clauses. The next L2 clause further illustrates the meta-level characteristic of the L2 clauses.

```
pursue(Goal, Why, Goal eq Value is TruthValue
because Reason) :-
   recorded(rule(if Condition then Goal =
      Expression),_),
   [!(pursue(Condition,[$To determine a value for$
      + Goal|Why], Reason1);(!,fail)),
   truth_value(Reason1,TruthValue)!],
   TruthValue == true,!,
   pursue(Expression,[$To determine a value for$ +
      Goal|Why],Reason2),!,
   evaluate(Expression,Value,$To determine a value
      for$ + Goal),!,
   Reason = ((Goal eq Expression is true because
      Reason1) and Reason2),!,
   recordz(wasderived(Goal,Value),_).
```

In the first subgoal of the clause, the rule base is searched for an IF..THEN rule whose consequent has a left operand that will unify with the current goal. The variables Condition, Goal, and Expression all range over subcomponents of the L1 clause that is retrieved. As with the previous L2 clause, the *structure* of the target object level clause is made explicit. As a result, the search for the L1 clause can be thought of as 'content-directed' [1].

The L3 clauses are the last expert system component that requires analysis. However, there is no need to do so here. The L3 clauses make up the inference engine intrinsic to the logic programming system being used - in this case, the Prolog engine. As such, they have been carefully and fully analyzed both declaratively and procedurally. See especially [5]. However, we should note their role in RAP. That role is analogous to the role played by the L2 clauses with respect to the L1 clauses. Obviously, the built-in (intrinsic) engine has as its *object* the L2 clauses. As a result, the variables in the L3 clauses range over L2 clauses and subcomponents. If we think of the L2 variables as metavariables, then the L3 variables should be thought of as metametavariables.

Overlapping Theories

Obviously, the variables in the three sets of clauses do not all range over the same universe of discourse. The variables in the L1 clauses range over the entities that are part of a particular problem domain, the L2 variables range over L1 clauses and subcomponents, and the L3 variables range over L2 clauses and subcomponents. How are we to interpret these facts within the standard model-theoretic framework?

We begin by observing that a theory (T1) can be constructed of the L1 and L2 clauses and the UD that the L1 variables range over (UD1). The universe of discourse for T1 consists of the entities listed earlier that are a part of the problem domain being addressed by the expert system. In this case, the L1 clauses form the axioms of T1. Since the L2 clauses are defined over the L1 clauses, the L2 clauses can be thought of as inference rules for T1. But we also have the components for a second theory (T2). The L1 clauses and subcomponents can be thought of as elements of a second UD (UD2). The L2 clauses become the axioms for T2 since the L2 variables range over UD2. Since the variables of the L3 clauses range over L2 clauses and subcomponents, the L3 clauses can be interpreted as the inference rules of T2.

Neither T1 nor T2 alone is sufficient for a model-theoretic interpretation of an expert system, since each theory in isolation would leave components unaccounted for. In fact, no single theory is adequate to the task. Consequently, we must assume that a model-theoretic analysis of expert systems requires at least a dual theory analysis. But such an analysis must go further and make explicit the relationship between the component theories.

We see immediately that T1 and T2 'overlap' in that they share a component. The L2 clauses are present in each theory. However, the L2 clauses are interpreted differently in each theory. In T1 the L2 clauses function as inference rules since the clauses' variables range over L1 clauses and subcomponents. In T2 they function as axioms since the variables range over the theories' UD. Thus, the L2 clauses serve a dual role.

As a result, the L2 clauses relate two distinct parallel deductive processes. When a consultation with an expert system is in progress, two 'proofs' are being carried out simultaneously - one within T1 and another within T2. Within T1 the L2 clauses direct inferences among the L1 clauses (and resulting theorems), and within T2 the L3 clauses direct inferences among the L2 clauses (and theorems).

There are several benefits to a purely declarative model-theoretic characterization of the components of an expert system. First, it satisfies the need to make the declarative interpretation of these systems explicit. Second, a declarative analysis serves to bridge the gap between our relatively high-level conceptualization of expert systems and their more fundamental structure. Third, it provides an informative explanation of Harmelen's observation that the object-level interpreter is simulated at the meta-level. It functions as a set of inference rules at the object-level (and thus exists in the object-theory), but it functions as a set of axioms at the meta-level (so that it is an integral component of the meta-theory). Finally, a declarative analysis clearly shows the relevance of discussions of completeness and

strictness (see [5]). Expert systems have as fundamental components inference rules, which are the true bearers of these sorts of meta-theoretical properties.

Conclusion

We have attempted a declaratively oriented analysis of pure meta-level expert systems in terms of the more fundamental components of logical theories. This analysis indicates that an expert system can be usefully characterized as a set of overlapping logical theories. It is hoped that the characterization is informative with respect to the basic nature and functioning of such systems.

References

[1] Davis, R. and B. Buchanan. "Meta-Level Knowledge: Overview and Applications." Readings in Knowledge Representation. Ed. Ronald J. Brachman and Hector J. Levesque. Los Altos, California: Morgan Kaufmann, 1985. 389-397.

[2] Gallaire, M. and C. Lasserre. "Meta-level control for logic programs." Logic Programming. Eds. K. Clark and S. Tarnlund. Academic Press, 1982. 173-188.

[3] Genesereth, M. and N. Nilsson. Logical Foundations of Artificial Intelligence. Los Altos, California: Morgan Kaufmann, 1987.

[4] Genesereth, M. and D. Smith. "An Overview of Meta-Level Architecture." Proceedings of AAAI-83. American Association for Artificial Intelligence, 1983. 119-124.

[5] Harmelen, F. "A Classification of Meta-level Architectures." Logic-Based Knowledge Representation. Eds. P. Jackson, H. Reichgelt, and F. harmelen. Cambridge: MIT Press, 1989. 13-35.

[6] Kowalski, R. Logic for Problem Solving. New York: North-Holland, 1979.

[7] Lloyd, J. Foundations of Logic Programming. Berlin: Springer-Verlag, 1984.

[8] Mendelson, E. Introduction to Mathematical Logic. Princeton, NJ: Van Nostrand, 1964.

[9] Pereira, L. "Logic Control with Logic." Proceedings of the First International Logic Programming Conference. Marseilles, 1982. 9-18.

[10] Roach, D., et al. "RAP: An Expert System for Relocation Allowance Planning." Proceedings of the 1989 ACM South Central Regional Conference. Tulsa, 1989. 106-111.

[11] Roach, D. and H. Berghel. "The Physiology of a Prolog Expert System Inference Engine." Proceedings of the 1990 ACM Symposium on Personal and Small Computers. Washington, D.C., 1990. [in press].

[12] Welham, B. "Declaratively Programmable Interpreters and Meta-level Inferences." Meta-level Architectures and Reflection. Eds. P. Maes and D. Nardi. North Holland Publishers, 1987.