Integrating Logic Programs and Schemata*

Bradley P. Allen
J.M. Wright
Intelligent Systems Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania U.S.A. 15213

ABSTRACT

(orn clause/Schema Representation Language) is the result of an athor to combine the- tools of logic programniinp. and schema based knowledge represention into a single hybrid system. knowledge,compressed in schemala can be accessed during; the execution of logic programs,and the retrieval of the values of a SLOT in a schema can involve- the execution of logic programs that attempt. to declare the- values prior to presenting- to inheritence, should the slot be empty. ISRI. supports the implementation of programs that take advantage of the best controles. of the logical and objec oriented approaches to knowledge represenlation,

I INTRODUCTION

The- dictionary between declarative knowledge- representations which use sentences of logic- as their basic units and those which use schemata (a.k.a. frames, units, scripts) has been a problem for Artificial Intelligence since the early 1970's. Aside from the different philosophical commitments implicit in either kind of approach, using one or the other has certain pragmatic Consequences.

knowledge bases constructed using schemata are generally easy for people to comprehence| because of their object-oriented, taxomonic structure. Unfortunately. most schema-based representations have no general inference mechanisms other than inheritance. Furthermore, Ihe semantics of such representations is usually unclear, and frequently their only rigorous description Is the implementation code itself.

On the other hand, k nowledge bases consisting of sentences of logic have a clearly defined semantics and almost always have a general inference mechanism (i.e., a theorem proved) with well understood properties, However, the lack of structure in logic knowledge bases makes it hard be for people to grasp what they contain when they are of sufficient size tp be usefull in practical applications, Such representations also lack the defalt, reasoning, ability which is available in schema-based representations through the use of inheritance.

The problem that presents itself is how to build a knowledge representation which has the advantages of both approaches and avoids their disadvantages, |2] and Rich have proposed systems in which knowledge is stored ill two parallel representations, one logical and the other schemah.ised. An explicit interface betweem the two representations allows changes to knowledge in one to be incrementally translated irto the other. While this approach has the merit, of allowing the user to exploit the strengths of either kind of representation as he chooses, it has the disadvantage that the strengths of one representation are not brought to bear on the weaknesses of the other.

Another proposal is to implement inheritance-like inference in a logical representation. This approach stems from the perception that schemata are simply a syntactic variant of logic [6]. A system of this sort has been described by Frisch and Allen [5]. This approach provides an answer to the imprecise semantics of ad-hoc inheritance mechanisms, but leaves unaddressed the problem of the usability of large but unstructured logical knowledge bases.

This paper describes a system called IISRL, which attempts to unify the schema based and logical approaches through the integration of an interpreter for logic programs with a schema-based knowledge representation. We first describe the component systems of IISRL and show how they are modified and then combined to create the unified system. Then we discuss the merits of IISRL as a logic programming language and as a schema-based knowledge representation, indicating where IISRL addresses the problems raised above. We close by indicating directions for future research.

II THE COMPONENT SYSTEMS

A. The SRL knowledge representation

SRL (Schema Representation Language) [13] is a schema-based knowledge representation written in Franz Lisp. SRL is used for factory and organizational modeling in projects in the Intelligent Systems Laboratory of the Robotics Institute. A schema in SRL is composed of a name (printed in **bold** font) which denotes the object represented by the schema and a set of slots (printed in *slanted* font). Slots are used to denote the attributes of the schema as well as relations along which inheritance may take place. Each slot may have any number of values, and can have a schema, called a meta-schema, associated with it. The slots of the meta-schema (printed in *italic* font) are used to store meta-information about the slot. An example of an SRL schema is the **chameleon** schema shown in Figure 1.

{{ chameleon
is-a: **lisard**
habitat: **tropical**
color: **green**
description: **chameleon-color**
location: **}}**

Figure 1. The schema chameleon

" This paper describes work done in the Intelligent Syslems Laboratory of the Robotics Institute at Carnegie Mellon Univcraily. This work was.supported by the Air Force Office of Scientific Research under contract F19620-82-k0017.

Values lor l.he slots in a schema may be obtained from slots in other schema!.a through inheritances relations. For example, the value tropical can be inherited in the *habitat* slot, of the schema Claude (shown in Figure *2)* through the inheritance rr-lation *instance* from the *habitat* slot, in l.he chameleon schema.

{{ Claude
instance: chameleon
habitat:
color:
locution: rock-59}}

Figure 2. The schema Claude, an *instance* of a chameleon

Inheritance relations in SRL are themselves represented by schemata. This allows the declarative specification of idiosyncratic inheiitance relations by the user [4],

B The HCPRVR Horn clause theorem prover

HCPRVR is a logic program interpreter written in Lisp. Using HCPUVR is much like using the logic programming language PROLOG [II|, with the exception that the syntax is somewhat simpler. A description of the operation ol" HCPVR and of the syntax of logic programs in HCPUVR is given in [3].

III INTEGRATING SRL AND HCPRVR

A. Using knowledge in schemata in the interpretation of logic programs

The first step in integrating SRL and HCPRVR into HSRL is to modify the function PROVE in HCPRVR, which controls the problem reduction process used to interpret logic programs. The definition of PROVE is altered so that if the formula for which an instantiation is sought starts with the name of a schema followed by the name of a slot in the schema, then the axioms matched against the formula arc obtained by interpreting the slot in the following way: A slot *sl* with *n* values in a schema sch can be considered as a set of in fix atomic formulae of the form

(sch *sch value1),* (ach *sl value2), ...,*(sch *sl* $value_n$*).*

Under this interpretation, retrieving the values of *si* in sch is the same as finding all instantiations of the atomic formula

(schsli),

where x is a free variable. For example, the formula

(Claude habitat *x)*

is matched against the formula

(Claude *habitat* tropical),

since the *habitat* of Claude is tropical by inheritance from the *hnbitnt* slot of chameleon.

B. Using logic programs to retrieve slot values in a schema

The second step in the integration is to modify the slot value access function VALIUM in SRL. In SRL, the values or the slot *si* in schema »ch are retrieved with a function call of the form (VAMJKC sch *si).* VAbUKC uses inheritance if values arc not directly stored in *si* However, in IISRL we want to be able to lirsl use deduction to liud the values of a slot whenever applicable axioms are in the knowledge base, and then use inheritance to find reasonable default values when deduction fails. Furthermore, we want, to place the information for deducing the values of slots together with the slots in the schemata.

We achieve this by changing the definition of VAIJJIvC so that it attempts to deduce slot values before inheriting them. The knowledge that VAI.UKC uses to deduce values is expressed in the form of *description schemata.* The role of description schemata in a IISRL knowledge base is roughly analogous to that of definite and indefinite descriptions in a theory of first-order logic (10). Descriptions have been used in several knowledge representations as a method for the declarative specification of deductive procedures, most notably in D-SCRIPT (8) and Omega [I]. An example of a description in 1ISKL is l.he schema chameleon-color, shown in Figure .'J. chameleon-color oxpresses the axiom that the color of a chameleon is that oT the object on which the chameleon happens to be located.

{{ chameleon-color
instance: description
schema: ch
filler: clr
variables: obj
clauses: (ch location obj) (obj color clr)}}

Figure 3. A description of the *color* slot of chameleon

To illustrate the new procedure with which IISRL obtains slot values, we show how the call (VALULG Claude *color)* is evaluated.

First, the *color* slot of Claude is examined for a value. When none is found, the *description* slot in the meta-schema of the *color* slot is examined for a value. No value is found and no meta-schema exists for the *description* slot but the value chameleon-color can be inherited from the *description* slot of the meta-schema of the *color* slot of chameleon, chameleon-color is then interpreted by IIOPRVI? as the axiom

(Claude *color clr)* < (Claude *location obj)(obj color clr).*

The antecedent of tins axiom is constructed from chameleon-color by substituting the name Claude for each occurrence of the value of the *schema* slot in the values of the *chaises* slot. The consequent is the atomic formula whose free variable is the value of the *Idler* slot and which represents a query for the values of *color* in Claude. The axiom is then used to find all instantiations of (Claude *color* clr). VALULC, then returns the value obtained, which is the value in the *color* slot of the schema rock-59. Hence Claude as represented in the knowledge base will always have the same color as the object on which he happens to be located. If the retrieval of the *color* of rock-59 had failed, the value green would have been inherited from chameleon.

IV TWO VIEWS OF HSRL

A. HSRL as a logic programming language

IISRL permits the use of HCPRVR in the way described in [3], so the user is allowed to write standard logic programs. Pure HCPRVR axioms can be also be combined with axioms referring to schemata in a knowledge base and with axioms in the form of

descriptions associated with slots in the schemata. Embedding logic, programs in descriptions provides a natural way to index axioms in the knowledge base that people as well as programs can use. Because inheritance is used only when an attempt to deduce the values of a slot using descriptions has failed, HSRL performs default reasoning in exactly those cases where it is needed.

B. HSRL as a schema-based knowledge representation

An HSRL knowledge base that does not contain descriptions behaves in exactly the same way as an SRL knowledge base. However, including descriptions in a knowledge base allows the user to exploit HCPRVR so that modus ponens inference, as well as inheritance, can be used in answering queries in an incomplete knowledge base. One shortcoming of HSRL is that it sidesteps the issue of clarity in the semantics of inheritance by relying on the inheritance algorithm of SRL. However, the idea of using logic programs in descriptions to deduce the values of a slot can be extended to include the use of descriptions to deduce the default values of a slot. We will explore this technique in a future paper.

V SUMMARY AND FUTURE RESEARCH

HSRL is a hybrid logic/schemata knowledge representation which unifies the methodologies of schema-based knowledge representation and logic programming. It does so in a way that uses (IK strengths of one representation to compensate for the weaknesses of the other. Issues that we plan to deal with in future extensions to USUI, are the dynamic ordering of clauses in descriptions for increased effiencicy in retrieval [12], the use of descriptions to represent data dependencies in a knowledge base, extending descriptions with arbitrary clauses [7], and specifying default, reasoning with descriptions [5]. HSRL is implemented in Franz Lisp running under Berkeley UNIX on a Digital Equipment Corporation VAX 11/780.

ACKNOWLEDGEMENTS

The authors would like to thank Mark Fox, Hill Scherlis, laime Carbonell, Raul Haley, and Jim Driscoll for their helpful comments on earlier draffs of this paper. Special thanks go to our colleagues in the Intelligent Systems Laboratory of the; Robotics Institute, whose collective blood and brains went, into the superb research environment that made this work possible.

REFERENCES

[I] Attardi, C, and Simi, M. Semantics of inheritance and attributions in the description system Omega. Al memo 642, Artificial Intelligence Laboratory, MIT, 1981.

[2] Brachrnan, R..L, arid Lcvcsque, II..1. Competence in knowledge representation. Rroc. AAAL82, 1982, pp.189 192.

{'.\\ Chester, D. HCPRVR.: an interpreter for logic programs. proc. AAAI-80, 1980, pp. 93 95.

[4] Kox, M.S. On inheritance in knowledge representation. Proc.) IJCAI-79, 1979, pp. 282 284.

[5] Frisch, A.M., and Allen, J.F Knowledge retrieval as limited inference. Proc. 6th Conference on Automated Deduction, 1982, pp. 274 291.

[6] Hayes, P.J. The Logic of Frames. In *Frame Conceptions and Text Understanding,* Metzing, D., ed., de Cruytcr, 1979, pp. A6 61.

[7] Kowalski, R.A. *Logic for Problem Solving.* North-Holland, 1979.

[8] Moore, R.C. D-SCRIPT: a computational theory oF descriptions. Rroc IJCAI-73, 1973, pp. 223 229.

[9] Rich, C. Knowledge representation languages and predicate calculus: how to have your cake and eat it too. Rroc. AAAI-82, 1982, pp. 193 196.

[10] Scott, D. Existence and description in formal'logic. In *Bertrand Russell: Philosopher of the Century,* Schocman, R., ed., Allen and Unwin, 1967, ch. 16, pp. 181 200.

[II] Warren, D.H. Prolog on the DECsystom-10. In *Expert Systems in the Micro-Mice electronic Age ,* Michic, D., ed., Edinburgh University Tress, 1979.

[12] Warren, D.H. Efficient processing of interactive relational database queries expressed in logic. Tech. Rept. 156, Department of Artificial Intelligence, University of Edinburgh, 1981.

[13] Wright, .I.M., and Fox, M.S. SRL/I.5 User Manual. Tech. Rept., Robotics Institute, Carnegie-Mellon University, 1983.