Feasibility of Optimised Disjunctive Reasoning for Approximate Matching

Ian Horrocks[1], Lin Padgham[2], and Laura Thomson[2]

[1] Dept. of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk
[2] Dept. of Computer Science
Royal Melbourne Institute of Technology
Melbourne, VIC 3001, Australia
{linpa, laura}@cs.rmit.edu.au

Abstract. Description logics are powerful knowledge representation systems providing well-founded and computationally tractable classification reasoning. However recognition of individuals as belonging to a concept based on some approximate match to a prototypical descriptor has been a recurring application issue as description logics support only strict subsumption reasoning. Expression of concepts as a disjunction of each possible combination of sufficient prototypical features has previously been infeasible due to computational cost. Recent optimisations have greatly improved disjunctive reasoning in description logic systems and this work explores whether these are sufficient to allow the heavy use of disjunction for approximate matching. The positive results obtained support further exploration of the representation proposed within real applications.

1 Introduction

Description Logic systems are knowledge representation systems based on First Order Logic or a subset of FOL chosen specifically for computational reasons. They have a Tarski-style semantics and a syntax which is particularly well suited to an object-oriented approach to describing concepts (classes) and individuals. Key functionality of DL systems is based on the notion of subsumption testing which is used both in building the class hierarchy and in recognising which concepts individuals belong to. This class of systems have been successfully used in a range of applications (e.g. [Berman *et al.*, 1994]). However a number of other applications have experienced difficulty due to the fact that recognition of individuals as belonging to a concept is done on the basis of the individuals having all characteristics defined for that concept. There is no notion in DLs of exceptions or of default or typical characteristics.[1]

We argue that in many applications what one wants is a notion of individuals being recognised as members of a concept, based on having a sufficient number of a cluster of

[1] Some work has been done on extending DLs to include defaults (e.g. [Baader and Hollunder, 1993, Padgham and Nebel, 1993, Padgham and Zhang, 1993]) but this is primarily directed towards default reasoning, rather than the issue of using defaults for recognition as is being explored here.

typical characteristics (plus possibly some necessary characteristics), rather than simply a fixed set of characteristics.

For example Coupey and Fouquere describe how for recognising faults in a telecommunications application it is absolutely necessary to be able to take account of default characteristics [Coupey and Fouquere, 1997]. However they take an approach of requiring that individuals explicitly have an exception to a default characteristic to allow recognition. This can be awkward, and is not always even possible. A medical diagnosis application [Padgham and Zhang, 1993] similarly needs to use typical symptoms to describe diseases, but does not want an individual presentation not to be recognised because it does not have **all** typical symptoms.

In order to meet the needs of the many applications similar to these it is necessary to be able to define a set of typical characteristics associated with a concept. An individual belonging to the concept is required to have some "critical mass" of these characteristics, and *should be recognised as belonging to the concept on this basis*. The most obvious way to achieve this within the semantics of first order logic (and Description Logics) is to define the concept as a disjunction of all the combinations of sufficiently many typical characteristics.

In an example from [Padgham and Zhang, 1993], chronic pyelonephritis is described as having the characteristics urine.dysuria, urine.casts, fatigue and urine.bacteria. Defining the concept on the basis of 75% of these characteristics would give us:

$$\begin{aligned} CPN \doteq\ & (urine.dysuria \sqcap urine.casts \sqcap fatigue) \sqcup \\ & (urine.casts \sqcap fatigue \sqcap urine.bacteria) \sqcup \\ & (fatigue \sqcap urine.bacteria \sqcap urine.dysuria) \sqcup \\ & (urine.bacteria \sqcap (urine.dysuria \sqcap urine.casts) \end{aligned}$$

Previously the intractability of the algorithms used for reasoning with disjunctions has meant that heavy usage of disjunctions is not a viable option computationally for real application systems.

For example, KRIS [Baader and Hollunder, 1991], one of the first DL systems that included principled reasoning with disjunction at all, exhibits very poor performance when reasoning with knowledge bases (KBs) containing significant numbers of disjunctive concepts.

Optimizations of KRIS that allowed it to obtain similar performance characteristics to CLASSIC [Baader *et al.*, 1992] (the most efficient of the set of tested DL systems [Heinsohn *et al.*, 1992]), did not address optimisations for disjunctive concepts (which cannot be represented in CLASSIC).

The new algorithms and optimisation techniques recently developed allow the typical case reasoning performance of DL systems to be radically improved [Horrocks, 1998]. These optimisations are particularly effective with respect to disjunctive reasoning. However there has been no experimentation which pushes the limits of these new algorithms, or examines whether they are adequate for particular application oriented needs which require heavy use of disjunction.

The work presented in this paper explores whether these techniques are in fact sufficiently powerful to support the routine use of disjunctive concepts to address the application issue of approximate matching to a prototype for recognition of individuals. Section 2 describes a representational model for defining concepts; section 3 describes in some detail the problem with disjunctive reasoning and the optimisations used in the FaCT system, which we hope will make the proposed representation viable. Section 4 describes the experiments done to investigate this viability and the results obtained. The results appear promising and we are building a bibliographic database application based on the techniques described, to further investigate the mechanisms within a genuine application.

2 Representation of Concepts

Literature from cognitive psychology supports the idea that when people think in terms of concepts, they actually think in terms of prototypical descriptions, rather than in terms of strictly necessary characteristics [Rosch, 1975]. However using a prototypical description for a concept descriptor in description logic systems (or any other system based on first order logic) will cause problems, as some sub-concepts as well as individuals will not have all characteristics of the prototype. In terms of recognising individuals, or automatically classifying sub-concepts, the prototypical description of the concept is over-defined. On the other hand, use of only necessary characteristics in defining a concept results in concepts being under-defined, with consequent lack of discrimination.

Earlier work by Padgham and others [Padgham, 1992, Padgham and Zhang, 1993] has explored describing concepts using two descriptors - a *core descriptor* for defining the strictly necessary characteristics and a default descriptor, (which we will call the *prototype descriptor*) which is subsumed by the core and in addition defines the prototypical characteristics. However this mechanism does not explicitly offer any assistance in recognising the specific concept an individual belongs to in cases where the core is under-defined and the individual does not fit the full prototype descriptor.

We build on this work by also defining a *basic descriptor* which explicitly captures the space of concept descriptions which are sufficiently close to the prototype descriptor that individuals subsumed by the basic descriptor should be recognised as instances of the concept.

The form of the basic descriptor is an "or" statement which defines any combination of 70% of the "features"[2] used in the prototype descriptor. The basic descriptor thus subsumes the prototype descriptor and an individual should be recognised as being an instance of a concept X based on subsumption by the basic descriptor for X.

Once users or application developers have defined the core and prototype descriptors the definition of basic descriptors can be automated. It would also be possible to generate descriptors capturing varying levels of agreement with the prototype (e.g. 90%, 70%, 50%) in different structures, allowing applications to attempt instance inference, or recognition of individuals at various levels of closeness to the prototype descriptor.

Further extensions where characteristics within a prototype can be grouped, requiring some critical mass in each group, can also be envisaged. However all these refine-

[2] Further investigation is needed regarding constraints that may need to be placed on the form of prototype descriptors. However this is outside the scope of the initial explorations presented in this paper. The agreement level of 70% may also be subject to variation.

ments rely on the adequacy of the optimisations being explored to provide computational viability when relatively large 'or' clauses are routinely used.

3 Subsumption Involving Disjunction

Description Logic systems provide a range of automated reasoning services, in particular inferring subsumption and instantiation (instance-of) relationships. Subsumption is the class/super-class relationship between concepts, while instantiation is the relationship between individuals and those concepts of which they are instances. The use of subsumption inference to build a concept hierarchy (partial order) is known as *classification* and the use of instantiation inference to determine the classes each individual belongs to is known as *recognition*.

A standard Tarski style model theoretic semantics is used to interpret descriptions and to justify inferences. The meaning of concepts and roles is given by an interpretation $\mathcal{I}$ which is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain (a set) and $\cdot^{\mathcal{I}}$ is an interpretation function. The interpretation function maps each concept to a subset of $\Delta^{\mathcal{I}}$, each role to subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual to a unique element of $\Delta^{\mathcal{I}}$. More complex descriptions can be built up by combining descriptions using a variety of operators, with the semantics of the resulting description being derived from its components.

A concept C is subsumed by (is more specific than) a concept D (written $C \sqsubseteq D$) if it can be inferred that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all possible interpretations $\mathcal{I}$. The result of classification procedures based on the subsumption relation is typically cached in the form of a directed acyclic graph called the concept hierarchy or taxonomy.

An individual x is an instance of a concept C (written $x \in C$) if it can be inferred that $x^{\mathcal{I}} \in C^{\mathcal{I}}$ for all possible interpretations $\mathcal{I}$. In many cases, instantiation reasoning, (or recognition), can be reduced to subsumption reasoning using either precompletion [Hollunder, 1994] or encoding [De Giacomo and Lenzerini, 1996] techniques; for this reason most recent studies have concentrated on subsumption reasoning. We follow this tradition and explore the tractability of recognition by obtaining experimental results for appropriate subsumption tests.

Most modern DL systems[3] perform subsumption reasoning by transforming the subsumption problem into an equivalent satisfiability problem: $C \sqsubseteq D$ if and only if the concept description $(C \sqcap \neg D)$ is *not* satisfiable. The satisfiability problem can then be solved using a provably sound and complete algorithm based on the tableaux calculus [Smullyan, 1968]. This approach was first described for the $\mathcal{ALC}$ DL and its practical application was demonstrated by the KRIS system.

The FaCT system uses an optimised implementation of a tableaux algorithm to perform subsumption reasoning. Like other tableaux algorithms it either proves the satisfiability of a concept C by constructing an example interpretation in which $C^{\mathcal{I}}$ has at least one member, or proves its unsatisfiability by demonstrating that all attempts to construct an example must lead to a contradiction. When C contains disjunction, trying to construct an example interpretation is non-deterministic. Earlier DLs dealt with this non-determinism by naively performing an exhaustive depth first search, and it is this

[3] At least those which provide sound and complete reasoning.

which leads to the poor performance of the KRIS system with highly disjunctive concepts. Although it still performs an exhaustive search, the FaCT system includes a range of optimisations which can dramatically reduce the size of the search space—these include the normalisation and encoding of concept descriptions, an improved search algorithm, the use of heuristics to guide the search, dependency directed backtracking, and the caching and re-use of partial results.

3.1 Example

A simple example illustrates the vital importance of optimisation techniques with the kinds of basic concept descriptors that will be generated using the representation discussed in section 2.

We will take a simple prototypical concept description consisting of only four "features" $\exists f_1.C_1 \sqcap \exists f_2.C_2 \sqcap \exists f_3.C_3 \sqcap \exists f_4.C_4$, where each of the C_i is a conjunction of three primitives such as $P_{i1} \sqcap P_{i2} \sqcap P_{i3}$, and generate a basic descriptor C_v that will subsume any conjunction containing at least two of the $\exists f_i.C_i$ terms:

$$\begin{aligned} C_v \doteq\ & (\exists f_1.C_1 \sqcap \exists f_2.C_2) \sqcup (\exists f_1.C_1 \sqcap \exists f_3.C_3) \sqcup \\ & (\exists f_1.C_1 \sqcap \exists f_4.C_4) \sqcup (\exists f_2.C_2 \sqcap \exists f_3.C_3) \sqcup \\ & (\exists f_2.C_2 \sqcap \exists f_4.C_4) \sqcup (\exists f_3.C_3 \sqcap \exists f_4.C_4) \end{aligned}$$

When classifying a concept $D \doteq \exists f_1.C_1 \sqcap \exists f_2.C_2$, it will be necessary to determine if C_v subsumes D. As described above, this will be transformed into a satisfiability test: C_v subsumes D iff $D \sqcap \neg C_v$ is not satisfiable. As a result of its being negated, the C_v part of this description becomes a conjunction of disjunctive clauses:

$$\begin{aligned} & (\exists f_1.C_1 \sqcap \exists f_2.C_2) \sqcap \\ & (\forall f_1.\neg C_1 \sqcup \forall f_2.\neg C_2) \sqcap (\forall f_1.\neg C_1 \sqcup \forall f_3.\neg C_3) \sqcap \\ & (\forall f_1.\neg C_1 \sqcup \forall f_4.\neg C_4) \sqcap (\forall f_2.\neg C_2 \sqcup \forall f_3.\neg C_3) \sqcap \\ & (\forall f_2.\neg C_2 \sqcup \forall f_4.\neg C_4) \sqcap (\forall f_3.\neg C_3 \sqcup \forall f_4.\neg C_4) \end{aligned}$$

To test the satisfiability of this concept, a naive tableau algorithm would try to build an example interpretation by proceeding roughly as follows:

1. Initialise the interpretation to contain a single individual x_0 which satisfies the concept. Expand all of the conjunctions, making it explicit that x_0 satisfies each of $\exists f_1.C_1, \ldots, (\forall f_3.\neg C_3 \sqcup \forall f_4.\neg C_4)$.
2. Search for a consistent expansion of the disjunctive concepts. Expand each unexpanded disjunction by selecting one of the disjuncts, backtracking and trying the other disjunct if that fails (leads to a contradiction). Typically, $\forall f_1.\neg C_1$ would be chosen from the first disjunction, $\forall f_2.\neg C_2$ from the fourth disjunction (disjunctions 2 and 3 are satisfied by the first choice), and $\forall f_3.\neg C_3$ from the last disjunction.[4]
3. Expand the $\exists f_i.C_i$ terms one at a time. For $\exists f_1.C_1$, this means creating a new individual x_1 satisfying the concept C_1 and related to x_0 by the role f_1. Due to the

[4] Completing all propositional reasoning before expanding $\exists R.C$ terms minimises space requirements [Hollunder and Nutt, 1990].

$\forall f_1.\neg C_1$ chosen from the first disjunction, x_1 must also satisfy $\neg C_1$. This seems to be an obvious contradiction, but as C_1 is actually the conjunction $P_{11} \sqcap P_{12} \sqcap P_{13}$, and $\neg C_1$ is the disjunction $\neg P_{11} \sqcup \neg P_{12} \sqcup \neg P_{13}$, discovering the contradiction in x_1 will mean expanding the conjunction and then searching the terms in the disjunction to discover that each choice leads to a contradiction with one of the expanded conjuncts.

4. Having discovered this contradiction, the algorithm will backtrack and continue searching different expansions of the conjunctions which x_0 must satisfy until it discovers that all possibilities lead to contradictions. It is then possible to conclude that $D \sqcap \neg C_v$ is not satisfiable, and that C_v thus subsumes D.

There are several obvious inefficiencies in this procedure, and some not so obvious. In the first place, there is the problem of the late discovery of "obvious" contradictions, for example when a complete (non-deterministic) expansion of C_1 and $\neg C_1$ is performed in order to discover the contradiction in x_1. This is a consequence of the fact that most tableaux algorithms assume the input concept to be fully *unfolded* (all defined concepts are substituted with their definitions), and in *negation normal form* (NNF), with negations applying only to primitive concepts [Hollunder and Nutt, 1990]. Arbitrary $\mathcal{ALC}$ concepts can be converted to NNF by internalising negations using DeMorgan's laws and the identities $\neg\exists R.C = \forall R.\neg C$ and $\neg\forall R.C = \exists R.\neg C$.

The KRIS system uses *lazy unfolding* to deal with the problem of late discovery, only unfolding and converting to NNF as required by the progress of the algorithm. Thus if C_1 were a named concept (introduced by a concept definition statement of the form $C_1 \doteq P_{11} \sqcap P_{12} \sqcap P_{13}$), then its unfolding would be postponed and the contradiction between C_1 and $\neg C_1$ immediately discovered. FaCT takes this idea to its logical conclusion by giving unique system generated names to all compound concepts. Moreover, the input is lexically analysed to ensure that the same name is given to lexically equivalent concepts. This means that the concepts $\exists f_1.C_1$ and $\forall f_1.\neg C_1$ would be named A and $\neg A$ respectively (for some system generated name A), and a contradiction would be detected without the need to create x_1.

Another problem with the naive search is that the same expansion can be explored more than once. For example, after some backtracking the algorithm will determine that choosing $\forall f_2.\neg C_2$ from the fourth disjunction always leads to a contradiction and will try the second choice, $\forall f_3.\neg C_3$. Expanding the fifth disjunction will then lead to $\forall f_2.\neg C_2$ being chosen, an identical solution to the first one. FaCT avoids this problem by using a *semantic branching* search technique adapted from the Davis-Putnam-Logemann-Loveland procedure (DPL) commonly use to solve propositional satisfiability (SAT) problems [Davis *et al.*, 1962, Giunchiglia and Sebastiani, 1996]. Semantic branching works by selecting a concept C such that C is an element of an unexpanded disjunction and $\neg C$ is not already in the solution, and searching the two possible expansions obtained by adding either C or $\neg C$. Wasted search is avoided because the two branches of the search tree are strictly disjoint. For example, when the choice of $\forall f_1.\neg C_1$, $\forall f_2.\neg C_2$ and $\forall f_3.\neg C_3$ leads to a contradiction, subsequent backtracking will cause the choice of $\forall f_2.\neg C_2$ to be changed to $\neg\forall f_2.\neg C_2$, so the first solution can never be repeated.

Finally, after the discovery of the contradiction in x_1, the naive search continues with *chronological* backtracking in spite of the fact that the contradiction was caused by $\forall f_1.\neg C_1$, the first choice made. FaCT deals with this problem by using *backjumping*, a form of dependency directed backtracking adapted from constraint satisfiability problem solving [Baker, 1995]. Each concept is labelled with a dependency set indicating the branching choices on which it depends, and when a contradiction is discovered the algorithm can jump back over intervening choice points without exploring alternative choices.

4 Empirical Investigations

An empirical evaluation was performed in order to determine the viability of using a real knowledge base developed using the representational model described in section 2. This evaluation used synthetically generated data in order to evaluate the performance of FaCT and to determine if the optimisation techniques described in Section 3.1 would be sufficiently powerful to permit empirically tractable reasoning with respect to the kinds of subsumption problem that would be encountered. The tests were also run using KRIS in order to identify levels which have previously caused problems, and as a way of identifying cases where FaCT may involve extra cost.

The testing used a variation of a random concept generation technique first described by [Giunchiglia and Sebastiani, 1996] and subsequently refined by [Hustadt and Schmidt, 1997]. The generated concepts are of the form $\exists f_1.C_1 \sqcap \ldots \sqcap \exists f_\ell.C_\ell$, where each f_i is an attribute (single valued role) and each C_i is a conjunction of n primitive concepts chosen from N possibilities.

For a given concept C and an approximation value V in the range 0–100, a concept C_v is formed, as in Section 3.1, consisting of a disjunction of all possible conjunctions containing $V\%$ of the $\exists f_i.C_i$ terms in C.[5] To represent the (hardest) kind of subsumption test that would be involved in the recognition process, a second concept C_r is formed from C by changing elements of the C_i from each $\exists f_i.C_i$ term so that C_r is subsumed by C_v with a probability P, and the time taken to test if C_v does in fact subsume C_r is measured. Varying ℓ (the number of "features") and V gives disjunctions of varying size, and varying P allows performance to be measured for tests ranging from "obvious" subsumption to "obvious" non-subsumption.

Initial explorations indicate that for a variety of applications the number of default features is likely to be in the range of 10–15, while the percentage match required is likely to be about 70%. Tests were performed for the 9 sets of values given in Table 1, with $n = 4$ and $N = 6$ in all cases. For each test, P was varied from 0–1 in steps of 0.05, with 10 randomly generated subsumption problems being solved at each data point, giving a total of 210 subsumption problems in each test. All the tests were performed on 300MHz Pentium machines, with Allegro CL 5.0 running under Linux, and in order to keep the CPU time required within reasonable limits a maximum of 1,000s was allowed for each problem.

Tests T1–T3 proved relatively easy for both FaCT and KRIS, with both systems able to solve any of the problems in less than 0.1s of CPU time. This is not particularly

[5] The number of terms is rounded down to the nearest integer.

Test	T1	T2	T3	T4	T5	T6	T7	T8	T9
ℓ	5	5	5	10	10	10	15	15	15
$V(\%)$	90	70	50	90	70	50	90	70	50

Table 1. Parametric values for tests

surprising as, even for T3, C_v will be a disjunction of only 10 conjuncts, each of which is of size 2. Tests T4 and T7 also proved relatively easy, with both systems able to solve any problem in less than 0.3s of CPU time. This is again due to the small size of the disjunctions, resulting in this case from the 90% approximation value.

For tests T5 and T6 the difference between FaCT and KRIS became more evident. For T5, FaCT is able to solve >90% of problems in less than 0.3s, while for T6 this increases to 0.4s. With KRIS, the time taken to solve a problem critically depends on whether C_v subsumes C_r (i.e., $C_r \sqcap \neg C_v$ is unsatisfiable) or not. For T5 most non-subsuming (satisfiable) problems are solved in less than 0.1s whereas subsuming problems take more than 3.5s, while for T6 these values are 0.1s and 21s respectively. KRIS's faster time for non-subsuming problems is due to the fact that, in most cases, a solution can quickly be found regardless of the search strategy; FaCT, on the other hand, still has the overhead of its more sophisticated search techniques, and in particular of the lexical analysis and naming of sub-concepts.

For tests T8 and T9, KRIS's difficulty with subsuming problems becomes critical and it proved unable to solve any such problem within the 1,000s of CPU time allowed. FaCT remained consistent with respect to both subsuming and non-subsuming problems, solving >90% of problems in less than 9s for T8 and less than 28s for T9, with FaCT's worst time in all tests being 31s. Figure 1 shows the 50th percentile (median) and 90th percentile[6] times for T9 with KRIS and FaCT plotted against the probability of generating subsuming concepts. Note that where the CPU time is shown as 1,000s no solution was found, and the time which would be required in order to find a solution could be $\gg$1,000s.

KRIS's poor performance is easily explained by the fact that for T8, C_v will be a disjunction of 3,003 conjuncts, each of which is of size 10. When C_v is negated in the subsumption test this becomes a conjunction of disjuncts which, using a naive strategy, leads to a search of 10^{3003} possible expansions (although only 2^{10} of these can be unique); for T9 C_v will be a disjunction of 6,435 conjuncts, each of which is of size 7.

5 Discussion and conclusions

Clearly the results using FaCT on the larger disjuncts (in tests T8 and T9) are encouraging compared to KRIS, indicating that frequent use of optimised disjunctive reasoning is potentially viable. To ascertain whether the very significant gains are sufficient to justify the proposed representation in real applications, some further questions should be considered: At what rate do individuals need to be categorised? Will one instance

[6] The 90th percentile is the maximum time taken to solve all but the hardest 10% of problems.

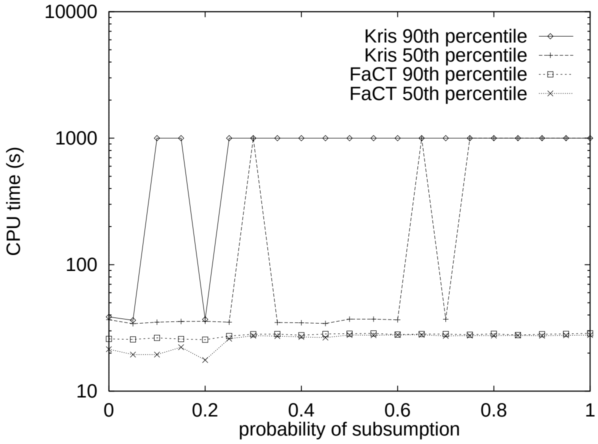

Fig. 1. Percentile times for T9 with KRIS and FaCT

inference, or recognition process, lead to further instance inferences? How many subsumption tests are needed for an instance recognition? How likely is it that the more difficult subsumption tests will occur?

An additional question also has to do with space complexity. The naive representation of the conceptual representation described results in exponential increase in space requirements. However we would expect to adapt existing techniques which only require keeping part of the concept hierarchy in memory, and to expand concepts to their full representation only at run-time. The exponential space increase will not result in exponential time increase using the described optimisations, due to the fact that most of the increase is in equivalent concepts which are pruned away.

The rate at which instance inference needs to be done can vary widely depending on the application. In a real-time telecommunications fault diagnostic system, individual descriptors needing to be classified as normal, or as a particular category of fault, may arrive at several per second. On the other hand a support system for medical diagnosis, being used by an individual doctor, could reasonably expect a descriptor of patient symptoms every 10 minutes. The rate for a bibliographic or travel KB, responding to user queries probably lies somewhere between these two. The experimental response times we have established are clearly adequate for some applications, but possibly inadequate for others.[7]

Applications with highly interrelated individuals can result in significant propagation when a single individual is modified. Consequently one recognition process can trigger several other such processes. Some applications (such as a bibliographic database or a travel information database) rely on a large set of individuals many of which may be interrelated. However, other applications (such as the medical diagnostic sup-

[7] Although if inadequate response times occur relatively infrequently it may be possible to achieve usability by supplementing the optimisation techniques with special purpose heuristics.

port described in [Padgham and Zhang, 1993], where individuals are descriptions of a set of patient symptoms) mostly deal with individuals which have no effect on other individuals and thus can only result in the subsumption tests necessary for a single recognition problem.

The number of subsumption tests required for a particular instance recognition task depends on both the number of concepts and the form of the hierarchy. Assuming that the hierarchy is close in form to a tree, and that individuals typically belong to only one sub-class at each level (at least until the bottom levels of the hierarchy are reached), then the number of subsumption tests needed at each level will be equal to the fan-out of the hierarchy at that level. Consequently, the total number of subsumption tests required will be roughly the average fan-out multiplied by the depth of the tree. Moreover, FaCT uses a caching optimisation to facilitate the quick discovery of non-subsumption, and this will typically work for all but one test at each level [Horrocks, 1997]. This effectively reduces the number of "full" subsumption tests to be equal to the depth of the tree.

The form of the hierarchy generated using the representation described in Section 2, with 3 descriptors per concept, obviously increases the number of nodes in the hierarchy by a factor of 3. It is also possible that the form of the hierarchy differs from concept hierarchies with which we are familiar, due to the various nuances of A *is-a* B which become available. For example in Figure 2 the hierarchy on the left represents the case where As are typically Bs, whereas the hierarchy on the right represents the case where As are always Bs. Further work is needed to determine the form of application taxonomies using this representation, but it is unlikely that the number of hard subsumption tests required per recognition task will change significantly: only the basic descriptors are highly disjunctive, and the caching optimisation should still allow "full" tests to be avoided in most cases. It is also likely that further optimisations can be developed, based on the particular representations we are using.

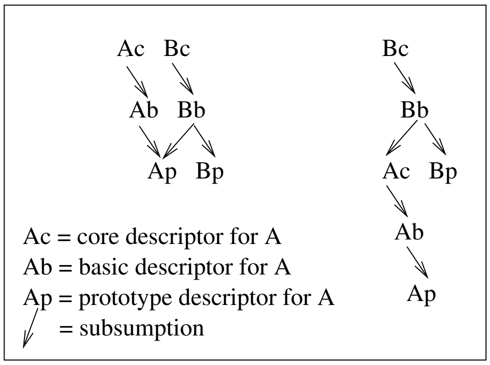

Fig. 2. Two nuances of A *is-a* B with 3 descriptors

The experimental subsumption problems generated were deliberately designed to be difficult, and it is unclear how often such problems would be encountered in a KB using the representation proposed (it is likely they would be more common than is usual for difficult subsumption problems in KBs not routinely using this representation). The best case would be that such difficult subsumption tests would be encountered only very occasionally, and never more than one per individual recognition process. Given that ontologies tend to be much broader than they are deep, typically with a depth in the range of 7 to 14, this would give (for the T9 situation) a response time which occasionally peaked at around 30s; the worst case would be that all "full" subsumptions for a given individual were difficult, giving a response time of 7 minutes for a typical hierarchy of depth 14. This may still be acceptable for an application such as that described in [Padgham and Zhang, 1993] where the system is being used as a diagnostic support tool for medicine.

To sum up, even making very pessimistic assumptions leads to a predicted worst-case response time of 7 minutes per recognition process. This is clearly within the range of useful response times for some applications. As a result of these explorations we are convinced that the recent optimisations make routine disjunctive reasoning feasible and thus justify using a representational approached based on disjunction. We are in the process of building a bibliographic KB application to further explore the representation of concepts as described and the associated computational properties.

References

[Baader and Hollunder, 1991] F. Baader and B. Hollunder. KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14, 1991.

[Baader and Hollunder, 1993] Franz Baader and Bernhard Hollunder. How to prefer more specific defaults in terminological default logic. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*, pages 669–674, 1993.

[Baader *et al.*, 1992] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 270–281, 1992.

[Baker, 1995] A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. PhD thesis, University of Oregon, 1995.

[Berman *et al.*, 1994] J. I. Berman, H. H. Moore IV, and J. R. Wright. CLASSIC and PROSE stories: Enabling technologies for knowledge based systems. *AT&T Technical Journal*, pages 69–78, January/February 1994.

[Coupey and Fouquere, 1997] P. Coupey and C. Fouquere. Extending conceptual definitions with default knowledge. *Computational Intelligence*, 13(2):258–299, 1997.

[Davis *et al.*, 1962] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[De Giacomo and Lenzerini, 1996] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327, 1996.

[Giunchiglia and Sebastiani, 1996] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 304–314, 1996.

[Heinsohn *et al.*, 1992] J. Heinsohn, D. Kudenko, B. Nebel, and H.J̃. Profitlich. An empirical analysis of terminological representation systems˙ In *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI-92)*, pages 676–773, 1992.

[Hollunder and Nutt, 1990] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. In *Proc. of the 9th European Conf. on Artificial Intelligence (ECAI-90)*, pages 348–353, 1990.

[Hollunder, 1994] B. Hollunder. *Algorithmic Foundations of Terminological Knowledge Representatin Systems*. PhD thesis, Universität des Saarlandes, 1994.

[Horrocks, 1997] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647, 1998.

[Hustadt and Schmidt, 1997] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97)*, volume 1, pages 202–207, 1997.

[Padgham and Nebel, 1993] Lin Padgham and Bernhard Nebel. Combining classification and non-monotonic inheritance reasoning: A first step. In *Proc. of the 7th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-93)*, pages 132–141, 1993. LNAI 689.

[Padgham and Zhang, 1993] Lin Padgham and Tingting Zhang. A terminological logic with defaults: A definition and an application. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*, volume 2, pages 662–668, 1993.

[Padgham, 1992] Lin Padgham. Defeasible inheritance: A lattice based approach. *Computers and Mathematics with Applications*, 23(6-9):527–541, 1992. Special Issue on Semantic Nets.

[Rosch, 1975] E. Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology*, 104, 1975.

[Smullyan, 1968] R. M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.