Using Resolution for Extending KL-ONE-type Languages

Tanel Tammet
Department of Computing Science
Chalmers University of Technology and
University of Göteborg
S-41296 Göteborg, Sweden
e-mail: tammet@cs.chalmers.se

Abstract

We argue that decidable classes of predicate logic combined with efficient decision methods are a suitable basis for developing powerful knowledge representation languages. The paper presents some methods for developing such languages based on decidable classes. Queries can be answered and databases can be completed by using efficient decision refinements of the resolution method. Several decidable classes usable as new knowledge representation languages are presented, containing some well-known ones, like the base language ALC of KL-ONE type systems, along with the resolution-based inference engines.

1 Introduction

We argue that the basic characteristic of deductive database languages (like DATALOG) and knowledge-representation languages (eg KL-ONE) is decidability. I.e, any query must terminate and give either a negative or a positive answer. This is in sharp contrast to using full first-order logic or non-decidable sublanguages like Horn clauses (the basis of PROLOG), where nonsuccessful queries do not, as a rule, terminate.

We consider certain decision methods developed in the context of automated theorem proving and show how these methods can be put to immediate use as an efficient inference engine (with the property that every query terminates). Some classes of predicate logic are suggested as languages for knowledge-representation systems. In particular, the ALC language used by KL-ONE systems is just a sublanguage of one the suggested classes. The no-circularities restriction of ALC is removed.

One of the advantages of the considered decision methods is their efficiency, clearly demonstrated in experiments (see [FLTZ 93]). The principal reason for this is that those methods are just further restrictions of a resolution method, which has been a preferred method for first-order theorem proving since its introduction. Roughly speaking, the decision methods at hand inherit the efficiency of resolution and then extend it further to the point of making a search space finite for decidable classes.

Another advantage of using resolution-based decision procedures is their usability for completing a database, i.e. deriving a finite set of inferred clauses D' from a database D such that when asking queries or adding new clauses to D in the future, no inferences from D or D' alone have to be considered. We could say that a dynamic use of D is replaced by the static clause set D'.

2 KL-ONE type languages

The knowledge representation system KL-ONE was introduced in [Brachmann, Schmolze 85] and a large number of systems based on the similar ideas have been built, for example BACK [Nebel, von Luck 88], CLASSIC [Borgida et al 89], KANDOR [Patel-Schneider 84], KL-TWO [Vilain 85], NIKL [Kaczmarek et al, 86].

KL-ONE type languages can be viewed as frame-based, database-oriented languages of terminological reasoning, where stress is put on the definition of concepts. Concepts can be viewed as unary predicates. Another common feature of these systems is the separation of the knowledge into terminological part (definitions of concepts), called "T-box" and the assertional part (database), called "A-box". There is a special concept language for defining concepts in the T-box, with a simply defined meaning in predicate calculus. The concept language can be viewed as a certain class of predicate logic.

Example given the concepts "person", "female" and "shy", the concept "persons who are female or not shy" can be expressed by the formula

```
(and person (or female (not shy)))
```

in concept logic, with the translation

$$(\forall x)(person(x) \,\&\, (female(x) \vee \neg shy(x)))$$

in predicate logic.

The user of the language can describe his concepts in the T-box, write down known facts in the A-box, and ask queries from the system, much like is done in PROLOG.

The crucial difference from PROLOG is that the language is decidable – any query must get either a positive or a negative answer. Several other important properties are also required to be decidable. For example, the question whether T-box is satisfiable, the question about derivability of formulae of the kind $(\forall x)(P(x) \Rightarrow Q(x))$ where P and Q are any concepts from T-box. The last question is called a "subsumption problem" in KL-ONE context: the problem can be read as "are all objects of type P also of type Q?". For example, given a definition of "female" as a "person" who is not "man", the concept "female" is subsumed by the concept "person". Several KL-ONE type systems have the capability to compute the full tree of concept subsumption, and this capability is considered to be rather important from the practical point of view.

The language called ALC which has been introduced in [Schmidt-Schauss, Smolka 88] can be considered as a basic language for KL-ONE systems, on which different extensions can be built. In [Schmidt-Schauss, Smolka 88] it is shown that satisfiability and KL-ONE-subsumption in ALC are PSPACE-complete.

ALC contains two kinds of predicates - concepts (unary predicates) and roles (binary predicates). ALC is meant for writing sets of concept definitions. Any formula in ALC has one of the following forms (we give a form in ALC and a well-known translation into predicate logic):

- `TRUE` - top concept in subsumption hierarchy. Translation: constant truth value "true".
- `FALSE` - bottom concept in subsumption hierarchy. Translation: constant truth value "false".
- F, where F is an arbitrary concept. Translation: $F(x)$, where x is defined by the context.
- (not F), where F is an arbitrary ALC-formula. Translation: $\neg F'$, where F' is the translation of F.
- (and F G), where F and G are arbitrary ALC-formulas. Translation: $(F' \,\&\, G')$, where F' and G' are translations of F and G.
- (or F G), where F and G are arbitrary ALC-formulas. Translation: $(F' \vee G')$, analogously to conjunction.
- (all R F), where R is a role, and F is an arbitrary ALC-formula. Translation: $(\forall y)(R(x,y) \Rightarrow F'[y])$, where the free variable x is determined by the context. $F'[y]$ cannot contain any free variables except y.
- (exists R F), where R is a role, and F is an arbitrary ALC- formula. Translation: $(\exists y)(R(x,y) \,\&\, F'[y])$, where the free variable x is determined by the context and $F'[y]$ cannot contain any free variables except y.

The concept definition in ALC has the following form: $P = F$, where F is an arbitrary ALC-formula. Translation: $(\forall x)(P(x) \Leftrightarrow F'[x])$.

An important additional restriction to the described language is that no circularities are allowed in concept definitions - that is, there must be a certain hierarchy of concepts such that all concepts in the right side of any concept definition are lower on the hierarchy than the defined concept. For example, it is not allowed to define a human as an animal whose mother is human: `human = (and animal (exists mother human))`, since this definition contains a circularity.

We will present a sample T-box given by F.Baader.

```
female = (not male)
person = (some sex (or male female))
parent = (and person (some child person))
mother = (and parent (some sex female))
father = (and parent (not mother))
grandparent = (and parent (some child parent))
parent_with_sons_only = (and parent (all child
(some sex male)))
```

The following is a translation of this T-box into predicate logic:

$(\forall x)(female(x) \Leftrightarrow \neg male(x))$
$(\forall x)(person(x) \Leftrightarrow (\exists y)(sex(x,y) \,\&\, (male(y) \vee female(y))))$
$(\forall x)(parent(x) \Leftrightarrow person(x) \& (\exists y)(child(x,y) \& person(y)))$
$(\forall x)(mother(x) \Leftrightarrow parent(x) \,\&\, (\exists y)(sex(x,y) \,\&\, female(y)))$
$(\forall x)(father(x) \Leftrightarrow parent(x) \,\&\, \neg mother(x))$
$(\forall x)(grandparent(x) \Leftrightarrow parent(x) \,\&\, (\exists y)(child(x,y) \&$ parent(y)))
$(\forall x)(parent\_with\_sons\_only(x) \Leftrightarrow parent(x) \&$
$\&(\forall y)(child(x,y) \Rightarrow (\exists z)(sex(y,z) \,\&\, male(z))))$

3 Using resolution to decide ALC

3.1 The resolution method

Since its introduction in [Robinson 65] the resolution method has been the most popular method of automated theorem proving in classical predicate logic. A large body of theory concerning various modifications and special strategies of resolution has been developed. First we will define the standard notions of the resolution. For further details see, for example, [Chang, Lee 73].

Definition *A* clause *is a finite set of literals. Any clause $\{L_1, L_2, \ldots, L_n\}$ is interpreted as the disjunction $L_1 \vee L_2 \vee \ldots \vee L_n$ of its member literals.*

Definition *The* clause form *of a formula F is a set of clauses $\{C_1, C_2, \ldots, C_m\}$ obtained from F by Skolemization (replacing quantified variables by new function terms) and equivalent transformations. Any clause set $\{C_1, C_2, \ldots, C_m\}$ is interpreted as a conjunction $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_m$ of its member clauses.*

Any formula of classical logic can be brought to the clausal form.

New clauses are derived by rules of binary resolution and factorization. Before each resolution step, all the variables in one of the resolved clauses have to be renamed, so that the two clauses will be variable disjoint.

Definition *If C and D are variable disjoint clauses, L_1 is a literal in C, L_2 is a literal in D such that $L_1^{\perp}$ and L_2 are unifiable by the most general unifier σ, then $E = (C - L_1)\sigma \cup (D - L_2)\sigma$ is a* resolvent *of C and D.*

The literals L_1 and L_2 are called the literals resolved upon.

Definition *A* factor *of a clause C is a clause $C\sigma$, where σ is the most general unifier of some $C' \subseteq C$.*

Definition *For a clause set S we define $Res\_Fac(S)$ as the set of resolvents of S and all factors of these resolvents. Additionally we then define:*

$$\mathcal{R}^0(S) = S,$$

$$\mathcal{R}^{i+1}(S) = \mathcal{R}^i(S) \cup Res\_Fac(\mathcal{R}^i(S)), \qquad and$$

$$\mathcal{R}^*(S) = \bigcup_i \mathcal{R}^i(S).$$

We say that a clause C is derivable from *a clause set S iff $C \in \mathcal{R}^*(S)$.*

The completeness theorem for resolution states that if and only if some set of clauses S is unsatisfiable (i.e. has no model), the resolution method will derive an empty clause (i.e. contradiction) from S. Any formula F of classical predicate calculus is true if and only if its negation $\neg F$ is unsatisfiable. The last fact can be proved by the resolution method: to prove that F is true, prove that the clausal form of $\neg F$ is unsatisfiable.

Presented with some unsatisfiable set of clauses S, the resolution method always (provided there is enough time and memory) terminates with success (derivation of an empty clause). However, as predicate calculus is undecidable, when presented with some satisfiable clause set G, the resolution method generally fails to stop and give a negative answer: instead, it will infinitely continue deriving new clauses.

Example Presented with the following *satisfiable* set of clauses the unrestricted resolution method will continue producing deeper and deeper clauses, without ever stopping:

$\{P(f(x)), P(x)\}$
$\{\neg P(f(x)), \neg P(x)\}$

The unrestricted resolution method or even any of the standard variants and strategies like like hyperresolution, set of support resolution, linear resolution and lock resolution are not usable as inference engines for predicate logic translations of KL-ONE type languages. These strategies generally fail to stop when presented with the underivable formula, thus they cannot be used for getting negative answers.

3.2 Decision refinements of the resolution method

Several papers (eg. [Maslov 68], [Zamov 72], [Joyner 76], [Zamov 89], [Leitsch 90], [Tammet 90], [Fermüller 91], [Tammet 92]) and a monography [FLTZ 93] deal with the problem of building special refinements of resolution for deciding formulas in decidable classes. The refinement of resolution decides some class of clauses C iff it always terminates on any clause set S in the class C, giving either a negative or a positive answer.

The main technique in the abovementioned papers for devising decision refinements of resolution is using special orderings on literals, prohibiting resolution upon some literals in a clause.

Quite obviously we are interested of refinements which are *complete* on the given class of formulas.

Definition *The resolution refinement is called* complete *for a class C iff the following holds: for any unsatisfiable set of clauses in C, an empty clause will be derived from the set by the resolution refinement.*

Consider the following ordering $\succ_k$ due to N.Zamov (see, for example, [Zamov 89]). You may also wish to consult [Tammet 92] and a monography [FLTZ 93].

Definition *The term t* covers *the term s iff at least one of the following conditions is satisfied:*

(i) $t = s$.

(ii) $t = f(t_1, \ldots, t_n)$, *s is a variable and* $s = t_i$ *for some* $i, 1 \leq i \leq n$.

(iii) $t = f(t_1, \ldots, t_n)$, $s = g(t_1, \ldots, t_n)$, $n \geq m \geq 0$.

Definition *The relation* $\succ'_k$:
$A \succ'_k B$ *iff for every argument s of the literal B there is an argument t of the literal A such that t covers s or s is a subterm of t.*

Definition *The ordering* $\succ_k$:
$A \succ_k B$ *iff* $A \succ'_k B$ *and* $B \not\succ'_k A$.

Definition *The* k-refinement *of resolution is binary resolution with the following restrictions:*

- *For any clause* $C = \{L_1, \ldots, L_n\}$ *some literal* L_i $(1 \leq i \leq n)$ *in C is allowed to be resolved upon iff there is no such literal* L_j $(1 \leq j \leq n)$ *in C that* $L_j \succ_k L_i$ *holds.*
- *For any clause* $C = \{L_1, \ldots, L_n\}$ *some literals* L_i *and* L_j $(1 \leq i, j \leq n)$ *in C are allowed to be factored upon iff the following holds:* σ *is the most general unifier of* L_i, L_j *and there is no literal* L_k $(1 \leq k \leq n)$ *in C such that* $L_k\sigma \succ_k L_i\sigma$.

For the purposes of our paper it is important that the k-refinement is a decision strategy for Gödel's Class: the class of formulas with no function symbols and the prenex form of the negated formula having a prefix of the following form: $\exists \ldots \exists\forall\forall\exists \ldots \exists$. The following class (Maslov's Class K) contains Gödel's Class.

Definition *The* prefix of the literal L *in a prenex predicate formula F is obtained from the prefix of F by discarding quantifiers for those variables which do not occur in L.*

Definition *Maslov's Class K:*
The prenex form of a negated predicate formula F is said to belong to the Class K if it does not contain function symbols and there are variables $x_1, \ldots, x_n$ *in a set of variables of F such that every* $x_i (1 \leq i \leq n)$ *is situated in the prefix of F to the left of all existential quantifiers (except the leading existential quantifiers, skolemized to constants) and for every prefix P of a literal in F at least one of the following holds:*

- *P consists of the single general quantifier* $\forall$.
- *P ends with the existential quantifier* $\exists$.
- *P is of the form* $(\forall x_1)(\forall x_2) \ldots (\forall x_n)$.

Maslov's Class K is introduced and shown to be solvable in [Maslov 68]. Notice that it contains the Gödel's Class, Monadic Class, Skolem's Class, to name a few.

Theorem 1 *The k-refinement of resolution is a decision algorithm for Maslov's Class K. That is, k-refinement is complete for this class and terminates on any formula in this class.*

Proof The proof is due to N.Zamov and can be found in [Zamov 89] and [FLTZ 93].

The k-refinement of resolution is compatible with full back subsumption, but forward subsumption and tautology elimination have to restricted in the following way (see [Tammet 92], [FLTZ 93]):

- a literal $P(\ldots x, \ldots y, \ldots)$ must not be allowed to forward-subsume a literal $P(\ldots x, \ldots x, \ldots)\{x := t\}$ for any t.

- a tautology $\{\ldots P(\ldots x, \ldots x, \ldots), \ldots, \neg P(\ldots x, \ldots x, \ldots)\}\{x := t\}$ should not be eliminated for any t. Eliminating other tautologies, like $\{P(x,y), \neg P(x,y)\}$ preserves completeness.

Example Presented with the following *satisfiable* set of clauses the k-refinement can resolve only upon the first literals of the clauses (since $P(f(x)) \succ_k P(x)$ and $\neg P(f(x)) \succ_k \neg P(x)$) and will produce a single clause $\{P(x), \neg P(x)\}$ which is a tautology and can be eliminated. As no more clauses can be generated, k-refinement stops with the answer that the given set is satisfiable:

$\{P(f(x)), P(x)\}$
$\{\neg P(f(x)), \neg P(x)\}$

3.3 Using decision refinements for the ALC-language

As the ALC-language allows arbitrary nesting of quantifiers, it does not immediately belong to any well-known decidable class. However, due to the fact that any subformula starting with the quantifier contains no more than one free variable, the following translation converts any ALC-formula (in predicate form) to a new formula, belonging to a decidable class close to Gödel's Class:

For any subformula $F[x]$ starting with the quantifier which is in the scope of two quantifiers and contains a single free variable x, introduce a new predicate symbol P, introduce the definition $(\forall x)(P(x) \Leftrightarrow F[x])$ and replace the subformula $F[x]$ with the atom $P(x)$.

As an example, we will present a translation of the last formula from our first sample T-box (the other formulas preserve unchanged):

$(\forall x)(parent\_with\_sons\_only(x) \Leftrightarrow$
$parent(x) \,\&\, (\forall y)(child(x,y) \Rightarrow P_1(y)))$
$(\forall y)(P_1(y) \Leftrightarrow (\exists z)(sex(y,z) \,\&\, male(z)))$

The following set *KL_ONE_example* is a clause form of the translated T-box above:

comment: definition of *female*:
$\{\neg female(x), \neg male(x)\}$
$\{male(x), female(x)\}$
comment: definition of *person*:
$\{\neg person(x), sex(x, f(x))\}$
$\{\neg person(x), male(f(x)), female(f(x))\}$
$\{\neg sex(x,y), \neg male(y), person(x)\}$
$\{\neg sex(x,y), \neg female(y), person(x)\}$
comment: definition of *parent*:
$\{\neg parent(x), person(x)\}$
$\{\neg parent(x), child(x, g(x))\}$
$\{\neg parent(x), person(g(x))\}$
$\{\neg person(x), \neg child(x,y), \neg person(y), parent(x)\}$
comment: definition of *mother*:
$\{\neg mother(x), parent(x)\}$
$\{\neg mother(x), sex(x, h(x))\}$
$\{\neg mother(x), female(h(x))\}$
$\{\neg parent(x), \neg sex(x,y), \neg female(y), mother(x)\}$
comment: definition of *father*:
$\{\neg father(x), parent(x)\}$
$\{\neg father(x), \neg mother(x)\}$
$\{\neg parent(x), mother(x), father(x)\}$
comment: definition of *grandparent*:
$\{\neg grandparent(x), parent(x)\}$
$\{\neg grandparent(x), child(x, k(x))\}$
$\{\neg granparent(x), parent(k(x))\}$
$\{\neg parent(x), \neg child(x,y), \neg parent(y), grandparent(x)\}$

comment: definition of *parent_with_sons_only* (abbreviated as *pwso*):
$\{\neg pwso(x), parent(x)\}$
$\{\neg pwso(x), \neg child(x,y), pl(y)\}$
$\{\neg parent(x), child(x, l(x)), pwso(x)\}$
$\{\neg parent(x), \neg pl(l(x)), pwso(x)\}$
$\{\neg pl(y), sex(y, r(y))\}$
$\{\neg pl(y), male(r(y))\}$
$\{\neg sex(y,z), \neg male(z), pl(y)\}$

Notice that the descriptive power of the T-box above is more limited than we'd expect. For example, given a following A-box

$$\{person(John), child(John, Peter),$$
$$sex(Peter, Male), male(Male)\}$$

it is not possible to deduce $pwso(John)$. In order to deduce $pwso(John)$, we'd need a fact $(\forall x)(child(John, x) \Rightarrow x = Peter)$. Unfortunately we do not have an equality predicate in the language!

4 Extending the ALC-class

We will consider the following natural extension "One-free" of the ALC-class:

Definition *Any formula F of the predicate logic without the equality predicate and function symbols belongs to the Class One-free iff any subformula of F starting with the quantifier contains no more than one free variable.*

We do not consider the constant symbols to be functional: thus we allow formulas in One-free to contain constant symbols.

Notice that the ALC-restriction of circularities does not hold for One-free Class.

The translation above is obviously sound for the One-free Class and guarantees that the translated formula will have a quantifier nesting maximally equal to two. Thus the translated formula can be seen as being "almost" in the Gödel's Class. It is not exactly in Gödel's Class, as the prenex form of the whole translated formula may have an arbitrarily deep quantifier nesting.

However, it is easy to see that each clause in the clausal form of the translated formula from the One-free Class has one of the two forms:

(i) A clause contains at most two variables and no function symbols.

(ii) A clause contains a single variable and at most one-place function symbols.

Termination of the k-refinement of resolution on this clausal class is easy to show. One has to modify the termination proof for Gödel's Class or the termination proof for the whole Maslov's Class K.

As for subsumption and tautology elimination, the same restrictions hold as for Maslov's Class K (see the section 2.2).

The following theorem shows that the completed form of an A-box contains explicitly all the clauses representing the mutual subsumption of concepts.

Theorem 2 *The clause set $\mathcal{R}^*_{\succ k}(F')$ for a translation F' of any One-free formula F contains all the clauses representing the mutual subsumption of concepts.*

Proof The proof is obtained by rewriting a k-refinement derivation. Certain inferences are moved downwards so that the rewritten derivation is still a k-refinement derivation.

As no new clauses can be derived from the set $\mathcal{R}^*_{\succ k}(F')$ with the k-refinement, $\mathcal{R}^*_{\succ k}(F')$ can be used as a completed form of F, making answering future questions about the A-box much easier than by using the original F alone.

Indeed, once we have computed the completed set $\mathcal{R}^*_{\succ k}(F')$ for some T-box (or the union of T-box and A-box) F', there is no need to attempt any derivations just from the clauses in F' or $\mathcal{R}^*_{\succ k}(F')$. Given some query Q, all the derivations will have some clause of Q as one of its ancestors.

5 Functional relations

Some extended versions of the ALC language (the one presented in [Hollunder, Nutt 90], for example) also allow to declare certain relations to be functional (eg, the relation "sex" in the example above would naturally be defined as a functional relation). The predicate logic equivalent of declaring a relation (say, R) to be functional on the first argument would be a following axiom:

$$(\forall xyz)((R(x,y) \;\&\; R(x,z)) \Rightarrow (y = z))$$

The axiom uses an equality predicate and thus we should also have to introduce an axiomatization for equality, that is, axioms for reflexivity, commutativity, transitivity and axioms for substitution into atoms. Including the full axiomatization for equality would mean that the resulting formulas do not fall into known decidable classes of predicate logic, thus making the translation useless for our purposes.

In principle functionality of a binary relation can be naturally expressed using one-place function symbols instead of two-place predicate symbols combined with the functionality axiom. This means that instead of using a quantifier to introduce a new functionally bound variable, we can use a term $f(x)$, where f is a function symbol of the functional relation and x depends on the context.

So, we would like to extend the One-free Class by allowing one-place function symbols. Unfortunately, this extension turns out to be undecidable, as it contains the ∀∃∀-Class.

We will consider ways of imposing additional restrictions to the extended One-free Class. One rather strong restriction, which we will shortly investigate in the following, is obtained by disallowing any non-functional relations. That is, we require that all binary relations were functional - then they can be represented by one-place function symbols and we can restrict the class to contain only monadic predicate symbols. It is natural to extend the resulting class to the full Class E.

Definition *Any formula F of the predicate logic without the equality predicate, but possibly containing function symbols belongs to the Class One-free-E (OFE, for short) iff any subformula of F starting with the quantifier contains no more than one free variable and all atoms contain no more than one variable.*

Obviously the translated form (translation above for the One-free Class) of any formula in the OFE Class belongs to the Class E:

Definition *Any formula F of the predicate logic without the equality predicate, but possibly containing function symbols belongs to the Class E iff no literal L in the Skolemized version of F contains more than one variable (which is allowed to have an arbitrary number of occurrences in L).*

There are several proofs of the decidability of Class E. The earliest proof seems to be given in [Gurevich 73].

We introduce an efficient resolution refinement deciding Class E:

Definition *Let A and B be literals. We say that $A \succ_v B$ iff the following is true:*

- *A contains some variable x and B contains the same variable x.*
- *the depth of the deepest occurrence of x in A is greater than the depth of the deepest occurrence of x in B.*

Definition *The* v-refinement *of resolution is binary resolution with the following restrictions:*

- *For any clause $C = \{L_1, \ldots, L_n\}$ some literal L_i $(1 \leq i \leq n)$ in C is allowed to be resolved upon iff there is no such literal L_j $(1 \leq j \leq n)$ in C that $L_j \succ_v L_i$ holds.*
- *For any clause $C = \{L_1, \ldots, L_n\}$ some literals L_i and L_j $(1 \leq i,j \leq n)$ in C are allowed to be factored upon iff the following holds: σ is the most general unifier of L_i, L_j and there is no literal L_k $(1 \leq k \leq n)$ in C such that $L_k\sigma \succ_v L_i\sigma$.*

Theorem 3 *The v-refinement of resolution is a decision algorithm for Class E. That is, v-refinement is complete for this class and terminates on any formula in this class.*

Proof The proof can be found in [Tammet 90] (termination only), [Tammet 92] and [FLTZ 93].

Again, v-refinement is compatible with full back subsumption, but tautology elimination cannot be used and forward subsumption has to be restricted by taking the ordering into consideration.

Example Consider the following refutable set of four clauses from E:

1: $\{P(f(a), f(x)), R(x, f(f(a)))\}$
2: $\{\neg R(a, f(x)), P(x,x)\}$
3: $\{R(a, f(x)), \neg P(x,x)\}$
4: $\{\neg P(f(a), f(x)), \neg R(x, f(f(a)))\}$

By $\succ_v$-ordering only the first literals in these four clauses are allowed to be resolved upon. At the first level we can derive two new clauses, both of them tautologies:

1,4 give 5: $\{R(x, f(f(a))), \neg R(x, f(f(a)))\}$
2,3 give 6: $\{P(x,x), \neg P(x,x)\}$

The clauses derived at the next level are subsumed by clauses 1-4. However, when subsumption is restricted by the ordering, these clauses are not subsumed and the refutation follows easily.

The OFE Class can be used both for the T-box and the A-box. As for the A-box, we also allow ground equality units such that the set of all equality units can be oriented to a complete (confluent and terminating) set of term rewriting rules. Thus all the facts of the form $R(a,b)$ for any functional R can be translated to equality units $f(a) = b$.

In order to decide any formula F in the OFE Class containing also ground equality units, do the following:

1. convert F to F' using the translation presented above for the One-free Class.
2. convert F' to a clause set C.
3. use Knuth-Bendix completion algorithm to complete the set. Notice that on the ground set of equality units the Knuth-Bendix completion algorithm will always terminate.
4. use narrowing to remove the rewrite rules resulting from the previous step: this gives a fully narrowed clause set C' in Class E. See the definition and completeness proofs of narrowing in [Slagle 74]. Alternatively, consult [Tammet 92] or [FLTZ 93].
5. use v-refinement of resolution to decide the clause set C'.

As an example we take the previous T-box, assume the relation "sex" to be declared functional and translate the selected part to OFE. Unfortunately, the OFE Class is too restrictive to translate the whole of the previous T-box in a "coherent" way. The reason here is that the relation *child* cannot be considered to be functional on the first argument. We will use the notion of "*first_child*" instead: *first_child* is assumed to be functional. Whenever some person x does not have children, we can define $first\_child(x) = \perp$, for some constant $\perp$ for which we know that $\neg person(\perp)$, say. In the following T-box we have replaced some equivalences by implications. The sole reason being that we felt the implications to correspond better to our intuitive understanding of the presented terminology.

$\neg person(\perp)$
$female(Female)$
$male(Male)$
$(\forall x)(female(x) \Rightarrow \neg male(x))$
$(\forall x)(person(x) \Rightarrow male(sex(x)) \vee female(sex(x)))$
$(\forall x)(parent(x) \Leftrightarrow person(x)\ \&\ person(first\_child(x)))$
$(\forall x)(mother(x) \Leftrightarrow parent(x)\ \&\ female(sex(x)))$
$(\forall x)(father(x) \Leftrightarrow parent(x)\ \&\ \neg mother(x))$

For the concepts *grandparent* and *parent_with_sons_only* we assume that the number of children a person can have is bounded by some N and use the following clumsy translation:

$(\forall x)(grandparent(x) \Leftrightarrow parent(x)\&$
$(parent(first\_child(x)) \vee \ldots \vee parent(N\_th\_child(x))))$
$(\forall x)(man(x) \Leftrightarrow (person(x)\ \&\ male(sex(x))))$
$(\forall x)(parent\_with\_sons\_only(x) \Leftrightarrow parent(x)\&$
$man(first\_child(x))\ \&\ \ldots\ \&\ man(N\_th\_child(x)))$

Now we can, for example, add a sample A-box to the given T-box (John, Peter and Male are constant symbols):

$first\_child(John) = Peter$
$second\_child(John) = \perp, \ldots, N\_th\_child(John) = \perp$
$sex(first\_child(John)) = Male$
$grandparent(John)$

Completion will add a new equality unit $sex(Peter) = Male$ to the ones above. Now it is easy to check out that from the full narrowing of the clause form of the last T-box above it is possible to derive $father(Peter)$, for example, without using the equality units any more. Following clauses from the fully narrowed set are used for this derivation:

$\{\neg person(\perp)\}$
$\{male(Male)\}$
$\{\neg female(x), \neg male(x)\}$
$\{\neg mother(Peter), female(Male)\}$
$\{\neg parent(x), mother(x), father(x)\}$
$\{\neg grandparent(John), parent(Peter), \ldots, parent(\perp)\}$
$\{\neg parent(x), person(x)\}$
$\{grandparent(John)\}$

6 Experiments

We have implemented various decision strategies of resolution in our new theorem prover written in a mixture of Scheme and C.

Recall the example clause set *KL_ONE_example* from the above section describing the KL-ONE type languages. Running on SUN Sparcstation ELC, our implementation of the k-refinement was able to show satisfiablity of the set and complete it (ie compute $\mathcal{R}^*_{\succ k}(F')$) in a fraction of the second. The completed set contains explicitly all the clauses representing the mutual KL-ONE-subsumption of the concepts. The whole derivation was 8 levels deep, 320 clauses were derived and 70 of those were retained.

An important point noticed during experimentation is that the decision refinement performs very well (when compared to any other known resolution strategy) also for the unsatisfiable formulas (eg queries with the positive answer).

References

[Borgida et al 89] Borgida, A., Brachmann, R.J., McGuinness, D.L., Resnick, L.A. CLASSIC: A Structural Data Model for Objects. Proceedings of the 4th National Conference of the AAAI, pp 34-37, Austin, Texas, 1984.

[Brachmann, Schmolze 85] Brachmann, R.J., Schmolze, J.G. An overview of the KL-ONE knowledge representation system. Cognitive Science 9(2), 1985, 171-216.

[Chang, Lee 73] Chang, C.L, Lee, R.C.T. Symbolic Logic and Mechanical Theorem Proving. (Academic Press, 1973).

[Fermüller 91] Fermüller, C. Deciding Classes of Clause Sets by Resolution. Ph.D. thesis, Technical University of Vienna 1991.

[FLTZ 93] Fermüller, C., Leitsch, A., Tammet, T., Zamov, N. Resolution methods for decision problems. LNAI (LNCS) 679, Springer Verlag, 1993.

[Gurevich 73] Gurevich, Y. Formuly s odnim ∀ (formulas with one ∀). In Izbrannye voprosy algebry i logiki (Selected Questions in Algebra and Logics). Nauka, Novosibirsk, 1973, pp. 97-110.

[Hollunder, Nutt 90] Hollunder, B. Nutt, W. Subsumption Algorithms for Concept Languages. Research report RR-90-04, DFKI GmbH, Kaiserslautern, Germany, 1990.

[Joyner 76] Joyner, W.H. Resolution strategies as decision procedures. J. ACM 23 (3)(1976), 396-417.

[Kaczmarek et al 86] Kaczmarek, T.S., Bates, R., Robins, G. Recent Developments in NIKL. Proceedings of the 5th National Conference of the AAAI, pp. 578-587, Philadelphia, Pa. 1986.

[Leitsch 90] Leitsch, A. Deciding Horn Classes by Hyperresolution. CSL'89, Springer LNCS 440, pp 225-241.

[Maslov 68] Maslov, S.Ju. The inverse method for establishing deducibility for logical calculi. Trudy Mat. Inst. Steklov 98 (1968) 26-87= Proc. Steklov. Inst. Math. 98 (1968) 25-96, MR 40 #5416; 43 #4620.

[Nebel, von Luck 88] Nebel, B., von Luck, K. Hybrid reasoning in BACK. In Z.W. Ras, L. Saitta (editors): Methodologies for Intelligent Systems, pp. 260-269, North Holland, Amsterdam, Netherlands, 1988.

[Patel-Schneider 84] Patel-Schneider, P. Small can be beautiful in knowledge representation. Proceedings of the IEEE Workshop on Principles of Knowledge-based Systems, pp. 11-16, Denver, Colorado, 1984.

[Robinson 65] Robinson, J.A. A Machine-oriented Logic Based on the Resolution Principle. Journal of the ACM 12, 1965, pp 23-41.

[Schmidt-Schauss, Smolka 88] Schmidt-Schauss, M., Smolka, G. Attributive Concept Descriptions with Unions and Complements. SEKI report SR-88-21, FB Informatik, Universität Kaiserslautern, Germany, 1988.

[Slagle 74] Slagle, J.R. Automated theorem proving for theories with simplifiers, commutativity and associativity. J. ACM 21 (4) (1974), 622-642.

[Tammet 90] Tammet, T. The resolution program, able to decide some solvable classes. COLOG-88, Springer LNCS 417, 1990, pp 300-312.

[Tammet 91] Tammet, T. Using Resolution for Deciding Solvable Classes and Building Finite Models. Baltic Computer Science, LNCS 502, 33-64, Springer Verlag 1991.

[Tammet 92] Tammet, T. Resolution methods for decision problems and finite-model building. Ph.D thesis. Department of Computer Science, Chalmers University of Technology / University of Göteborg, 1992.

[Vilain 85] Vilain, M.B. The restricted language architecture of a hybrid representation system. In R.J. Bachmann, H.J. Levesque, R. Reiter (editors): Proceedings of the 9th IJCAI, pp. 547-551, Los Angeles, California, 1985.

[Zamov 72] Zamov, N.K., On a bound for the complexity of terms in the resolution method. Trudy Mat. Inst. Steklov 128 (1972), pp 5-13.

[Zamov 89] Zamov, N.K., Maslov's inverse method and decidable classes. Annals of Pure and Applied Logic 42 (1989), 165-194.