Results of an Experiment in Domain Knowledge Base Construction: A Comparison of the Classic and Algernon Knowledge Representation Systems

Raman Rajagopalan
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
raman@cs.utexas.edu

Abstract

Jon Doyle and Ramesh Patil have recently argued that classification languages that obtain polynomial worst case response times by eliminating inefficient constructs are in fact too restrictive to represent many forms of knowledge. They suggest that some of these restrictions could be avoided by incorporating and constraining the use of problematic constructs rather than eliminating them. We extend the observations of Doyle and Patil by reporting our experience in using the CLASSIC and Algernon knowledge representation languages to model the domain of magnetic field induced emf problems. We found that both languages had representational advantages not found in the other. We discuss the impact that the lack of a given feature has on the model builder and examine ways to add additional features to the languages while minimizing changes to each language.

1 Introduction

Doyle and Patil [Doyle and Patil 1991] argue that certain classification languages are representationally impoverished because of the design decision which eliminates problematic representational features to achieve efficient response times. They advocate the retention of representational features with limitations on their use as an alternative to totally eliminating such features. We compare the knowledge representation languages CLASSIC [Brachman et al. 1991] and Algernon [Crawford and Kuipers 1991A] with this suggestion in mind.

We present the results of an experiment in which CLASSIC and Algernon were used to model the domain of magnetic field induced emf problems. The goal for the models was to recognize the spatial property of overlap between a pair of objects. Our objective was to determine the unique representational features of the languages, to determine how the lack of a feature affected the expressive power of a language, and to study means of incorporating missing features. In the remainder of this section, we provide a brief introduction to the knowledge representation features of CLASSIC and Algernon. Readers who are familiar with the two languages may proceed to Section 2.

*This paper was presented at the AAAI Workshop on Tractable Reasoning, AAAI-92, San Jose, CA. This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8905494 and IRI-8904454, by NASA grant NAG 2-507, by the Texas Advanced Research Program under grant no. 003658175, and by the Jet Propulsion Laboratory.

1.1 Descriptions of CLASSIC and Algernon

CLASSIC is a classification language with many restrictions on expressive power, as noted by the language designers themselves [Brachman et al. 1991]. Algernon is a general purpose language for defining classes of concepts and for expressing and proving relations between concepts. The user is free to define any number of inference rules in access-limited logic [Crawford and Kuipers 1989] to perform a desired reasoning task. At the highest level, both are frame-based languages which provide constructs for defining concept classes, relations on classes, and properties of individuals. Classes are organized in taxonomic hierarchies. Both languages are, in general, monotonic and operate with an open-world assumption. Algernon does provide a limited set of non-monotonic operators to allow reasoning with assumptions and to allow negation-by-failure.

The languages differ most in the way in which concepts are defined and the methods by which relations may be inferred. The CLASSIC user defines concepts through specifying membership criteria. CLASSIC then derives the taxonomic hierarchy from these definitions. The Algernon user typically provides a taxonomic hierarchy of classes, and writes inference rules to prove relations on classes.

CLASSIC is designed for a specific purpose, classification and contains an intrinsic inference method to prove set membership (isa) relations between concepts and individuals. CLASSIC provides only a weak treatment of other types of relations, in that the user has little flexibility to define additional inference rules. The CLASSIC user may define if-added rules and attach these to class definitions, but the power of these rules is limited as only ground sentences may be asserted. In contrast, Algernon is a system designed to describe and prove any user defined relation, where ISA is only one such relation. The Algernon user may write both if-added and if-needed types of inference rules using universally quantified variables.

1.1.1 A CLASSIC Example

Figure 1 shows a partial CLASSIC knowledge for describing magnetic fields. We describe a three tier hierarchy, with *objects* at the top, *magnetic fields* as a subset of *objects*, and *steady-uniform-magnetic-fields* as a subset of *magnetic fields*. The 'primitive' statement declares *objects* to be a primitive concept to be named as objects. CLASSIC also allows one to define disjoint sets, where an individual which is found to belong to one such set may not belong to any of the other sets in the disjoint grouping.

The description of magnetic-fields includes conditions that state that a candidate must first belong to the set objects, and must satisfy the object-type relation with the filler magnetic-fields. The definition of steady-uniform-magnetic-fields includes an if-added rule that states all individuals found to be a member of this set have a field direction along the Z-axis.

The create-ind description creates *field*, an object which satisfies the conditions of the sets magnetic-fields and steady-uniform-magnetic-fields. While the if-added rule for steady-uniform-magnetic-fields states that the field direction is one of the positive or negative Z-axis, in our individual description, we state explicitly that the field direction is along the positive Z-axis.

1.1.2 An Algernon Example

Figure 2 shows a simple Algernon description of magnetic fields. The Taxonomy defines the hierarchy of concepts *objects*, *magnetic-fields*, and *steady-uniform-magnetic-fields*. We also define the set, *object-properties*, with *field-types* being a subset. Field-types is defined to have at least two individual members, *Steady* and *Uniform*.

Field-type-of is then declared to be a relation between the set of magnetic-fields and the set of field-types. A forward chaining rule is included to infer that certain magnetic fields can belong to the set steady-uniform-magnetic-fields. Finally, an individual called *field* is created. Algernon discovers that *field* is a member of the set magnetic-fields when a field-type-of relation is defined for it. The typing on the field-type-of relation states that the first argument must be a magnetic field. The forward chaining rule can then fire to conclude that field is also a steady-uniform-magnetic-field.

2 Results

This section describes our experiences in building domain models in CLASSIC and Algernon. Both models contain definitions of concepts and relations, and operate on descriptions of individuals. For the Algernon model, a taxonomic hierarchy of object classes (concepts) was provided explicitly, and many inference rules were written to prove relations between these concepts. The CLASSIC model builder gains an intrinsic inference mechanism for proving subsumption relations [Resnick et al. 1991], but has little freedom to define additional inference rules. We therefore organized the domain knowledge into a hierarchy of concepts, and considered the CLASSIC model to be successful if the individuals in a given world description were properly classified. Therefore, some definitions which were expressed as relations in Algernon, such as overlap between two objects, were defined as concepts in CLASSIC.

We were able to recognize overlap with each of the models, but the Algernon model was far easier to create and use. In particular, we found that CLASSIC is limited most by the fact that it is designed to operate on one individual at a time. We found that we often had to resort to CLASSIC's TEST-C mechanism [Resnick et al. 1991] to access more than one object at once, and to prove that certain relationships were true in the domain. The TEST-C mechanism allows the user to use an external LISP function to determine a truth value for an expression. The LISP function cannot have side effects, and may return only three possible values: T if the expression is provable, F if the expression proven false, and ? if T or F cannot be returned. Although the CLASSIC designers may intend that the TEST-C mechanism is to be used only for simple tests, our LISP functions were fairly complex, and involved such potentially inefficient operations as searches of the knowledge base.

We found that both CLASSIC and Algernon had representational advantages not found in the other. Our results are summarized in Table 1. The first column lists several methods of defining concepts that Doyle and Patil [Doyle and Patil 1991] have found to be inexpressible in CLASSIC-like languages. Under the columns CLASSIC and Algernon, we note whether a given definitional form could be represented in each language, and if not, whether a simple change could be sufficient to capture that knowledge.

2.1 Explanation of the Tabulated Results

This section provides an explanation of the results given in Table 1. We discuss if each of the following forms of knowledge could be represented in CLASSIC and Algernon, and if not, whether a simple addition to the language could improve matters. Other than negation by failure, Algernon already provides the ability to express and reason about all the definitional forms discussed below. Therefore, we concentrate primarily on describing the difficulties encountered in using CLASSSIC.

Disjunction Disjunctive definitions for a concept can be expressed in Algernon by writing a unique inference rule to cover each case of a disjunctive definition. CLASSIC allows only one definition for a concept, so the model builder cannot capture disjunction by providing multiple definitions. Alternate definitions can be captured by defining multiple subclasses of a common parent concept, but these are treated by CLASSIC as descriptions of distinct concepts, rather than alternate definitions for the same concept.

Both CLASSIC and Algernon allow the definition of a disjunctive set of values for a relation. In general, the value set is defined implicitly by specifying a class whose individual members are to be legal values. CLASSIC also allows the model builder to provide an enumerated list of possible values through its ONE-OF roles restriction operator [Brachman et al. 1991].

Explicit lists of values are more difficult to include in an Algernon model. Lists may be asserted as values directly, but this strategy requires external LISP calls to operate on the list. Alternatively, since Algernon allows the model-builder to modify the taxonomy hi-

```
(cl-define-concept 'objects
                   '(and (primitive classic-thing objects)))

(cl-define-role 'object-type)
(cl-define-concept 'magnetic-fields
                   '(and objects (fills object-type magnetic-field)))

(cl-define-role 'field-type)
(cl-define-role 'field-direction-of)
(cl-define-concept 'steady-uniform-mag-fields
                   '(and magnetic-fields (fills field-type steady)
                                         (fills field-type uniform))
                   '(all field-direction-of (and coordinate-axes
                                                 (one-of z-axis neg-z-axis))))

(cl-create-ind 'field '(AND OBJECTS
                            (fills object-type magnetic-field)
                            (fills field-type steady)
                            (fills field-type uniform)
                            (fills field-direction-of z-axis)))
```

Figure 1: Examples of CLASSIC Definitions

```
(a-assert "Taxonomy"
   '((:taxonomy
       (objects (magnetic-fields (steady-uniform-magnetic-fields)))
       (things (object-properties (field-types steady uniform))))))

(a-assert "New slots"
   '((:slot field-type-of (magnetic-fields field-types))
     (:rules magnetic-fields
       ((field-type-of ?field steady)
        (field-type-of ?field uniform)
        ->
        (isa ?field steady-uniform-magnetic-fields)))))

(a-assert "defining a field"
   '((:create ?obj1 field)
     (field-type-of field steady)
     (field-type-of field uniform)))
```

Figure 2: Examples of Algernon Definitions

Category	Algernon	CLASSIC
Disjunction	Allows disjunctive definitions	Allows enumerated sets
Negation	No Negation by Failure / Improvement Possible	Limited
Equivalence	YES	NO / With Limited Use of Variables
Particularization	YES	YES, Provided by FILLS
Transitive Relations	YES	NO / NO
Functions Over Ordered Sets	Indirectly	Handles Some Cases via TEST-C / Allow TEST-C to return values
Mapping Between Ordered Sets	Indirectly	Handles Limited Cases via TEST-C / Requires Greater Use of Variables
N-ary Relations	YES	NO
Typed-Slots	YES	NO / Simple Extensions Possible

Table 1: The items in the first column, except typed-slots, are categories of definitional forms as given by Doyle and Patil. The CLASSIC and Algernon columns list whether a given definition can be expressed in the language, and after the bar, indicate if the limited use of a new construct will be beneficial.

erarchy of classes, a new class can be created whose members are the values given in the enumerated set. An additional inference rule is then required to enforce the condition that membership in the new class is closed. Otherwise, given an assertion *(relation individual value)*, where *value* is not in the enumerated set, Algernon would simply add the value to the set as a newly discovered piece of knowledge instead of outputting an error message.

Negation Algernon supports classical negation, and provides limited support for negation-by-failure through the non-monotonic *:unp* operator [Crawford and Kuipers 1991C]. That is, Algernon can conclude that an assertion is not true of the knowledge base through proof by contradiction, but because of its open-world assumption, Algernon will not conclude NOT A from the observation that A is currently not true in the knowledge base, unless explicitly directed to do so.

In general, neither Algernon nor CLASSIC support negation-by-failure since they operate with an open-world. Algernon's *:unp* operator allows negation-by-failure under the assumption of a closed world [Crawford and Kuipers 1991B], and the Algernon user can write a rule (NOT A) <- (*:unp* A) to conclude NOT A if A is currently unprovable. The problem is that A could later become true.

CLASSIC allows negation-by-failure to be a sound inference method by allowing the user to explicitly close the world regarding the fillers of a role (relation). The model builder can state the number of fillers that a role can have, and once all the role fillers have been found for an object of interest, CLASSIC will automatically close the world [Resnick et al. 1991], thus allowing inferences involving negation-by-failure. The actual inference has to be done in LISP through the TEST-C mechanism since CLASSIC does not provide a NOT operator.

Algernon currently provides the *:cardinality* keyword argument in definitions of relations [Crawford and Kuipers 1991B] to define the maximum number of fillers. This mechanism could be used in a manner similar to CLASSIC to explicitly close the world when all the fillers for a relation have been found. Then, Algernon could use negation-by-failure to infer NOT A from the fact that A is currently unprovable.

Equivalence Under the category of equivalence, we place the need to express that the value of one role is equal to the value of another role. CLASSIC's SAME-AS restriction [Resnick et al. 1991] allows one to express the condition that the value returned by two different access paths must be the same. However, there isn't a means to infer a value for a role filler. This presented many difficulties. Consider the following:

A physical support exerts a force on the objects it supports. We defined two different concepts, supporting-object and supported-object, with roles force-exerted-by and force-on, respectively. The concept supported-object also had a role, supported-by, to identify the supporting object. The same individual (a force) fills both force-exerted-by and force-on, but we could not find a method to allow CLASSIC to infer the filler of one role from knowledge of the filler of the other.

This problem could be solved by allowing a call to an external function such as TEST-C to return a value for the FILLS restriction. Alternatively, variables can be used to hold values returned by following a restricted access path, and the variables could then be used to bind roles in a FILLS restriction. The efficiency of the system will not be greatly affected if the search space for determining the binding of a variable is constrained.

Transitive Relations Another consequence of the lack of variables in CLASSIC is the inability to write inference rules to derive new knowledge from the facts already in the knowledge base. We needed to express ordinal relationships between the positions of vertices of objects, and had to enter all such relationships by hand since we could not write inference rules (e.g., transitivity) to do the job. In general, the inability to define inference rules to conclude new facts forces the model builder to enter complete knowledge about the world. We are not able to suggest methods other than adding a preprocessor to CLASSIC for eliminating this problem. The ability to add arbitrary inference rules appears to require drastic changes in the CLASSIC implementation - on the order of transforming CLASSIC into Algernon.

Functions Over Ordered Sets Algernon and CLASSIC both operate on individual objects rather than sets of objects, with the exception of CLASSIC's ability to handle enumerated sets. Neither language provides a mechanism to define a sequence, for example. As with enumerated sets, the Algernon model builder may explicitly provide a sequence as the value of a slot (relation) and operate on this ordered set through external LISP calls. The CLASSIC user may similarly access LISP through the TEST-C mechanism. A set of all bindings of a role may be created and operated on, allowing such operations as greatest and least to be performed. However, in the current implementation, the TEST-C function cannot return any values to be stored in the knowledge base. The previously suggested solution of allowing the TEST-C function to return a value for a FILLS restriction applies here as well.

As with simple enumerated sets, ordered sets may be represented indirectly in Algernon by creating a new class to hold the set of legal values and by adding an inference rule to close membership in the class. The ordering between values can be captured by adding a new relation which pairs each value with an integer to represent its place in the ordered list. For example, given the ordered list (first second third), we could add additional relations such as (first 1) and (second 2) to capture the rank of each entry in the list. Any functions over ordered sets could be applied to the numerical rank of each of the values, and the results could be mapped back to the actual value set.

Note that this method could be used in a CLASSIC knowledge base as well to express the order of a list, although it would not be possible to operate on such a list directly within CLASSIC. An alternate method for implementing ordered sets in Algernon is to explicitly store order relations between the values in the set. This method is used by the qualitative process compiler, QPC [Crawford, Farquhar, and Kuipers 1990].

Mapping Between Ordered Sets Although we did not encounter the need to map between ordered sets, we feel that the methods used to represent and operate on individual ordered sets in Algernon could be easily extended to deal with multiple ordered sets. Whereas Algernon allows one to write inference rules to simultaneously examine the properties of multiple objects, CLASSIC is designed to examine one individual at a time and compare its properties against concept definitions to determine its place in the taxonomic hierarchy. It is particularly difficult, for example, to compare properties of objects which belong to the same class. In general, we found that we had to escape to the TEST-C mechanism whenever we wanted to examine the properties of two different objects at once.

For example, consider a concept such as OVERLAPPED-OBJECTS, which, in our model, is a property of two polygons. CLASSIC can only examine a single polygon and compare its properties against the restrictions of concept definitions to determine if it is an OVERLAPPED-OBJECT. We therefore included a TEST-C restriction in the definition for the OVERLAPPED-OBJECT concept which attempts to find another polygon which is overlapped with the one being examined. Unfortunately, when this test is successful, only one of the polygons is classified as an overlapped-object! We could not find a means to store the identity of both polygons.

Again, since CLASSIC already allows the user to perform searches of the knowledge base through LISP functions, it would be beneficial to allow limited searches through the built-in mechanisms of CLASSIC. For example, properties of multiple objects could be defined if a RETRIEVE restriction is added to access properties of another object to compare against properties of the one being examined.

2-place vs. n-ary relations CLASSIC allows only 2-place relations. Algernon allows n-ary relations. Whereas it may be possible to build up n-ary relations from 2-place relations, the process is tedious at best, and involves the explicit creation of intermediate concepts that the user does not care about.

2.2 Other Observations

We wish to make a final point, based on our experiment, that is outside the scope of Doyle and Patil. This is the observation that unlike Algernon, the fillers of roles in CLASSIC can only be partially typed - by using an ALL role restriction inside a concept or individual definition. For example, the Algernon user can define the relation 'father-of' as (:slot father-of (people people)), to state explicitly that father-of is a relation between members of the set people. The CLASSIC definition would be (cl-define-role 'father-of). Inside the concept definition for people, we could add a restriction that all the fillers of the father-of role must be people. However, this does not prevent the concept COORDINATE-AXES from having a father-of role!

In general, typed relations allow the user to express clearly the intent of each role, and allow the system to flag type errors. CLASSIC roles have little semantic meaning attached to them, and the user may attach any role to any object. A CLASSIC user must be very careful about the names chosen for roles, as the name of the role is the only clue the user has regarding the purpose of a particular role. This situation makes a knowledge base much more difficult to debug and maintain. Since concepts are well defined in CLASSIC, roles could easily be typed by concepts to be related without much effect on efficiency.

3 Conclusions

We have described the results of an experiment in which the knowledge representation languages, CLASSIC [Brachman et al. 1991] and Algernon [Crawford and Kuipers 1991B] were used to model the domain of magnetic field induced emf problems, and have used our experimental results to extend the observations of Doyle and Patil [Doyle and Patil 1991] regarding the features of the CLASSIC-like languages. We found that both languages had features not found in the other,

For example, CLASSIC can handle some cases of negation-by-failure through its ability to partially close the world, while Algernon only allows negation-by-failure through the non-monotonic *:unp* operator. CLASSIC allows enumerated lists to be represented directly, while the Algernon user can represent enumerated and ordered lists only in an indirect manner. However, we also found that Algernon knowledge bases are easier to create and maintain since definitions of relations are fully typed. CLASSIC roles are not typed, and therefore provide little information about their intended use. Since concepts in CLASSIC are well-defined, we feel the language could be modified without great difficultly to enforce typing of roles by the concepts to be related.

Doyle and Patil conclude that users of CLASSIC-like systems will often resort to ad-hoc methods to overcome the limitations in the language. We found that we made ad-hoc use of the TEST-C mechanism to overcome the lack of expressive power in the remaining constructs of CLASSIC. We needed the TEST-C mechanism to examine multiple objects at once and to search for and store temporary information necessary to draw additional inferences. Additional difficulties in using CLASSIC arose from the fact that, in general, CLASSIC constructs do not allow the fillers of roles to be computed. Since TEST-C functions often do the necessary work to compute a value of interest, at the very least, CLASSIC should allow the value to be returned to be stored in the knowledge base.

A better solution is to allow the limited use of variables in CLASSIC functions. This will reduce the dependency of the user on the TEST-C mechanism. We have found that our use of variables falls into three classes. The first use is to store a value obtained by following a narrowly constrained access path, and to later compare this value against other values or to bind the value as the filler of another role. The second use of variables is in examining multiple objects at once. A variable can be used to store the identity of each object, and can be used to retrieve additional properties of these objects. Search will be required to find the additional objects, and will be based on a description of the objects. The form of the descriptions could be heavily constrained to reduce the search space. The third use of variables is in general purpose inference rules such as a rule for expressing transitive relations. We feel that this feature cannot be added to CLASSIC through simple changes alone, as methods for constraining the search space will be complicated. Adding the last feature will effectively convert CLASSIC into Algernon.

In general, the problem with introducing variables in CLASSIC is that determining the bindings for variables involves search, and this could adversely affect the run-time efficiency of the system. We argue that the LISP functions used to prove TEST-C restrictions simply transfer the inefficiency from CLASSIC constructs to user-defined external functions. We feel that constraining mechanisms could be introduced to reduce the search space and to improve efficiency, rather than eliminating the use of variables altogether.

4 Acknowledgements

I would like to thank Benjamin Kuipers, Bert Kay, and Adam Farquhar of the qualitative reasoning group, Jimi Crawford of AT&T Bell Laboratories and Neil Iscoe of EDS Austin for providing valuable comments.

5 References

[Brachman et al. 1991] R. Brachman, D. McGuinness, P. Patel-Schneider, L. Resnick. LIVING WITH CLASSIC: When and How to Use a KL-ONE-Like Language, In J. Sowa, Ed. *Principles of Semantic Networks: Explorations in the representation of knowledge* San Mateo, CA.: Morgon-Kaufmann.

[Borgida et al. 1989] A. Borgida, R. Brachman, D. McGuinness, L. Resnick. CLASSIC: A structured data model for objects. In Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, 59-67.

[Crawford and Kuipers 89] J. Crawford and B. Kuipers. Towards a Theory of Access-Limited Logics for Knowledge Representation. In Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning.

[Crawford, Farquhar, and Kuipers 90] J. Crawford, A. Farquhar, and B. Kuipers. QPC: A Compiler from Physical Models into Qualitative Differential Equations. In Proceedings of the Eigth National Conference on Artificial Intelligence. Boston, MA, 1990.

[Crawford and Kuipers 1991A] J. Crawford and B. Kuipers. Algernon - A Tractable System for Knowledge-Representation. In Proceedings of the AAAI Spring Symposium on Implemented Knowledge Representation and Reasoning Systems, Palo Alto, CA.

[Crawford and Kuipers 1991B] J. Crawford and B. Kuipers. Algernon User's Manual for Algernon Version 1.2, Technical Report, AI 91-166. Department of Computer Sciences, University of Texas at Austin.

[Crawford and Kuipers 1991C] J. Crawford and B. Kuipers. Negation and Proof by Contradiction in Access-Limited Logic. In Proceedings of the Ninth National Conference on Artificial Intelligence, 897-903.

[Devanbu et al. 1991] Devanbu, P., et. al., LaSSIE: A Knowledge Based Software Information System. *CACM* 34 (5): 35-48.

[Doyle and Patil 1991] J. Doyle and R. Patil. Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence* 48 (3): 261-297.

[Resnick et al. 1991] L. Resnick, A. Borgida, R. Brachman, D. McGuinness, P. Patel-Schneider, K. Zalondek. CLASSIC Description and Reference Manual For the COMMON LISP Implementation, Version 1.1. Technical Report. ATT Bell Laboratories, Murray Hill, NJ.