Issues of Decidability for Description Logics in the Framework of Resolution

Ullrich Hustadt and Renate A. Schmidt

Department of Computing, Manchester Metropolitan University,
Chester Street, Manchester M1 5GD, United Kingdom
{U.Hustadt, R.A.Schmidt}@doc.mmu.ac.uk

Abstract. We describe two methods on the basis of which efficient resolution decision procedures can be developed for a range of description logics. The first method uses an ordering restriction and applies to the description logic $\mathcal{ALCB}$, which extends $\mathcal{ALC}$ with the top role, full role negation, role intersection, role disjunction, role converse, domain restriction, range restriction, and role hierarchies. The second method is based solely on a selection restriction and applies to reducts of $\mathcal{ALCB}$ without the top role and role negation. The latter method can be viewed as a polynomial simulation of familiar tableaux-based decision procedures. It can also be employed for automated model generation.

1 Introduction

Since the work of Kallick [13] resolution-based decision procedures for subclasses of first-order logic have drawn continuous attention [5,7,12]. There are two research areas where decidability issues also play a prominent role: extended modal logics and description logics [6,9,14]. Although is is not difficult to see that most of the logics under consideration can be translated into first-order logic, the exact relation to decidable subclasses of first-order logic and in particular to subclasses decidable by resolution is still under investigation. A recent important result describes a resolution decision procedure for the guarded fragment using a non-liftable ordering refinement [5]. But, the restrictions on the polarity of guards in guarded formulae are too strong to capture description logics with role negation (correspondingly, extended modal logics with relational negation).

Description logics with role negation can be embedded into the class One-Free, for which a resolution decision procedure using a non-liftable ordering refinement exists [7,19]. However, this method cannot be extended easily to description logics with transitive roles. The method of this paper is based on the resolution framework of Bachmair and Ganzinger [4] which is also suitable for overcoming the problems associated with transitivity axioms, in particular, non-termination of resolution on the relational translation of certain transitive modal logics [3,8].

The most prominent description logic is $\mathcal{ALC}$ [18]. It can be embedded into a subclass of the Bernays-Schönfinkel class, called basic path logic, which can

be decided by resolution and condensing using any compatible ordering or selection strategy [17]. This embedding was used in recent experimental work which provides evidence that resolution theorem provers can serve as reasonable and efficient inference tools for description logics [11,15].

In this paper we consider an expressive description logic, which we believe has not been considered before in the literature on description logics or modal logics. We call the logic $\mathcal{ALB}$ which is short for 'attribute language with boolean algebras on concepts and roles'. $\mathcal{ALB}$ extends $\mathcal{ALC}$ with the top role, full role negation, role intersection, role disjunction, role converse, domain restriction, range restriction, and role hierarchies.

We describe two methods on the basis of which efficient resolution decision procedures can be developed for a number of description logics. The first method uses an ordering restriction and applies to $\mathcal{ALB}$ and its reducts. The second method is based solely on a selection restriction and applies to reducts of $\mathcal{ALB}$ without the top role and role negation. On $\mathcal{ALC}$, the latter method can be viewed as a polynomial simulation of tableaux-based theorem proving. This result is a first contribution towards a better understanding of the relationship of tableaux-based and resolution-based reasoning for description logics, similar to our understanding of the relationship of various calculi for propositional logic [20].

This paper is organised as follows. In Section 2 we introduce the description logic $\mathcal{ALB}$ and give an embedding into first-order logic. The two resolution decision procedures described in this paper are instances of a general resolution calculus outlined in Section 3. Section 4 describes a decision procedure for the satisfiability of general $\mathcal{ALB}$ knowledge bases which is based on an ordering refinement of resolution. Section 5 describes a decision procedure for the satisfiability of descriptive knowledge bases over a reduct of $\mathcal{ALB}$ based on a selection refinement. We show in Section 6 that with this decision procedure we can polynomially simulate standard tableaux decision procedures for $\mathcal{ALC}$. In Section 7 we turn our attention to automated model generation. The Conclusion mentions related work.

2 Syntax and semantics of $\mathcal{ALB}$

We start with the definition of the description logic $\mathcal{ALB}$. The signature is given by a tuple $\Sigma = (\mathsf{O}, \mathsf{C}, \mathsf{R})$ of three disjoint alphabets, the set C of *concept symbols*, the set R of *role symbols*, and the set O of *object symbols*. *Concept terms* (or just *concepts*) and *role terms* (or just *roles*) are defined as follows. Every concept symbol is a concept and every role symbol is a role. If C and D are concepts, and R and S are roles, then $\top$, $\bot$, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, and $\exists R.C$ are *concept terms*, and $\bigtriangledown$, $\bigtriangleup$, $R \sqcap S$, $R \sqcup S$, $\neg R$, R^{-1} (*converse*), $R\restriction C$ (*domain restriction*), and $R\downharpoonright C$ (*range restriction*) are roles. Concept and role symbols are also called *atomic* terms.

A *knowledge base* has two parts: A *TBox* comprising of terminological sentences and an *ABox* comprising of assertional sentences. *Terminological sen-*

Embedding of sentences:

$$\begin{aligned}
\Pi(C \dot{\sqsubseteq} D) &= \forall x{:}\, \pi(C,x) \rightarrow \pi(D,x) & \Pi(R \dot{\sqsubseteq} S) &= \forall x,y{:}\, \pi(R,x,y) \rightarrow \pi(S,x,y)\\
\Pi(C \doteq D) &= \forall x{:}\, \pi(C,x) \leftrightarrow \pi(D,x) & \Pi(R \doteq S) &= \forall x,y{:}\, \pi(R,x,y) \leftrightarrow \pi(S,x,y)\\
\Pi(a \in C) &= \pi(C,\underline{a}) & \Pi((a,b) \in R) &= \pi(R,\underline{a},\underline{b})
\end{aligned}$$

where $\underline{a}$ and $\underline{b}$ are constants uniquely associated with a and b.
Embedding of terms:

$$\begin{aligned}
\pi(A,X) &= p_A(X) & \pi(P,X,Y) &= p_P(X,Y)\\
\pi(\neg C,X) &= \neg\pi(C,X) & \pi(\neg R,X,Y) &= \neg\pi(R,X,Y)\\
\pi(\top,X) &= \top & \pi(\nabla,X,Y) &= \top\\
\pi(\bot,X) &= \bot & \pi(\triangle,X,Y) &= \bot\\
\pi(C \sqcap D,X) &= \pi(C,X) \wedge \pi(D,X) & \pi(R \sqcap S,X,Y) &= \pi(R,X,Y) \wedge \pi(S,X,Y)\\
\pi(C \sqcup D,X) &= \pi(C,X) \vee \pi(D,X) & \pi(R \sqcup S,X,Y) &= \pi(R,X,Y) \vee \pi(S,X,Y)\\
\pi(\forall R.C,X) &= \forall y{:}\, \pi(R,X,y) \rightarrow \pi(C,y) & \pi(R\lfloor C,X,Y) &= \pi(R,X,Y) \wedge \pi(C,Y)\\
\pi(\exists R.C,X) &= \exists y{:}\, \pi(R,X,y) \wedge \pi(C,y) & \pi(R\uparrow C,X,Y) &= \pi(R,X,Y) \wedge \pi(C,X)\\
& & \pi(R^{-1},X,Y) &= \pi(R,Y,X)
\end{aligned}$$

where X and Y are meta-variables for variables and constants, and p_A (respectively p_P) denotes a unary (binary) predicate symbol uniquely associated with the concept symbol A (role symbol P). The variable y is distinct from X.

Fig. 1. The standard embedding of $\mathcal{ALB}$ into first-order logic

tences are of the form $C \dot{\sqsubseteq} D$, $C \doteq D$, $R \dot{\sqsubseteq} S$, and $R \doteq S$, and *assertional sentences* are of the form $a \in C$ and $(a,b) \in R$, where C and D are concepts, R and S are roles, and a and b are object symbols.

A symbol S_0 *uses* a symbol S_1 *directly* in a TBox T if and only if T contains a sentence of the form $S_0 \doteq E$ or $S_0 \dot{\sqsubseteq} E$ such that S_1 occurs in E. A symbol S_0 *uses* S_n if and only if there is a chain of symbols $S_0, \ldots, S_n$ such that S_i uses S_{i+1} directly, for every i, $1 \leq i \leq n-1$. A knowledge base Γ is said to contain a *terminological cycle* if and only if some symbol uses itself in the TBox of Γ.

The standard definition of knowledge bases imposes the following restrictions on the set of admissible terminological sentences: (i) The left-hand sides of terminological sentences have to be concept symbols or role symbols, (ii) any concept or role symbol occurs at most once on the left-hand side of any terminological sentence, and (iii) there are no terminological cycles. Knowledge bases obeying these restrictions are known as *descriptive knowledge bases*. In this context terminological sentences are called *definitions*. When no restrictions are imposed, we speak of *general knowledge bases*.

The semantics of $\mathcal{ALB}$ is specified by the embedding into first-order logic defined in Figure 1.

All common inferential services for knowledge bases, like subsumption tests for concepts, TBox classification, realization, retrieval, can be reduced to tests of the satisfiability of a knowledge base.

3 The resolution framework

We adopt the resolution framework of Bachmair and Ganzinger [4], from which we recall some basic definitions.

As usual clauses are assumed to be multisets of literals. The components in the variable partition of a clause are called split components, that is, split components do not share variables. A clause which is identical to its split component is indecomposable. The condensation $\mathrm{Cond}(C)$ of a clause C is a minimal subclause of C which is a factor of C.

In the framework inference rules are parameterised by an ordering $\succ$ and a selection function S. A well-founded and total ordering on ground literals is called *admissible* if it is compatible with a certain *complexity measure* c_L on ground literals L. If $c_L \succ c_{L'}$ implies $L \succ L'$ for any two ground literals L and L', then the ordering is said to be *compatible*. For the exact requirements orderings have to satisfy refer to [4]. A *selection* function assigns to each clause a possibly empty set of occurrences of negative literals. If C is a clause, then the literal occurrences in $S(C)$ are *selected*. No restrictions are imposed on the selection function.

The calculus consists of general *expansion rules* of the form

$$\frac{N}{N_1 \mid \cdots \mid N_n},$$

each representing a finite derivation of the leaves $N_1, \ldots, N_k$ from the root N. The following rules describe how derivations can be expanded at leaves.

Deduce:
$$\frac{N}{N \cup \{\mathrm{Cond}(C)\}}$$
if C is either a resolvent or a factor of clauses in N.

Delete:
$$\frac{N \cup \{C\}}{N}$$
if C is a tautology or N contains a clause which is a variant of C.

Split:
$$\frac{N \cup \{C \cup D\}}{N \cup \{C\} \mid N \cup \{D\}}$$
if C and D are variable-disjoint.

Resolvents and factors are derived by the following rules.

Ordered Resolution:
$$\frac{C \cup \{A_1\} \quad D \cup \{\neg A_2\}}{(C \cup D)\sigma}$$
where (i) σ is the most general unifier of A_1 and A_2, (ii) no literal is selected in C and $A_1\sigma$ is strictly $\succ$-maximal with respect to $C\sigma$, and (iii) $\neg A_2$ is either selected, or $\neg A_2\sigma$ is maximal with respect to $D\sigma$ and no literal is selected in D. $C \vee A$ is called the *positive premise* and $D \vee \neg B$ the *negative premise*.[1]

Ordered Factoring:
$$\frac{C \cup \{A_1, A_2\}}{(C \cup \{A_1\})\sigma}$$
where (i) σ is the most general unifier of A_1 and A_2; and (ii) no literal is selected in C and $A_1\sigma$ is $\succ$-maximal with respect to $C\sigma$.

We will restrict our attention to derivations which are generated by strategies in which "Delete", "Split", and "Deduce" are applied in this order. In addition, no application of the "Deduce" expansion rule with identical premises and identical consequence may occur twice on the same path in the derivation.

4 Decidability by ordered resolution

The conversion to clausal form of first-order formulae resulting from the translation of $\mathcal{ALB}$ knowledge bases, makes use of a particular form of structural transformation (or renaming) [2,16]. For ease of presentation we assume any first-order formula ϕ is in negation normal form.

Let $\mathrm{Pos}(\phi)$ be the set of positions of a formula ϕ. If λ is a position in ϕ, then $\phi|_\lambda$ denotes the subformula of ϕ at position λ and $\phi[\lambda \leftarrow \psi]$ is the result of replacing ϕ at position λ by ψ. We associate with each element λ of $\Lambda \subseteq \mathrm{Pos}(\phi)$ a predicate symbol Q_λ and a literal $Q_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the free variables of $\phi|_\lambda$, Q_λ does not occur in ϕ and two symbols Q_λ and $Q_{\lambda'}$ are equal only if ϕ_λ and $\phi_{\lambda'}$ are variant formulae. The *definition* of Q_λ is the formula

$$\mathrm{Def}_\lambda(\phi) = \forall\, x_1, \ldots, x_n{:}(Q_\lambda(x_1, \ldots, x_n) \leftrightarrow \phi|_\lambda).^2$$

Define $\mathrm{Def}_\Lambda(\phi)$ inductively by: $\mathrm{Def}_\emptyset(\phi) = \phi$ and

$$\mathrm{Def}_{\Lambda\cup\{\lambda\}}(\phi) = \mathrm{Def}_\Lambda(\phi[\lambda \leftarrow Q_\lambda(x_1, \ldots, x_n)]) \wedge \mathrm{Def}_\lambda(\phi)$$

where λ is maximal in $\Lambda\cup\{\lambda\}$ with respect to the prefix ordering on positions. Let $\mathrm{Pos}_r(\phi)$ be the set of positions of non-atomic subformulae of ϕ with at least one free variable. By Ξ we denote the transformation taking $\Pi(\Gamma)$ to its *definitional form* $\mathrm{Def}_{\mathrm{Pos}_r(\Pi(\Gamma))}(\Pi(\Gamma))$.

Theorem 1. *Let Γ be any knowledge base. $\Xi\Pi(\Gamma)$ can be computed in polynomial time, and Γ is satisfiable iff $\Xi\Pi(\Gamma)$ is satisfiable.*

Next we characterise a class of clauses which we call *DL-clauses*. Let C be a clause and L a literal in C. We refer to a literal L as *embracing* in C if for every L' in C, $\mathcal{V}(L') \cap \mathcal{V}(L) \neq \emptyset$[3] implies $\mathcal{V}(L') \subseteq \mathcal{V}(L)$ (that is, it contains all variables occurring in the split component in which it occurs). A term t in C is called *embracing* if for every L' in C, $\mathcal{V}(L') \cap \mathcal{V}(t) \neq \emptyset$ implies $\mathcal{V}(L) \subseteq \mathcal{V}(t)$. A term is called *compound* if it is neither a variable nor a constant. A literal L is *singular* if it contains no compound term and $\mathcal{V}(L)$ is a singleton. A literal is *flat* if it is non-ground and contains no compound term.

In the context of this paper a *regular* literal has either no compound term arguments or if it does then there is a compound term argument which contains

[1] As usual we implicitly assume that the premises have no common variables.

[2] As any formula ϕ is assumed to be in negation normal form, we could of course also use $\mathrm{Def}^+_\lambda(\phi) = \forall\, x_1, \ldots, x_n : (Q_\lambda(x_1, \ldots, x_n) \rightarrow \phi|_\lambda)$ without altering the validity of our results.

[3] $\mathcal{V}(L)$ denotes the set of variables of L.

all the variables of the literal and does not itself have a compound term argument. By definition, L is a *DL-literal* if the following is true.

1. L is regular,
2. L is either monadic or dyadic, and contains at most 2 variables,
3. L is ground whenever it contains a constant symbol, and
4. the maximal arity of any function symbol in L is 1.

A clause C is a *DL-clause*, if

1. when C contains a compound term t, then t is embracing,
2. C is ground whenever C contains a constant symbol,
3. all literals in C are DL-literals, and
4. the argument multisets of all flat, dyadic literals coincide.

Property (4) is important to enable us to employ an ordering which is stable under substitutions. It excludes clauses like $\{p(x,x), q(x,y)\}$. In order to avoid possibly unbounded chains of variables across literals we would need to restrict resolution inferences to the literal $q(x,y)$. But, when such clauses are present it is in general not possible to devise an ordering stable under substitution such that only the second literal is maximal. By contrast, clauses like $\{p(x,x), q(x,x)\}$ and $\{p(x,y), q(x,y)\}$ are DL-clauses, and may occur in a derivation from $\Xi\Pi(\Gamma)$.

Lemma 2. *Let Γ be a knowledge base. Every clause in the clausal form of $\Xi\Pi(\Gamma)$ belongs to the class of DL-clauses.*

For every ground literal L, let the complexity measure c_L be the multiset of arguments of L. We compare complexity measures by the multiset extension of the strict subterm ordering $\succ^s_{mul}$. The ordering is lifted from ground to non-ground expressions as follows: $E \succ E'$ if and only if $E\sigma \succ E'\sigma$, for all ground instances $E\sigma$ and $E'\sigma$. We show that ordered resolution and ordered factoring on DL-clauses with respect to any ordering $\succ_{ccv}$ which is compatible with this complexity measure will result in DL-clauses.

Lemma 3. *Let $C = \{L_1, L_2\} \cup D$ be an indecomposable, DL-clause with σ a most general unifier of L_1 and L_2 such that $L_1\sigma$ is $\succ_{ccv}$-maximal with respect to $D\sigma$. The split components of $(\{L_1\} \cup D)\sigma$ are DL-clauses.*

Proof. Since C is indecomposable and contains at least two literals, it is not a ground clause. By property (2) it contains no constants. So, the most general unifier σ will not introduce any constant symbols. Also, there are no compound terms in the codomain of σ, since all compound terms in C are embracing. Therefore, σ is a variable renaming. It is straightforward to see that variable renamings preserve the properties (1)–(4). □

Lemma 4. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be two variable-disjoint, indecomposable, DL-clauses such that A_1 and A_2 are unifiable with most general unifier σ, and $A_1\sigma$ and $A_2\sigma$ are $\succ_{ccv}$-maximal with respect to $D_1\sigma$ and $D_2\sigma$, respectively. The split components of $(D_1 \cup D_2)\sigma$ are DL-clauses.*

Proof. It is not difficult to show that the following holds:

1. A_1 is embracing in C_1.

2. If A_1 is a flat, singular literal, then C_1 contains only flat, singular literals.
3. If A_1 is flat, then C_1 contains no compound term.
4. If $A_1\sigma$ contains a compound term t, then for no variable x in C_2 is $x\sigma$ a compound term, and t is embracing in $(D_1 \cup D_2)\sigma$.
5. If A_1 contains a compound term t, then for no variable x in C_1, is $x\sigma$ a compound term and t is embracing in $(D_1 \cup D_2)\sigma$.

Analogously, for $\neg A_2$ and C_2. Let E be a split component of $(D_1 \cup D_2)\sigma$.

1. It follows that no compound term in E has a compound term argument and that compound terms in E are embracing.
2. Suppose $A_1\sigma$ contains a constant. Then either A_1 or A_2 is ground. Since A_1 and $\neg A_2$ are embracing, E is a unit ground clause.
3. It follows from (1) and (2) that all literals in E are DL-literals.
4. Only if $A_1\sigma = A_2\sigma$ is a flat, dyadic literal does E possibly contain a flat, dyadic literal $L\sigma$. The argument multiset of L coincide with that of either A_1 or A_2. Therefore, the argument multiset of $L\sigma$ coincides with $A_1\sigma$. So, does the argument multiset of any flat, dyadic literal in E. □

Theorem 5. *Let Γ be a knowledge base of $\mathcal{ALB}$ and let N be the clausal form of $\Xi\Pi(\Gamma)$. Then any derivation from N by ordered resolution and ordered factoring based on $\succ_{ccv}$ terminates.*

Proof. By Theorem 1 and Lemmas 2, 3 and 4, because any class of non-variant, condensed, indecomposable DL-clauses built from finitely many predicate and function symbols is finitely bounded; and the fact that any application of "Deduce" will be followed immediately by applications of the "Split" rule, as well as the fact that "Delete" is applied eagerly. □

The techniques used by Tammet and Zamov to decide the classes One-Free [19] and KS [7] can also be utilised to provide a decision procedure for the class of DL-clauses. However, these techniques are based on non-liftable orderings which have limitations regarding the application of some standard simplification rules.

5 Decidability by selection

Most decision procedures presented in the literature for the satisfiability problem of concepts and knowledge bases in description logics are specialised tableaux-based transformation systems. Since derivations in these system are quite different from those of a resolution decision procedure based on an ordering refinement, very little can be said about the relative proof or search complexity of these systems. In this section we define a resolution decision procedure based solely on the use of a particular selection function for a reduct of $\mathcal{ALB}$ including $\mathcal{ALC}$. In the next section we will establish that this procedure is able to polynomially simulate standard tableaux-based transformation systems for $\mathcal{ALC}$, thereby establishing a result relating resolution and tableaux decision procedures.

We focus on descriptive knowledge bases Γ over reducts of $\mathcal{ALB}$ without role negation and the top role. Determining the satisfiability of Γ using a tableaux decision procedure is a two-step process. For a non-empty TBox, concept and role symbols in the ABox are *unfolded* in the first step, that is, replaced by the right-hand side of their definitions, before the tableaux-based transformation system is applied to the ABox, in the next step.

To simulate the unfolding steps in our resolution decision procedure, we have to restrict resolution inferences with clauses stemming from the translation of terminological sentences of the form $S \doteq E$ and $S \dot{\sqsubseteq} E$ to the literals associated with S. As only negative literals can be selected, it is necessary to transform the given knowledge base. Essentially, for concepts, occurrences of $\neg A$ will be replaced by a new symbol $\overline{A}$, and for roles, positive occurrences[4] of P will be replaced by a new symbol P^d while negative occurrences will be replaced by P^u.

Without loss of generality, all expressions occurring in Γ are assumed to be in negation normal form. Formally, let $\mathrm{D}_{\doteq}(\Gamma)$ denote the set of symbols $S_0 \in \mathsf{C} \cup \mathsf{R}$ such that Γ contains a terminological sentence $S_0 \doteq E$. We obtain the transformed knowledge base $\overline{\Gamma}$, defined over $(\mathsf{O}, \overline{\mathsf{C}}, \overline{\mathsf{R}})$, in the following way. Extend C to $\overline{\mathsf{C}}$, by adding a concept symbol $\overline{A}$ for every concept symbol A in $\mathrm{D}_{\doteq}(\Gamma)$. Replace R by $\overline{\mathsf{R}}$, which is obtained by replacing every role symbol $P \in \mathrm{D}_{\doteq}(\Gamma)$ by new symbols P^u and P^d. The following steps transform Γ to $\overline{\Gamma}$.

1. Replace concept definitions $A \doteq C$, by $A \dot{\sqsubseteq} C$ and $\neg A \dot{\sqsubseteq} \mathrm{nnf}(\neg C)$, and replace role definitions $P \doteq R$, by $P^d \dot{\sqsubseteq} R$ and $R \dot{\sqsubseteq} P^u$, where $\mathrm{nnf}(\neg C)$ is the negation normal form of $\neg C$.
2. Replace every occurrence of a concept $\neg A$, for A in $\mathrm{D}_{\doteq}(\Gamma)$, by $\overline{A}$.
3. Replace every positive occurrence of a role symbol $P \in \mathrm{D}_{\doteq}(\Gamma)$ by P^d, and every negative occurrence of P by P^u.
4. For every concept symbol A in $\mathrm{D}_{\doteq}(\Gamma)$, add the sentence $A \dot{\sqsubseteq} \neg\overline{A}$ and add for every role symbol P in $\mathrm{D}_{\doteq}(\Gamma)$, the sentence $P^d \dot{\sqsubseteq} P^u$.

For example, if Γ contains the terminological axioms $A \doteq B \sqcup C$, $B \dot{\sqsubseteq} \forall P.C$, and $P \doteq R \sqcap S$ where A, B, and C are concept symbols and P, R, and S are role symbols, then the transformed knowledge base contains:

$$
\begin{array}{ll}
A \dot{\sqsubseteq} B \sqcup C & P^d \dot{\sqsubseteq} R \sqcap S \\
\overline{A} \dot{\sqsubseteq} \neg B \sqcap \neg C & R \sqcap S \dot{\sqsubseteq} P^u \\
A \dot{\sqsubseteq} \neg\overline{A} & P^d \dot{\sqsubseteq} P^u \\
B \dot{\sqsubseteq} \forall P^u.C &
\end{array}
$$

The reason for transforming concept definitions differently from role definitions is that no negative information about roles in the form of ground unit clauses can be derived.

[4] An occurrence of a subformula (subexpression) is a *positive occurrence* if it is one inside the scope of an even number of (explicit or implicit) negations, and an occurrence is a *negative occurrence* if it is one inside the scope of an odd number of negations.

In this section, the definitional form is produced by a variant $\overline{\Xi}$ of the transformation Ξ described in the previous section. First, $\overline{\Xi}$ uses *definitions*, $\mathrm{Def}_\lambda(\phi)$, of the form

$$\forall\, x_1, \ldots, x_n \colon (Q_\lambda(x_1, \ldots, x_n) \to \phi|_\lambda) \text{ or } \forall\, x_1, \ldots, x_n \colon (\phi|_\lambda \to Q_\lambda(x_1, \ldots, x_n)),$$

depending on whether $\phi|_\lambda$ occurs positively or negatively in ϕ. Second, only subformulae χ associated with non-atomic terms in $\overline{\Gamma}$, with the exception of non-atomic terms in terminological sentences introduced by step 4 of the transformation, are renamed. In particular, this means for subformulae χ of the form $\forall\, y \colon (\phi(x,y) \to \psi(y))$ or $\exists\, y \colon (\phi(x,y) \wedge \psi(y))$, $\overline{\Xi}$ will only introduce definitions for χ itself and $\phi(x,y)$ and $\psi(y)$ if necessary, but not for $\neg\phi(x,y) \vee \psi(y)$ and $\phi(x,y) \wedge \psi(y)$. Third, two symbols Q_λ and $Q_{\lambda'}$ are equal iff ϕ_λ and $\phi_{\lambda'}$ are variant formulae. Thus, every newly introduced symbol Q_λ is associated with an expression E of the original language, and hence, we will denote Q_λ by p_E.

For the sample knowledge base above we obtain the following set of clauses.

$$\begin{array}{ll}
\{\neg p_A(x), p_B(x), p_C(x)\} & \{\neg p^d_P(x,y), p_R(x,y)\} \\
\{\neg p_{\overline{A}}(x), \neg p_B(x)\} & \{\neg p^d_P(x,y), p_S(x,y)\} \\
\{\neg p_{\overline{A}}(x), \neg p_C(x)\} & \{\neg p_R(x,y), \neg p_S(x,y), p^u_P(x,y)\} \\
\{\neg p_A(x), \neg p_{\overline{A}}(x)\} & \{\neg p^d_P(x,y), p^u_P(x,y)\} \\
\{\neg p_B(x), \neg p^u_P(x,y), p_C(x)\} &
\end{array}$$

To keep the clause set small, no renaming was performed on the non-atomic terms $\neg B$, $\neg C$, $\neg B \sqcup \neg C$, etc.

Define a dependency relation $\succ^1_c$ on the predicate symbols by $p_A \succ^1_c p_B$, if there is a definition $\phi \to \psi$ in $\overline{\Xi}\Pi(\overline{\Gamma})$ such that p_A occurs in ϕ and p_B occurs in ψ. Let $\succ_{\mathsf{S}}$ be an ordering on the predicate symbols in $\overline{\Xi}\Pi(\overline{\Gamma})$ which is compatible with the transitive closure of $\succ^1_c$. Due to the acyclicity of the terminology and due to fact that we split role definitions, it is possible to find such an ordering.

While an ordering restriction of resolution and factoring is optional, our selection function $S_{\mathcal{TAB}}$ selects the literal $\neg p_{\overline{A}}(x)$ in a clause $\{\neg p_{\overline{A}}(x), \neg p_A(x)\}$ originating from $A \dot{\sqsubseteq} \neg\overline{A}$. For all other clauses C, let L be an occurrence of a negative literal in C with predicate symbol p_A. Then L is selected in C if and only if either the predicate symbol of L is $\succ_{\mathsf{S}}$-maximal in C, or L is a literal of the form $\neg p_A(s,y)$, where s is a ground term and y is a variable. In our sample clause set above all negative literal occurrences are selected except for the last two clauses in the left column, where in each case only $\neg p_{\overline{A}}(x)$ is selected. All clauses originating from a terminological sentence or from a definition introduced by $\overline{\Xi}$ contain negative literals, one of which is selected. Consequently, no factoring steps are possible and the clauses may only be used as negative premises of resolution steps. Observe that all clauses originating from the translation of assertional sentences are ground unit clauses.

In all clauses except those of the form

$$\{\neg p_A(x)_+, \neg p_R(x,y), p_B(y)\} \tag{1}$$

the selected literal (marked by $_+$) contains all variables of the clause, and with the exception of

$$\{\neg p_A(x)_+, p_R(x, f(x))\} \quad \text{and} \quad \{\neg p_A(x)_+, p_B(f(x))\} \tag{2}$$

no variables occur as arguments of compound terms.

Inferences with premises like (1) are problematic, since the resolvent may contain more free variables than the positive premise of the inference step. Suppose we have derived a clause of the form $\{p_{A_1}(\underline{a}), p_{A_2}(\underline{a}), p_{A_3}(\underline{a})\}$ (momentarily ignoring the "Split" rule) and $\overline{\Xi}\Pi(\overline{\Gamma})$ contains $\{\neg p_{A_i}(x)_+, \neg p_{R_i}(x, y), p_{B_i}(y)\}$, for $1 \leq i \leq 3$. Without taking further restrictions into account, we can derive the clause

$$\{\neg p_{R_1}(\underline{a}, x), p_{B_1}(x), \neg p_{R_2}(\underline{a}, y), p_{B_2}(y), \neg p_{R_3}(\underline{a}, z), p_{B_3}(z)\}.$$

It contains more variables than any clause in $\overline{\Xi}\Pi(\overline{\Gamma})$.

In general, the positive premise of a resolution inference step with a clause like (1) is a ground clause $\{p_0(s)\} \cup D_1$ such that no literals in D_1 are selected. The conclusion of the inference step is a clause $C_1 = \{\neg p_1(s, y)_+, p_2(y)\} \cup D_1$, with one free variable. However, the literal $\neg p_1(s, y)$ is selected by $S_{\mathcal{TAB}}$ and no inference steps are possible on $\{p_2(y)\} \cup D_1$ (which again contains no selected literals). The only clauses we can derive containing a positive literal with predicate symbol p_1 will be ground clauses, that is, clauses of the form $C_2 = \{p_1(s, t)\} \cup D_2$. The conclusion of an inference step between C_1 and C_2 is the ground clause $\{p_2(t)\} \cup D_1 \cup D_2$. Consequently, all clauses occurring in a derivation from the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$ contain at most two variables.

Note that C_1 is not a DL-clause. Also note for the argument it is important that a negative binary literal occurs in (1). This is the reason for excluding role negation as well as the top role from the language.

The problem with inferences involving negative premises of the forms (2) is that resolvents may contain terms of greater depth than the positive premise of the inference. Nevertheless, we can still show that there is an upper bound on the depth of terms.

With every literal $L = (\neg)p(t_1, \ldots, t_n)$ we associate a complexity measure $c_L = (p, \{t_1, \ldots, t_n\})$. Complexity measures on ground literals are compared by the ordering $\succ_c^{\text{lit}}$ given by the lexicographic combination of the ordering $\succ_{\mathsf{S}}$ and the multiset extension of the strict subterm ordering $\succ_{mul}^{s}$. The ordering is lifted from ground to non-ground expressions in the usual way. The ordering on clauses is the multiset extension of $\succ_c^{\text{lit}}$. It is straightforward to check that any inference step from a positive premise C by (ordered) resolution or (ordered) factoring will result in a clause D such that c_C is greater than c_D with respect to $\succ_c^{\text{mul}}$.

Theorem 6. *Let Γ be a descriptive knowledge base and let N be the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$. Then any derivation from N by (ordered) resolution with selection as determined by $S_{\mathcal{TAB}}$ and (ordered) factoring terminates.*

6 Simulation of tableaux for $\mathcal{ALC}$

We now consider the relation between the selection-based decision procedure and a standard tableaux decision procedure for $\mathcal{ALC}$.

Recall, for example, from [18], the definition of unfolding and the inferences rules. Unfolding iteratively replaces the concept, respectively role, symbols in ABox sentences by the right-hand sides of their definitions. For example, given $A \doteq B \sqcup C$ and $B \dot{\sqsubseteq} \forall P.C$ two consecutive unfolding steps replace $a \in A$ first by $a \in B \sqcup C$ and then by $a \in (\forall P.C) \sqcup C$. Note that fully unfolding all definitions may lead to an exponential increase in the size of the ABox. For this reason it is advisable to interleave unfolding steps with applications of the transformation rules described next. This is known as *lazy unfolding* [1,10].

The set of transformation rules for testing the satisfiability of an $\mathcal{ALC}$-ABox are:

1. $\Delta \Rightarrow_{\sqcap} \Delta \cup \{a \in C, a \in D\}$, if $a \in (C \sqcap D)$ is in Δ, $a \in C$ and $a \in D$ are not both in Δ.
2. $\Delta \Rightarrow_{\sqcup} \Delta \cup \{a \in E\}$, if $a \in (C \sqcup D)$ is in Δ, neither $a \in C$ nor $a \in D$ is in Δ, and $E = C$ or $E = D$.
3. $\Delta \Rightarrow_{\exists} \Delta \cup \{(a, b) \in R, b \in C\}$, if $a \in \exists R.C$ is in Δ, there is no c such that both $(a, c) \in R$ and $c \in C$ are in Δ, and b is a new object symbol with respect to Δ.
4. $\Delta \Rightarrow_{\forall} \Delta \cup \{b \in C\}$, if $a \in \forall R.C$ and $(a, b) \in R$ are in Δ, and $b \in C$ is not in Δ.
5. $\Delta \Rightarrow_{\bot} \Delta \cup \{a \in \bot\}$, if $a \in A$ and $a \in \neg A$ are in Δ, where A is a concept symbol.

Let $\Rightarrow_{\mathcal{TAB}}$ be the transitive closure of the union of the transformation rules given above. An ABox Δ contains a *clash* if $a \in \bot$ is in Δ. An ABox Δ is satisfiable iff there exists an ABox Δ' such that (i) $\Delta \Rightarrow_{\mathcal{TAB}} \Delta'$, (ii) no further applications of $\Rightarrow_{\mathcal{TAB}}$ to Δ' are possible, and (iii) Δ' is clash-free.

The correspondence between the tableaux-based decision procedure and the selection-based decision procedure is not difficult to see. Remember that for every concept C and every role R, which may possibly occur in an ABox during a satisfiability test, there exist corresponding predicate symbols p_C and p_R in the clausal form of $\overline{\exists}\Pi(\overline{\Gamma})$. Likewise for every object symbol a we will have a corresponding term t_a.

Lazy unfolding steps can be simulated by resolution inference steps with the clauses we obtain from the translation of the TBox. Suppose the knowledge base Γ contains either the terminological sentence $A \dot{\sqsubseteq} C$ or $A \doteq C$. Then unfolding will replace an assertional sentence $a \in A$ by $a \in C$. Correspondingly, our clause set contains the clauses $\{\neg p_A(x)_+, p_C(x)\}$ and $\{p_A(t_a)\}$ and one resolution step produces $\{p_C(t_a)\}$. Suppose Γ contains $A \doteq C$ and $a \in \neg A$. In this case, unfolding replaces the assertional sentence by $a \in D$, where D is the negation normal form of $\neg C$. Since $A \in \mathrm{D}_{\doteq}(\Gamma)$, occurrences of $\neg A$ have been replaced by $\overline{A}$ in $\overline{\Gamma}$, and the clause set contains $\{\neg p_{\overline{A}}(x)_+, p_D(x)\}$ and $\{p_{\overline{A}}(t_a)\}$. We derive $\{p_D(t_a)\}$. These inference steps obey the restrictions enforced by the selection

function $S_{\mathcal{TAB}}$. p_C and p_D are the predicate symbols associated with the concepts C and D. Thus, in case C and D are non-atomic concept terms, the clause set contains additional clauses originating from the translation of the definition $\forall\, x{:}\, p_C(x) \rightarrow \pi(C, x)$. The inference steps described above will be followed by resolution steps with these clauses which completes the unfolding step.

Transformation by the inference rules is simulated according to the following.

1. An application of the $\Rightarrow_\sqcap$ rule corresponds to a resolution inference step between a ground clause $\{p_{C\sqcap D}(t_a)\}$ and clauses $\{\neg p_{C\sqcap D}(x), p_C(x)\}$ and $\{\neg p_{C\sqcap D}(x), p_D(x)\}$, generating the resolvents $\{p_C(t_a)\}$ and $\{p_D(t_a)\}$.
2. An application of the $\Rightarrow_\sqcup$ rule corresponds to a resolution inference step between a ground unit clause $\{p_{C\sqcup D}(t_a)\}$ and $\{\neg p_{C\sqcup D}(x), p_C(x), p_D(x)\}$. We then apply the splitting rule to the conclusion $\{p_C(t_a), p_D(t_a)\}$ which will generate two branches, one on which our set of clauses contains $\{p_C(t_a)\}$ and one on which it contains $\{p_D(t_a)\}$.
3. An application of the $\Rightarrow_\exists$ rule corresponds to resolution inference steps between $\{p_{\exists R.C}(t_a)\}$, $\{\neg p_{\exists R.C}(x), p_R(x, f(x))\}$, and $\{\neg p_{\exists R.C}(x), p_C(f(x))\}$. This will add $\{p_R(t_a, f(t_a))\}$ and $\{p_C(f(t_a))\}$ to the clause set. The term $f(t_a)$ corresponds to the new object symbol b introduced by the $\Rightarrow_\exists$ rule, that is, $t_b = f(t_a)$.
4. An application of the $\Rightarrow_\forall$ rule corresponds to two consecutive inference steps. Here, the set of clauses contains $\{p_{\forall R.C}(t_a)\}$ and $\{p^u_R(t_a, t_b)\}$ (to obtain $\{p^u_R(t_a, t_b)\}$ an inference step with a clause $\{\neg p^d_R(x, y), p^u_R(x, y)\}$ may be necessary). First, $\{p_{\forall R.C}(t_a)\}$ is resolved with $\{\neg p_{\forall R.C}(x), \neg p^u_R(x, y), p_C(y)\}$ to obtain $\{\neg p^u_R(t_a, y), p_C(y)\}$. Then this clause is resolved with $\{p^u_R(t_a, t_b)\}$ to obtain $\{p_C(t_b)\}$.
5. For applications of the $\Rightarrow_\bot$ rule we distinguish two cases. If A is not in $\mathrm{D}_{\doteq}(\Gamma)$, then the set of clauses contains $\{p_A(t_a)\}$ and $\{p_{\neg A}(t_a)\}$. Two consecutive inference steps using these two clauses and $\{\neg p_{\neg A}(x)_+, \neg p_A(x)\}$, the definition of $\neg A$, produce the empty clause. Otherwise, A is in $\mathrm{D}_{\doteq}(\Gamma)$ and the set of clauses contains $\{p_A(t_a)\}$ and $\{p_{\overline{A}}(t_a)\}$. In this case the empty clause can be derived with $\{\neg p_{\overline{A}}(x)_+, \neg p_A(x)\}$.

All these resolution inference steps strictly obey the restrictions enforced by the selection function $S_{\mathcal{TAB}}$. Consequently:

Theorem 7. *The selection-based resolution decision procedure with selection function $S_{\mathcal{TAB}}$ p-simulates tableaux-based decision procedures for $\mathcal{ALC}$.*

Moreover, the described procedure provides a basis for defining tableaux-based decision procedures for extensions of $\mathcal{ALC}$ with role hierarchies, role conjunction and/or role disjunction.

For example, Theorem 7 also holds in the presence of role conjunction. Descriptions of tableaux-based procedures often hide some of the inferential effort in side conditions of inference rules. Commonly the following modified version of the $\Rightarrow_\forall$ rule is used.

4′. $\Delta \Rightarrow_\forall \Delta \cup \{b \in C\}$ if $a \in \forall R.C$ is in Δ and $(a, b) \in R$ holds in Δ, and $b \in C$ is not in Δ.

By definition, $(a,b) \in R_1 \sqcap \ldots \sqcap R_n$ *holds in* Δ if and only if, for all i, $1 \leq i \leq n$, $(a,b) \in R_i$ is in Δ.

Obviously, the implicit inferential steps needed to determine whether $(a,b) \in R$ holds in Δ, for complex roles R, have to be taken into account when establishing a relationship to the selection-based decision procedure. Then it becomes evident that any tableaux inference step can be simulated by at most two inference steps in the selection-based decision procedure.

For example, reconsider the knowledge base containing $B \dot{\sqsubseteq} \forall P.C$ and $P \doteq R \sqcap S$. Assume that, in addition, the knowledge base contains the assertional sentences $a \in B$ and $(a,b) \in P$. Unfolding will replace P by $R \sqcap S$. By an application of the analogue of $\Rightarrow_\sqcap$ for roles, $(a,b) \in R$ and $(a,b) \in S$ will be derived. Then, $b \in C$ will be derived by using the modified version of $\Rightarrow_\forall$.

The clauses corresponding to the two assertional sentences are $\{p_B(\underline{a})\}$ and $\{p_P^d(\underline{a},\underline{b})\}$. Simulating the unfolding step and applying $\Rightarrow_\sqcap$ we derive $\{p_R(\underline{a},\underline{b})\}$ and $\{p_S(\underline{a},\underline{b})\}$ from $\{\neg p_P^d(x,y), p_R(x,y)\}$ and $\{\neg p_P^d(x,y), p_S(x,y)\}$. Then resolution steps with $\{\neg p_R(x,y), \neg p_S(x,y), p_P^u(x,y)\}$ give $\{p_P^u(\underline{a},\underline{b})\}$. Finally, $\{p_C(\underline{b})\}$ can be obtained by the simulation of inference with the $\Rightarrow_\forall$ rule.

7 Model generation

As with tableaux-based procedures our selection-based procedure lends itself for the construction of a model when the empty clause was not derived. We briefly describe how this can be done. The results of this section concern the reduct of $\mathcal{ALB}$ without the top role and role negation.

First, define a translation mapping which maps the first-order syntax back to the original syntax. This exploits the one to one correspondence between ground terms and objects, and predicate symbols and concept and role subexpressions. For any ground term t_a, let $\widehat{t_a}$ denote the object symbol a uniquely associated with t_a. Let $\widehat{C}$ and $\widehat{R}$ denote the concept and role obtained by replacing any occurrences of $\overline{A}$ by $\neg A$, and P^u and P^d by P, respectively. Now, define the mapping Π^{-1} by $\Pi^{-1}(\{p_C(t_a)\}) = \widehat{t_a} \in \widehat{C}$, $\Pi^{-1}(\{p_R(t_a,t_b)\}) = (\widehat{t_a},\widehat{t_b}) \in \widehat{R}$ and the straightforward extension to clauses and sets of clauses.

By a model we mean a set I of ground atoms. The presence of an atom A in I means A is true in I, and the absence of A means $\neg A$ is true in I. In general, a clause C is true in I iff for all ground substitutions σ there is a literal L in $C\sigma$ which is true in I. Falsehood is defined dually.

Theorem 8. *Let Γ be a descriptive knowledge base, N the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$, and N_∞ a satisfiable, saturated clause set derivable from N by using the selection-based decision procedure. Let I be the set of ground positive unit clauses in N_∞. Then, I is a model of N_∞ and N, and $\Pi^{-1}(I)$ is a model of Γ.*

Proof. As noted before, during the derivation only ground unit clauses and clauses of the form $\{\neg p_1(s,y), p_2(y)\}$ (with s ground) are generated. To prove that I is a model of N_∞ we have to show that any ground instance of a clause C in N_∞ is true in I. This obviously holds for any of the positive ground unit

clauses in N_∞. Also, any negative ground unit clause $\{\neg A\}$ is true in I. Let $C = \{\neg A\sigma\} \cup D\sigma$ be the ground instance of clause in N_∞ with selected literal $\neg A$. If $A\sigma$ is not in I, then C is true. Otherwise, $\{A\sigma\}$ is an element of N_∞ and we have derived $D\sigma$ at one stage of the derivation. Consequently, one of the split components of $D\sigma$ is in N_∞. The split components of a ground clause are unit ground clauses, for which we have already shown that they are true in I. It follows that C is true in I. Hence I is a model of N_∞ and also N.

By Theorem 7 and the correspondence between predicate symbols and ground terms in the clause set and the symbols in the knowledge base, it follows that $\Pi^{-1}(I)$ is identical to a clash-free knowledge base Γ_∞ derivable from Γ such that no further applications of $\Rightarrow_{\mathcal{TAB}}$ are possible. It follows from the results of [18] that $\Pi^{-1}(I) = \Gamma_\infty$ is (a syntactic representation of) a model of Γ. □

The finite model property is an immediate consequence of Theorems 5 and 8.

Corollary 9. *Let Γ be a descriptive knowledge base. If Γ is satisfiable, then it has a model of finite size.*

8 Conclusion

The class of DL-clauses is not comparable with the guarded fragment or the loosely guarded fragment. In the guarded fragments the conditional quantifiers may not include negations or disjunctions. On the other hand, the guarded fragments allow predicates of arbitrary arity. Recently it has been shown that the extension of the guarded fragment with two interacting transitive relations and equality is undecidable. However, basic modal logic plus transitivity is known to be decidable. Therefore, looking at more restricted classes than the guarded fragment may lead to better characterisations of the connection between modal logics and decidable subclasses of first-order logic (e.g. [8]).

The class of DL-clauses is more restrictive than the class One-Free, which stipulates that quantified subformulae have at most one free variable. But it is possible to extend $\mathcal{ALB}$ by certain restricted forms of role composition (e.g., positive occurrences), for which the procedure described in Section 4 remains a decision procedure. The corresponding clausal class is distinct from the One-Free class. It is known from the literature on algebraic logic that arbitrary occurrences of composition in the presence of role negation leads to undecidability, though.

The resolution decision procedures of [7,19] have the disadvantage that they are based on a non-liftable ordering refinement. As a consequence certain standard simplification rules, e.g. tautology deletion, have to be restricted for completeness. Real world knowledge bases typically contain hundreds of concept definitions. The corresponding clauses can be used to derive an extensive number of tautologies. Our approach does not have this drawback. In addition to using liftable orderings, the resolution framework here is equipped with a general notion of redundancy which accommodates most standard simplification rules including tautology deletion, condensing, subsumption deletion, as well as non-standard theory-specific simplification rules. For a discussion of redundancy and fairness consult [4].

Acknowledgements. The authors wish to thank the anonymous referees for their comments. The work of the first author was supported by EPSRC Grant GR/K57282.

References

1. F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems or "making KRIS get a move on". *Applied Intelligence*, 4(2):109–132, 1994.
2. M. Baaz, C. Fermüller, and A. Leitsch. A non-elementary speed-up in proof length by structural clause form transformation. In *Proc. LICS'94*, pages 213–219. IEEE Computer Society Press, 1994.
3. L. Bachmair and H. Ganzinger. Ordered chaining calculi for first-order theories of binary relations. Research report MPI-I-95-2-009, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1995. To appear in *J. ACM*.
4. L. Bachmair and H. Ganzinger. A theory of resolution. Research report MPI-I-97-2-005, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1997. To appear in J. A. Robinson and A. Voronkov (eds.), *Handbook of Automated Reasoning*.
5. H. de Nivelle. A resolution decision procedure for the guarded fragment. In *Proc. CADE-15*, LNAI 1421, pages 191–204. Springer, 1998.
6. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. Logic and Computation*, 4:423–452, 1994.
7. C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Method for the Decicion Problem*. LNCS 679. Springer, 1993.
8. H. Ganzinger, U. Hustadt, C. Meyer, and R. A. Schmidt. A resolution-based decision procedure for extensions of K4. To appear in *Advances in Modal Logic, Volume 2*. CSLI Publications, 1999.
9. E. Hemaspaandra. The price of universality. *Notre Dame J. Formal Logic*, 37(2):174–203, 1996.
10. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, Manchester, UK, 1997.
11. U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. IJCAI'97*, pages 202–207. Morgan Kaufmann, 1997.
12. W. H. Joyner Jr. Resolution strategies as decision procedures. *J. ACM*, 23(3):398–417, 1976.
13. B. Kallick. A decision procedure based on the resolution method. In *Information Processing 68, Volume 1*, pages 269–275. North-Holland, 1968.
14. M. Marx. Mosaics and cylindric modal logic of dimension 2. In *Advances in Modal Logic, Volume 1*, Lecture Notes 87, pages 141–156. CSLI Publications, 1996.
15. M. Paramasivam and D. A. Plaisted. Automated deduction techniques for classification in description logic systems. *J. Automated Reasoning*, 20:337–364, 1998.
16. D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *J. Symbolic Computation*, 2:293–304, 1986.
17. R. A. Schmidt. Decidability by resolution for propositional modal logics. To appear in *J. Automated Reasoning*.
18. M. Schmidt-Schauß and G. Smolka. Attributive concept description with complements. *Artifical Intelligence*, 48:1–26, 1991.
19. T. Tammet. Using resolution for extending KL-ONE-type languages. In *Proc. CIKM'95*, 1995.
20. A. Urquhart. The complexity of propositional proofs. *Bull. Symbolic Logic*, 1(4):425–467, 1995.