MANAGING TRACTABILITY IN AN OPEN ARCHITECTURE KNOWLEDGE REPRESENTATION SERVER

Brian R Gaines

Knowledge Science Institute
University of Calgary
Calgary, Alberta, Canada T2N 1N4
(403) 220-5901, gaines@cpsc.ucalgary.ca

Abstract: This paper describes the management of tractability issues in the design, implementation and application of an open architecture knowledge representation server. It is shown that the trade-off between representational power, deductive completeness and tractability in knowledge representation servers is not one of absolute design decisions, but rather one of server management through the provision of appropriate server layers and feedback on the expected cost of using certain services. In the short term it is feasible to develop kernel server modules optimized for efficient representation and constraint propagation within well-defined limitations verified against algebraic models. In the longer term it should be feasible to analyze generic problem areas in terms of the server architectures and underlying theory to come to understand issues of domain-dependent tractability.

INTRODUCTION

As the theoretical foundations of term subsumption knowledge representation systems become well-established, it is feasible to treat their representation and reasoning capabilities as standard services to be utilized by other, more specialized, knowledge-based systems. Since complexity analyses shows that such systems are able to represent inherently intractable problems (Brachman and Levesque, 1984), it has been suggested that both the representational and deductive capabilities of such services should be limited (Levesque and Brachman, 1987). For example, CLASSIC (Borgida, Brachman, McGuiness and Resnick, 1989) was designed for applications such as software engineering library maintenance, and its capabilities were restricted to attempt to avoid intractability. However, there remain intractable reasoning situations representable in CLASSIC (Nebel, 1990), and its extension to be deductively complete would add additional possibilities for intractability (Lenzerini and Schaerf, 1991). In the light of such results, some implementors are attempting to design "pay as you go" systems with rich representational power and deductive completeness, but subject to the computational penalties of worst-case intractability only on worst-case problems (Baader and Hollunder, 1991).

The reasons for and against restrictions on representational and deductive capabilities have been surveyed by Doyle and Patil (1991) who conclude that, while there are sound arguments for such limitations, the capabilities of the resultant systems will too often fail to satisfy reasonable application requirements. One impact of this is that designers could be forced to add functionality that provided the missing capabilities in ways that were less principled, and whose defects were less manageable, than those of a general server. A second is that problems might have to be represented in unnatural ways that were themselves conducive to errors and poor performance. A third is that the effect of the limitations on deductive capabilities might not be apparent to users, leading to serious errors in particular domains and applications.

This paper reports on experience of managing tractability issues in developing a light-weight open architecture knowledge representation server, initially based on CLASSIC, but now extended to a wide variety of data types and additional modes of reasoning.

MODELING COMPLEXITY ISSUES IN KR SERVERS

KRS (Gaines, 1991a) is a knowledge representation server designed to be used as an embeddable component in a wide range of systems. It is implemented as a class-library in C++ to make it available as an extensible system of components with an open architecture that can be enhanced with additional forms of representation and additional modes of reasoning. The reasoning services in the basic class library have been designed to take into account complexity analyses and provide layers of services that are explicitly distinguished in terms of the tractability/completeness trade-off. In particular the system is designed such that tractable deductions involving constraint propagation can be carried out rapidly by one process, while intractable one involving searches can be requested as required and delegated to background tasks that may be aborted if useful results are not obtained through use of allocated space/time resources. In addition meta-services are provided to report on the complexity of knowledge structures and bounds on search spaces, so that a system using the server has some basis for estimating the cost of a request.

This management of tractability issues has been structured in a general way to be largely independent of the data types available within the basic knowledge representation schema. New data types are themselves implemented through sub-service modules that interface to the kernel inference system through a small number of standard routines. The inferential routines are basically: computation of subsumption between types; classification of an instance against a type; and composition of two types. From a theoretical perspective, the sub-services are treated as 'oracles' in the complexity analyses such that the cost of their services are assumed known. From a practical perspective, the currently implemented sub-services deal with basic data types such as set constraints with upper and lower bounds and cardinality restrictions, or integer ranges, and do not themselves involve intractable computations.

The basic types are extended to complex types through the lattice-theoretic construction introduced by Cardelli (1984)

and Aït-Kaci (1986). This gives a type-theoretic model of inheritance in the T-box that allows semantic notions appropriate to knowledge representation to be introduced by interpreting a bounded lattice a *constraint algebra* in which the atoms can be thought of as *values*, other elements as *constraints* upon those values, ideals as *types* of values, the order relation as one of constraint *subsumption*, the incompatibility relation as one of *disjoint* constraints, the unit element as a *universal* constraint that is always satisfied and the zero element as an *incoherent* constraint which can never be satisfied. Complex types are introduced through a Cartesian product construction may be iterated indefinitely to construct a well-founded sequence of constraint lattices of increasing order corresponding to the well-founded construction of sets in classical set theory. The *order* of an element of one of the lattices is defined to be the position in this sequence of its initial definition. The *complexity* of an element is defined recursively as its arity plus the arities of each of its roles—this corresponds to the number of nodes in its expansion or graph. The dynamics of truth-maintenance in the A-box can also be interpreted within this construction as a trajectory of a world of individual states that, for monotonic reasoning, is essentially downwards in the lattice.

The following section gives a brief overview of the main features of CLASSIC and KRS, and the following sections develop detailed algebraic semantics for them in terms of the ψ-calculus. These are used to analyze the representational power and the tractability of deduction in the T-box and the A-box, showing how services may be layered, extended and instrumented in each subsystem.

OVERVIEW OF CLASSIC-LIKE SYSTEMS

The syntax and semantics of CLASSIC and related systems are well-documented in the references cited above. As an *aide memoire*, the syntax of concept definitions in the T-box and of individual assertions in the A-box are basically:

<concept-expr> ::=
  <concept-name> |
  (**AND** <concept-expr>$^{+}$) |
  (**ALL** <role-expr> <concept-expr>) |
  (**AT-LEAST** <positive-integer> <role-expr>) |
  (**AT-MOST** <non-negative-integer> <role-expr>) |
  (**SAME-AS** (<role-expr>)$^{+}$ (<role-expr>$^{+}$)) |
  (**ONE-OF** <individual-name>*) |
  (**NOT** <individual-name>*) |
  (**FILLS** <role-expr> <individual-name>$^{+}$) |
  (**CLOSE** <role-expr>) |
  (**PRIMITIVE** <concept-expr> <index>) |
  (**DISJOINT-PRIMITIVE** <concept-expr> <group-index> <index>) |
  (**TEST** <fn> <argument>*)

That is, concept expressions may be built up as the composition (through **AND**) of previously defined concept expressions and role expressions constrained by previously defined concept expressions (**ALL**), cardinality constraints (**AT-LEAST**, **AT-MOST**), equality constraints (**SAME-AS**) and extensional constraints (**ONE-OF**, **NOT**, **FILLS**, **CLOSE**).

In CLASSIC the **SAME-AS** constraint is defined to be applied to only single-valued roles to avoid known problems of tractability (Nebel and Smolka, 1990), but the problem can also be avoided by treating role fillers as set-valued. This changes the semantics of equality between roles in the multi-valued case when interpreted as set-valued, but has the advantage that single-valued roles do not have to be treated as a special case just for this one constraint.

The **NOT** constraint is peculiar to KRS and has the same semantics as **ONE-OF** except that the complement of a set of individuals is specified. This does not change the complexity of reasoning but allows negative restrictions to be placed on role fillers. This is significant in taking into account that new individuals can be generated later so that the complement cannot necessarily be enumerated when the constraint is defined. This extension also enables primitive and disjoint primitives to be implemented through hidden role fillers, unifying them theoretically.

In CLASSIC the **CLOSE** constraint is defined to be applied only to individual expressions, but it can be simply interpreted as syntactic sugar for the set being specified by **ONE-OF** being defined or asserted to be the same as that specified by **FILLS**. This interpretation has the intended semantics and the advantage that definitions in the T-Box and assertions in the A-box have the same form.

The **PRIMITIVE** constraint allows a concept definition to be defined to be incomplete so that it can only be asserted of an individual rather than computed as applying. The **DISJOINT-PRIMITIVE** constraint allows one primitive concept to be defined as incompatible with another. These allow natural kinds and their oppositions to be represented.

The **TEST** constraint allows an arbitrary function to be called with arguments that evaluates whether an individual to which the concept expression might be applied satisfies the test, does not satisfy the test, or is indeterminate in that future assertions may lead to either of these outcomes. This feature allows CLASSIC to be extended with additional datatypes but, since the semantics of the functions and datatypes are not specified, it creates a gap in the theoretical treatment of reasoning in CLASSIC. In KRS similar capabilities are provided by the capability to plug in additional datatypes through well-defined interfaces that can be modeled in terms of the constraint algebra.

CONSTRAINT ALGEBRAS

This and the following sections show that all the structures necessary to support the semantics of CLASSIC-like knowledge representation schema can be represented through one algebraic structure, that of a distributed lattice with zero and unit (Birkhoff, 1948). A number of different lattice structures are involved in modeling the full semantics, and these will be derived constructively by commencing with a simple lattice and extending it.

The semantics of a lattice are those of a pair of binary operations upon a set, L, each of which is idempotent, associative, commutative, and which together are mutually absorptive.

Idempotent $\quad x \in L \Rightarrow x \sqcap x = x, x \sqcup x = x \quad (L1)$

Commutative $x, y \in L \Rightarrow x \sqcap y = y \sqcap x$, $x \sqcup y = y \sqcup x$ (L2)

Associative $x, y, z \in L \Rightarrow x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$, $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$ (L3)

Absorptive $x, y \in L \Rightarrow x \sqcap (x \sqcup y) = x$, $x \sqcup (x \sqcap y) = x$ (L4)

The operation $\sqcap$ is termed the join, infimum or greatest lower bound (glb), and $\sqcup$ is termed the meet, supremum or least upper bound (lub). Note the symmetry of these definitions reflecting the duality of the lattice operations, a dual lattice is obtained by interchanging the $\sqcap$ and $\sqcup$ operations. A sublattice is generated by any subset together with the joins and meets of its elements. There is also a natural preorder on a lattice, $x \rightarrow y$, x is lower than y, y is higher than x, defined by:

Order $x \rightarrow y \Leftrightarrow x \sqcap y = x, x \sqcup y = y$ (L5)

An ideal, J, of a lattice, L, is a sublattice such that:

Ideal $x \in J, y \in L \Rightarrow x \sqcap y \in J$ (L6)

The ideal generated by a subset of L is the sublattice generated by the subset together with all lower elements.

In terms of the order relation, a bounded lattice has a unique lowest element, or zero, **O**, and a unique greatest element, or unit, **1**, such that:

Zero $x \sqcap \mathbf{0} = \mathbf{0}, x \sqcup \mathbf{0} = x$ (L7)

Unit $x \sqcap \mathbf{1} = x, x \sqcup \mathbf{1} = \mathbf{1}$ (L8)

There is a natural incompatibility relation on a bounded lattice, x—y, x and y are incompatible, defined by:

Incompatible $x—y \Leftrightarrow x \sqcap y = \mathbf{0}$ (L9)

An element is an atom if there is no lower element, other than **0**:

Atom $x \in J \Rightarrow a \sqcap x = x$ or $a \sqcap x = \mathbf{0}$ (L10)

Atoms generate 0-minimal ideals, that is ideals that properly contain no ideals other than **0**. An atom is incompatible with any element in an ideal that does not contain it.

An element, x', is a relative complement of an element x, relative to a lower bound, $\ell \rightarrow x$, and an upper bound, $x \rightarrow u$, if and only if:

Complement $x \sqcap x' = \ell, x \sqcup x' = u$ (L11)

If $\ell = \mathbf{0}$ and $u = \mathbf{1}$ then x' is just termed a complement. Complements are incompatible with one another in both their own lattice and its dual.

In many lattices derived from set theory and logic the operations are mutually distributive, and a lattice is termed distributive if it is such that:

Distributive $x, y, z \in L \Rightarrow x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z), x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$ (L12)

In a distributive lattice complements and relative complements, if they exist, are unique. A bounded distributive lattice in which all complements are defined is termed a Boolean algebra.

As noted above, semantic notions appropriate to knowledge representation can now be introduced by terming a bounded lattice a *constraint algebra* in which the atoms can be thought of as *values*, other elements as *constraints* upon those values, ideals as *types* of values, the order relation as one of constraint *subsumption*, the incompatibility relation as one of *disjoint* constraints, the unit element as a *universal* constraint that is always satisfied and the zero element as an *incoherent* constraint which can never be satisfied.

CONSTRAINT ALGEBRAS FOR THE T- AND A-BOX

To develop the semantics for concepts in the T-box and worlds of individuals in the A-box, take any constraint algebra, L, and consider an indefinite product of such algebras, $X = L \times L \times L \ldots$, indexed by a set of projections, Π, such that $\pi \in \Pi, x \in X \Rightarrow \pi x \in L$. The *arity* of an element of X is defined to be the number of non-1 projections. There is a natural constraint lattice formed by X under the definitions:

$x, y \in X \Rightarrow \pi(x \sqcap y) = \pi x \sqcap \pi y, \pi(x \sqcup y) = \pi x \sqcup \pi y$ (X1)

$\pi \mathbf{1} = \mathbf{1}$ (X2)

$\pi \mathbf{O} = \mathbf{O}$ (X3)

$\pi x = \mathbf{O} \Rightarrow x = \mathbf{O}$ (X4)

That is, the lattice operations operate on a component by component basis, and any projection becoming zero implies that the product is itself zero. The order relation of (L5) may also be computed on a component by component basis:

$x, y \in X, \; x \rightarrow y \Leftrightarrow \forall \pi \in \Pi \; \pi y \rightarrow \pi y$ (X5)

Note that even if the base lattice L is distributive, the derived lattice X will generally not be so.

The base algebra L may be treated as a sublattice of X by defining a specific $\mathbb{1} \in \Pi$ and the mapping, μ: $A \rightarrow X$:

$a \in A \Rightarrow \pi\mu a = a$ if $\pi = \mathbb{1}$, else $\pi\mu a = \mathbf{1}$ (X6)

(X1) through (X6) construct one constraint lattice as the indefinite product of another such that the original lattice may be mapped to a sublattice of the product. This construction may be iterated indefinitely to construct a well-founded sequence of constraint lattices of increasing order corresponding to the well-founded construction of sets in classical set theory. The *order* of an element of one of the lattices is defined to be the position in this sequence of its initial definition. The *complexity* of an element is defined recursively as its arity plus the arities of each of its non-unit projections—this corresponds to the number of non-unit modes in its expansion or graph.

T-box Representation

If the projections are defined to be *roles* and the lattice elements *concepts*, this construction is adequate to account for representation and deduction in the T-box of CLASSIC-like systems. Concept subsumption is the order relation of (X5) and individual classification is the same relation if the A-Box is taken simply to associate an individual with a concept asserted of it.

Some issues relating to the complexity of subsumption and classification are now apparent. The subsumption of (X5) for elements of a given order in a system that stores only non-unit projections and caches precomputed lower order subsumptions involves only n cache accesses where n is the minimum of the arities of the elements being compared. This model of subsumption computation gives an upper bound that makes it appear highly tractable. However, there are a number of reasons why this result is misleading.

First, the subsumption relation in the base lattice will generally not be cached because the space complexity will be high even if the time complexity is not, and the time complexity could also be high in an open architecture system in which arbitrary lattices can be plugged in (including the product lattices just defined). In practice, the base lattices currently used in CLASSIC-like systems are such that subsumption computation is tractable as will be shown in the next section.

Second, the composition of concepts may generate large numbers of additional concepts (anonymous concepts) such that the caching assumed may itself become intractable in space and time. Nebel (1990) has given a simple construction of a series of related concepts where the number of anonymous concepts grows exponentially with the number of originally defined concepts. In KRS the number of anonymous concepts generated by a given concept definition is reported as an auxiliary complexity measure.

Third, it is unrealistic to implement assertions about individuals within a caching scheme and individual classification may involve a full expansion of the concepts testing each node at the base lattice level. The number of tests is then the complexity defined above which can also grow exponentially with the number of concepts defined.

Fourth, the subsumption relation defined in (X5) is intensional. It does not take into account the additional constraints that may be propagated in the A-box to ensure that the concepts associated with individuals can not become incoherent. Extensional subsumption taking into account such constraint propagation is consistent with, but a stronger partial order, than intensional subsumption. This is how intractability may be shown in CLASSIC-like systems by representing satisfiability of a CNF formula with only positive or negative clauses (Lenzerini and Schaerf, 1991).

A-box Representation

If the projections are defined to be *individuals* and the lattice elements *worlds*, the construction, (X1) through (X5), is adequate to account for representation and some aspects of deduction in the A-box. World subsumption through the order relation of (X5) corresponds to monotonic reasoning, that the subsumed world can be reached from the other by a series of assertions about individuals. However, there are additional modes of reasoning in the A-box that make its analysis more complex.

First, CLASSIC-like systems implement rules as pairs of concepts such that if an individual is classified as falling under the first then the second is asserted of it.

Second, individuals may participate in base lattice constructions, for example through extension and cardinality constraints on sets of individuals, and hence assertions of a concept as applying to one individual may imply that other concepts apply to other individuals. In particular, cardinality constraints may imply the existence of (Skolem) individuals not previously created.

Third, as noted above, there may be assertions about individuals that, while they do not lead to incoherence directly, do so indirectly through the previous two mechanisms. Taking into account these implicit constraints may involve a search over possible worlds that grows exponentially with the number of individuals involved.

The first two modes of reasoning do not in themselves lead to intractability but, taken together with the third, they allow intractable problems to be represented in the A-box that go beyond those already discussed in relation to the T-box.

The representation of rules with exceptions can be used to support nonmonotonic reasoning in the A-box through normal defaults (Gaines, 1991b), and that determining possible worlds with consistent sets of defaults involves the same form of reasoning as that of the third mode above. **O**

BASE CONSTRAINT ALGEBRAS

Specific base lattices may be constructed from a four element constraint algebra in which **O** is the zero element, **I** the unit element, and **A** and **N** are complementary elements distinct from them. The semantics of this lattice is that of role constraints:

Indeterminate	**I**	a role possibly existing.
Overdeterminate	**O**	a role existing but unfillable.
Applicable	**A**	a role definitely existing.
Nonapplicable	**N**	a role definitely not existing.

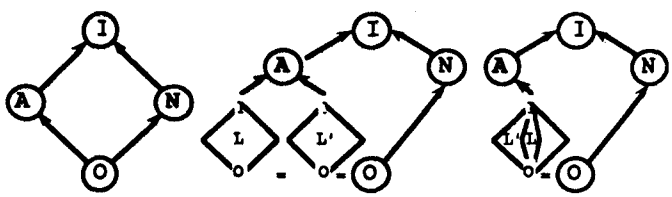

Figure 1 Constraint algebra and insertion of data types

Figure 1 on the left shows the structure of this lattice. It supports no data types in itself but it can be extended to support any arbitrary combination of data types by inserting their constraint lattices between **A** and **O** such that their unit elements represents the relevant types and their zero elements are mapped to **O**. Figure 1 center shows two mutually incompatible constraint lattices inserted in this way. Figure 1 right shows two compatible lattices inserted such that one is a sublattice of the other, corresponding to type coercion.

In CLASSIC-like systems there is a distinguished data type based on sets of individuals. The algebra of subsets of a set with set union as meet and set intersection as join is a constraint lattice. However, its semantics do not support the

distinction between **FILLS** and **ONE-OF** constraints, the **CLOSE** constraint, or cardinality constraints. The following constructions do so by combining the constraints of the algebra of subsets with those of the dual lattice.

Consider an arbitrary set, S, its powerset, $\mathcal{P}S$, and the Cartesian product on its powerset, $P = \mathcal{P}S \times \mathcal{P}S$. There is a natural constraint lattice formed by P under the definitions:

$$x = (l,u), y = (l',u') \in P \Rightarrow x \sqcap y = (l \cup l', u \cap u'),$$
$$x \sqcup y = (l \cap l', u \cup u') \quad \text{(P1)}$$

$$x = (\{\},S) \Leftrightarrow x = \mathbf{1} \quad \text{(P2)}$$

$$x = (l,u), l \not\subset u \Leftrightarrow x = \mathbf{0} \quad \text{(P3)}$$

The semantics of this lattice are those of constraints upon the subsets of S with the first projection of the product being a lower bound corresponding to **FILL**, the second projection being an upper bound corresponding to **ONE-OF**, unsatisfiable constraints in which the lower bound is not contained in the upper bound mapping to zero, and the lattice operations corresponding to complementary pairs of intersections and unions of lower bounds and upper bounds. The atoms of this lattice are elements in which the lower and upper bounds coincide. They provide the semantics for **CLOSE** in corresponding to a set constrained to consist precisely of its specified members.

The semantics of **NOT** in KRS are represented by having a complement as the second projection, that is S-u rather than u. Complements behave well under intersection and union with sets and other complements, generating either sets or complements, and causing no problems in implementation. In particular, this allows a very simple implementation of disjoint primitives as shown later.

Cardinality constraints may be added to this structure to give an extended constraint lattice based on the product $C = \mathcal{P}S \times \mathcal{P}S \times I \times I$, where I is the set of non-negative integers. There is a natural constraint lattice formed by C under the definitions:

$$x = (l,u,m,n), y = (l',u',m',n') \in C \Rightarrow$$
$$x \sqcap y = (l \cup l', u \cap u', \max(m,m', \mathcal{C}(l \cup l'), \min(n,n', \mathcal{C}(u \cap u'))))$$
$$x \sqcup y = (l \cap l', u \cup u', \min(m,m', \mathcal{C}(l \cap l'), \max(n,n', \mathcal{C}(u \cup u')))) \quad \text{(C1)}$$

$$x = (\{\},S,0,\mathcal{C}S) \Leftrightarrow x = \mathbf{1} \quad \text{(C2)}$$

$$x = (l,u), l \not\subset u \vee m > n \Leftrightarrow x = \mathbf{0} \quad \text{(C3)}$$

The semantics of this lattice are those of constraints upon the subsets of S as before with the additional constraints on cardinality corresponding to **AT-LEAST** and **AT-MOST**.

The **PRIMITIVE** and **DISJOINT-PRIMITIVE** constraints may be implemented through a hidden role filled by elements of a P lattice based on the set of concepts rather than that of individuals. A concept defined to be primitive stores its own name as **FILLS** in this role. A concept defined to be disjoint to another stores the other's name as **NOT** in this role. This has the required semantics and is the form of implementation in KRS.

ARCHITECTURAL IMPLICATIONS

The analysis of CLASSIC-like systems in terms of lattice structures allows modular, open implementation architectures to be developed that are instrumented in terms of complexity and layered in terms of services.

The kernel system is the T-box defined based on the base constraint algebras defined. This supports concepts and roles based on an extensible set of data types including open and closed sets of individuals. It is subject to intractability through the generation of anonymous concepts and through the expansion of concepts to classify individuals. This is unavoidable but instrumentable through the complexity measures defined. KRS provides statistics on the complexity of concepts defined and the number of anonymous concepts generated. For example, in the knowledge modeling application involving a room allocation problem circulated as part of the *Sisyphus* project (Gaines, 1991c): the maximum concept complexity is 143; the maximum complexity of the premise of a rule is 11; and the proportion of anonymous concepts is 117 out of 161. This contrasts with Nebel's (1990) worst-case construction where the maximum concept complexity grows to 3,584 with 78 anonymous concepts for 10 defined concepts, and to 3,687,788 with 1683 anonymous for 20 defined concepts. This illustrates the difference between worst case complexity analyses and domain complexity analyses (Vilain, 1991) and the need for instrumentation that can be applied to specific problem areas and problems.

There are two major limitations to the kernel system. The first is in representational power. It has been noted that the product lattices used to model roles are not distributive even if their base lattices are, and that the P and C set constraint lattices are not distributive anyway. Thus, even though complements and relative complements can always be constructed, they are not necessarily unique and the lattices are not Boolean algebras. This is the source of one problem in the semantics of CLASSIC-like knowledge representation schema, that the assertion of a general negation does not lead to a well-defined knowledge structure, that is a unique lattice element, but rather a set of elements whose management in an implementation is problematic. A second source of semantic problems is related to this in that the join operation in the lattice corresponds to logical conjunction, but the meet operation does not so correspond to logical disjunction—it is a generally a substantially weaker constraint. Hence again, assertions of logical disjunctions requires the management of a set of elements.

This should not be taken as implying that the lattice meet operation is somehow incorrect. It corresponds to the *minimum common generalization* of two concept structures and is an important operation in reasoning by analogy (Leishman, 1989). What is needed to support the representation of general disjunction and negation is the capability to supplement the lattice with a fine structure of meets and complements that does capture the semantics of disjunction and negation. Constructions to do this (Aït-Kaci, 1986) and algorithms to compute subsumption (Schmidt-Schauß and Smolka, 1991) have been given. However, the

expansion in the size of the lattice, from 3^n elements for P above to 2^{2^n} for the full structure, leads to intractability.

There are two approaches to extending the representation of CLASSIC-like systems to include general disjunction and negation. The first is to offer them in the query language rather than in the assertional language (Lenzerini and Schaerf, 1991), since the problems largely arise from managing the multiple possible worlds resulting from disjunctive and negative assertions. The second is to allow them in the assertional language but manage them effectively so that they do not adversely affect speed or space if not used, and affect it only to the extent necessary if used (Baader and Hollunder, 1991). In presenting the server to users, particularly in knowledge modeling applications, it is helpful to make it clear that the use of disjunctive and negative definitions may lead additional complexity in deduction which should be avoided if possible. The layer being developed for this in KRS uses a light-weight possible worlds implementation similar to that of an ATMS to represent disjunction and negation, and feeds back to users the number of worlds involved as a complexity measure.

The second limitation to the kernel system comes from the additional reasoning necessary in the A-box to determine whether additional constraints apply because some sets of additional assertions would lead to incoherency. It corresponds to unbound Skolem individuals that may not be freely bound within the scope of individual constraints since further deductions resulting from some bindings lead to incoherency. Again a possible worlds implementation provides for this form of reasoning and can be added to CLASSIC or KRS. The worse case intractability for this form of reasoning corresponds to it being essential to search all possible worlds. For some applications, such as design, such a search may be seen by users as an enumeration of possibilities and an inappropriate approach. In other applications, such as diagnosis, it may be a very effective strategy.

CONCLUSIONS

In conclusion, the trade-off between representational power, deductive completeness and tractability in knowledge representation servers is not one of absolute design decisions, but rather one of server management through the provision of appropriate server layers and feedback on the expected cost of using certain services. In the short term it is feasible to develop kernel server modules optimized for efficient representation and constraint propagation within well-defined limitations verified against algebraic models. In the medium term it is feasible to extend these through modular additions to be as representationally and deductively complete as required within theoretical limits. In the longer term it should be feasible to analyze generic problem areas in terms of the server architectures and underlying theory to come to understand issues of domain-dependent tractability.

REFERENCES

Aït-Kaci, H. (1986). An algebraic semantics approach to the effective resolution of type equations. **Theoretical Computer Science 45** 293-351.

Baader, F. and Hollunder, B. (1991). A terminological knowledge representation system with complete inference algorithms. pp.67-86 **Processing Declarative Knowledge**. Berlin, Springer. .

Birkhoff, G. (1948). **Lattice Theory**. Providence, Rhode Island, American Mathematical Society.

Borgida, A., Brachman, R.J., McGuiness, D.L. and Resnick, L.A. (1989). CLASSIC: a structural data model for objects. **Proc. 1989 SIGMOD Conference on the Management of Data**. New York, ACM Press. 58-67.

Brachman, R.J. and Levesque, H.J. (1984). The tractability of subsumption in frame-based description languages. **Proceedings of AAAI-84**. pp.34-37. Morgan Kauffman.

Cardelli, L. (1984). A semantics of multiple inheritance. **Semantics of Data Types**. Berlin, Springer. 51-67.

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

Gaines, B.R. (1991a). Empirical investigations of knowledge representation servers: Design issues and applications experience with KRS. **ACM SIGART Bull. 2**(3) 45-56.

Gaines, B.R. (1991b). Integrating rules in term subsumption knowledge representation servers. **AAAI'91: Proceedings of the Ninth National Conference on Artificial Intelligence**. Menlo Park, California, AAAI Press/MIT Press. 458-463.

Gaines, B.R. (1991c). Organizational modeling and problem solving using an object-oriented knowledge representation server and visual language. **COCS'91: Proce. Conference on Organizational Computing Systems**. pp.80-94. ACM Press..

Leishman, D. (1989). Analogy as constrained partial correspondence over conceptual graphs. **Proceedings of KR'89: First International Conference on Principles of Knowledge Representation and Reasoning**. San Mateo, California, Morgan Kauffman. 223-234.

Lenzerini, M. and Schaerf, A. (1991). Concept languages as query languages. **AAAI'91: Proceedings of the Ninth National Conference on Artificial Intelligence**. Menlo Park, California, AAAI Press/MIT Press. 471-476.

Levesque, H.J. and Brachman, R.J. (1987). Expressiveness and tractability in knowledge representation and reasoning. **Computational Intelligence 3** 78-93.

Nebel, B. (1990). **Reasoning and Revision in Hybrid Representation Systems**. Berlin, Springer.

Nebel, B. and Smolka, G. (1990). Representation and reasoning with attributive descriptions. **Sorts and Types in Artificial Intelligence**. Berlin, Springer. 112-139.

Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. **Artificial Intelligence 48**(1) 1-26.

Vilain, M. (1991). Deduction as parsing: tractable classification in the KL-ONE framework. **AAAI'91: Proceedings of the Ninth National Conference on Artificial Intelligence**. Menlo Park, California, AAAI Press/MIT Press. 464-470.