SCAN is complete for all Sahlqvist formulae

V. Goranko[1], U. Hustadt[2], R. A. Schmidt[3,4] and D. Vakarelov[5]

[1] Department of Mathematics, Rand Afrikaans University, Johannesburg, South Africa
vfg@na.rau.ac.za
[2] Department of Computer Science, University of Liverpool, UK,
U.Hustadt@csc.liv.ac.uk
[3] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[4] Department of Computer Science, University of Manchester, UK,
schmidt@cs.man.ac.uk
[5] Department of Mathematical Logic with Laboratory for Applied Logic, Sofia University, Bulgaria, dvak@fmi.uni-sofia.bg

Abstract. SCAN is an algorithm for reducing monadic existential second-order logic formulae to equivalent simpler formulae, often first-order logic formulae. It is provably impossible for such a reduction to first-order logic to be always successful, even if there is an equivalent first-order formula for a second-order logic formula. In this paper we show that SCAN successfully computes the first-order equivalents of all Sahlqvist formulae in the classical (multi-)modal language.

1 Introduction

One of the most general result on first-order definability and completeness in modal logic is Sahlqvist's [15] theorem where two notable facts are proved for a large, syntactically defined class of modal formulae, now called Sahlqvist formulae: first, the *correspondence result,* which says that Sahlqvist formulae all define first-order conditions on Kripke frames and these conditions can be effectively "computed" from the modal formulae; and second, the *completeness result*, which says that all those formulae are canonical, i.e. valid in their respective canonical frames, hence axiomatise completely the classes of frames satisfying the corresponding first-order conditions. The class of Sahlqvist formulae has been studied extensively, and several alternative proofs and generalisations have been proposed (see [18, 16, 9, 4, 8, 7]).

Computing the first-order equivalent (if it exists) of a modal formula amounts to elimination of the universal monadic second-order quantifiers expressing the validity in a frame of that formula or, equivalently, the elimination of the existential monadic second-order quantifiers expressing the satisfiability of the formula. The best know algorithm for elimination of existential second-order quantifiers is SCAN [6], though a number of alternative algorithms can be found in the literature [5, 12, 17]. Of course, the algorithm does not work for all first-order definable modal formulae (at least because first-order definability of modal formulae is not decidable), but it has been proved correct whenever it gives an answer.

In this paper we show that SCAN successfully computes the first-order equivalents of all classical Sahlqvist formulae. To our knowledge, this is the first completeness result for an automated quantifier elimination procedure.

In this paper basic familiarity with the syntax and semantics of modal logic and first-order resolution is assumed. State-of-the-art references on modal logic are [2, 3, 10, 19]. State-of-the-art references on automated deduction, including resolution, are [1, 14].

2 Sahlqvist formulae

We fix an arbitrary propositional (multi-)modal language. For technical convenience we assume that the primitive connectives in the language are $\neg$, $\wedge$, and the diamonds, while the others are definable as usual. Most of the following definitions are quoted from [2].

An occurrence of a propositional variable in a modal formula φ is *positive (negative)* iff it is in the scope of an even (odd) number of negations. A modal formula φ is *positive (negative) in a variable* q iff all occurrences of q in φ are positive (negative). A modal formula φ is *positive* (*negative*) iff all occurrences of propositional variables in φ are positive (negative). A *boxed atom* is a formula of the form $\Box_{k_1} \dots \Box_{k_n} q$, where $\Box_{k_1}, \dots, \Box_{k_n}$ is a (possibly empty) string of (possibly different) boxes and q is a propositional variable.

A *Sahlqvist antecedent* is a modal formula constructed from the propositional constants $\bot$ and $\top$, boxed atoms and negative formulae by applying $\vee$, $\wedge$, and diamonds. A *definite Sahlqvist antecedent* is a Sahlqvist antecedent obtained without applying $\vee$ (i.e. constructed from the propositional constants $\bot$ and $\top$, boxed atoms and negative formulae by applying only $\wedge$ and diamonds). A *(definite) Sahlqvist implication* is a modal formula $\varphi \to \psi$ where φ is a (definite) Sahlqvist antecedent and ψ is a positive formula. A *(definite) Sahlqvist formula* is a modal formula constructed from (definite) Sahlqvist implications by freely applying boxes and conjunctions, and by applying disjunctions to formulae without common propositional variables. A *basic Sahlqvist formula* is a definite Sahlqvist formula obtained from definite Sahlqvist implications without applying conjunctions.

The following easy observations (see [7]) will be used further.

Proposition 1.

1. *Every Sahlqvist implication is equivalent to a conjunction of definite Sahlqvist implications.*
2. *Every Sahlqvist formula is equivalent to a conjunction of basic Sahlqvist formulae.*

Theorem 1. *Every Sahlqvist formula φ is locally first-order definable, i.e. there is a first-order formula $\alpha_\varphi(x)$ of the first-order language with equality and binary relational symbols corresponding to the modal operators in φ, such that for every Kripke frame F with a domain W and $w \in W$, $F, w \models \varphi$ iff $F \Vdash_w \alpha_\varphi(x)$ (where $\models$ denotes modal validity, while $\Vdash$ denotes first-order truth).*

3 The SCAN algorithm

SCAN reduces existentially quantified second-order sentences to equivalent first-order formulations. Given a second-order sentence containing existentially quantified second-order variables, the algorithm generates sufficiently many logical consequences, eventually keeping from the resulting set of formulae only those in which no second-order variables occur. The algorithm involves three stages:

(i) transformation to clausal form and Skolemization;
(ii) C-resolution;
(iii) reverse Skolemization.

The input of SCAN is a second-order formula of the form $\exists Q_1 \ldots \exists Q_k\, \psi$, where the Q_i are existentially quantified unary predicate variables and ψ is a first-order formula. In the first stage SCAN converts ψ into clausal form, written $\mathrm{Cls}(\psi)$, by transformation into conjunctive normal form, inner Skolemization, and clausifying the Skolemized formula [11, 13].

In the second stage SCAN performs a special kind of constraint resolution, called *C-resolution*. It generates all and only resolvents and factors with the second-order variables that are to be eliminated. When computing frame correspondence properties all existentially quantified second-order variables $Q_1, \ldots, Q_k$ are eliminated. In the process the algorithm produces equations and inequations of terms.

We use a slightly different presentation of C-resolution compared to [6] which allows for a more concise proof of our main result.

In the following *clauses* are considered to be multisets of literals. A *literal* is either an *atom* $P(t_1, \ldots, t_n)$ (also called *positive literal*) where P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, an equation $t_1 \approx t_2$ where t_1 and t_2 are terms, a *negative literal* $\neg P(t_1, \ldots, t_n)$, or an inequation $t_1 \not\approx t_2$. We consider clauses to be identical up to variable renaming, that is, if C and D are clauses such that there exists a variable renaming σ with $C\sigma = D$, then we consider C and D to be identical. A *subclause* of a clause C is a submultiset of C. If $D = C \vee s_1 \not\approx t_1 \vee \ldots \vee s_n \not\approx t_n$ is a clause such that C does not contain any inequations, and σ is a most general unifier of $s_1 \approx t_1 \wedge \ldots \wedge s_n \approx t_n$ (that is, $s_i\sigma \approx t_i\sigma$ for every i, $1 \leq i \leq n$, and σ is the most general substitution with this property), then we say $C\sigma$ is obtained from D by *constraint elimination*. Note that $C\sigma$ is unique up to variable renaming.

A literal with a unary predicate symbol among $Q_1, \ldots, Q_k$, is a **Q**-literal, while a literal with a binary predicate symbol (not including equality) is a **R**-literal.

Let C be a clause. A subclause D of C is a *split component* of C iff (i) if L' is a literal in C but not in D, then L' and D are variable-disjoint and (ii) there is no proper subclause $D' \subset D$ satisfying property (i). If $C \vee L_1 \vee \ldots \vee L_n$, $n \geq 2$, is a clause and there exists a most general unifier σ of $L_1, \ldots, L_n$, then $(C \vee L_1)\sigma$ is called a *factor* of $C \vee L_1 \vee \ldots \vee L_n$. The *condensation* $\mathrm{cond}(C)$ of a clause C is a minimal subclause D of C such that there exists a substitution σ with $L\sigma \in D$ for every $L \in C$.

A *clause set* is a set of clauses. We use $\uplus$ to denote the *disjoint union* of two sets.

Derivations in the C-resolution calculus are constructed using expansion rules and deletion rules:

The only expansion rule in the C-resolution calculus is the following:

Deduction: $$\frac{N}{N \cup \{C\}}$$
if C is a C-resolvent or C-factor of premises in N.

C-resolvents and C-factors are computed using the following two *inference rules*:

C-resolution:
$$\frac{C \vee Q(s_1,\ldots,s_n) \quad D \vee \neg Q(t_1,\ldots,t_n)}{C \vee D \vee s_1 \not\approx t_1 \vee \ldots \vee s_n \not\approx t_n}$$
provided the two premises have no variables in common and $C \vee Q(s_1,\ldots,s_n)$ and $D \vee \neg Q(t_1,\ldots,t_n)$ are distinct clauses. (As usual we assume the clauses are normalised by variable renaming so that the premises of the C-resolution do not share any variables.) The clause $C \vee Q(s_1,\ldots,s_n)$ is called the *positive premise* and the clause $D \vee \neg Q(t_1,\ldots,t_n)$ the *negative premise* of the inference step. The conclusion is called a *C-resolvent* with respect to Q.

C-factoring:
$$\frac{C \vee Q(s_1,\ldots,s_n) \vee Q(t_1,\ldots,t_n)}{C \vee Q(s_1,\ldots,s_n) \vee s_1 \not\approx t_1 \vee \ldots \vee s_n \not\approx t_n}$$
The conclusion is called a *C-factor* with respect to Q.

In addition, the C-resolution calculus includes four deletion rules:

Purity deletion:
$$\frac{N \uplus \{C \vee Q(s_1,\ldots,s_n)\}}{N}$$
if all inferences with respect to Q with $C \vee Q(s_1,\ldots,s_n)$ as a premise have been performed.

Subsumption deletion:
$$\frac{N \uplus \{C, D\}}{N \uplus \{C\}}$$
if C subsumes D, i.e. there is a substitution σ such that $C\sigma \subseteq D$.

Constraint elimination:
$$\frac{N \uplus \{C\}}{N \uplus \{D\}}$$
if D is obtained from C by constraint elimination.

Condensation:
$$\frac{N \uplus \{C\}}{N \uplus \{\mathrm{cond}(C)\}}$$

A *derivation* in the C-resolution calculus is a (possibly infinite) sequence $N_0, N_1, \ldots$ of clause sets such that for every $i \geq 0$, N_{i+1} is obtained from N_i by application of an *expansion rule* or a *deletion rule* such that if a deletion rule can be applied to N_i then N_{i+1} is obtained from N_i by application of a deletion rule.

The algorithm generates all possible C-resolvents and C-factors with respect to the predicate variables $Q_1, \ldots, Q_k$. When all C-resolvents and C-factors with respect to a particular Q_i-literal and the rest of the clause set have been generated, purity deletion removes all clauses in which this literal occurs. Subsumption deletion helps simplify clause sets in the derivation, though, in general, the subsumption deletion rule is optional for the sake of soundness. For the present application subsumption deletion is essential for avoiding simple forms of infinite derivations.

Note that in constraint resolution inferences variables are never instantiated.

If the C-resolution stage terminates, this yields a set N of clauses in which the specified second-order variables are eliminated. This set is satisfiability-equivalent to the second-order formula originally given to the algorithm. If no clauses remain after purity deletion, i.e. $N = \emptyset$, then the original formula is a tautology; if C-resolution produces the empty clause, then it is unsatisfiable. If N is non-empty, finite and does not contain the empty clause, then in the third stage, SCAN attempts to reconstruct the quantifiers for the Skolem functions by reversing Skolemization. Reversing Skolemization is not always possible, for instance, it is not possible if the input formula is not first-order definable.

If the input formula is not first-order definable and stage two terminates successfully yielding a non-empty set not containing the empty clause then SCAN produces equivalent second-order formulae in which the specified second-order variables are eliminated but quantifiers involving functions occur. These typically involve Henkin quantifiers. If SCAN terminates and the reverse Skolemization (unskolemization) is successful, then the result is a first-order formula logically equivalent to the second-order input formula.

Since we intend to apply SCAN to Sahlqvist formulae, we define a translation of modal formulae into second-order formulae. Let $\Pi(\varphi) = \forall Q_1 \ldots \forall Q_m \forall x\, \mathrm{ST}(\varphi, x)$, where $\mathrm{ST}(\varphi, x)$ is the (local) standard translation of a modal formula φ with a free variable x, and $Q_1, \ldots, Q_m$ are all the unary predicates occurring in $\mathrm{ST}(\varphi, x)$. The standard translation $\mathrm{ST}(\varphi, x)$ itself is inductively defined as follows:

$$\begin{aligned} \mathrm{ST}(q_i, x) &= Q_i(x) & \mathrm{ST}(\neg\phi, x) &= \neg\, \mathrm{ST}(\phi, x) \\ \mathrm{ST}(\phi \wedge \psi, x) &= \mathrm{ST}(\phi, x) \wedge \mathrm{ST}(\psi, x) & \mathrm{ST}(\Diamond_{k_i}\phi, x) &= \exists y(xR_{k_i}y \wedge \mathrm{ST}(\phi, y)) \end{aligned}$$

where Q_i is a unary predicate symbol uniquely associated with the propositional variable q_i, R_{k_i} and is a binary predicate symbol uniquely associated with the modal operator $\Diamond_{k_i}$ (representing the accessibility relation associated with $\Diamond_{k_i}$), and x is either a first-order variable or a constant.

Let SCAN* denote the extension of the SCAN algorithm with the preprocessing and postprocessing necessary for computing first-order correspondences for modal logic formulae. The preprocessing involves translating the given modal formula to the second-order formula $\Pi(\varphi)$ and negating the result. The postprocessing involves negating the result output by SCAN, if it terminates successfully.

4 Completeness of SCAN for Sahlqvist formulae

In the following, we show that SCAN* is complete for Sahlqvist formulae. We need to prove that for any second-order formula ψ we obtain as result of preprocessing of a Sahlqvist formula φ, SCAN can compute a first-order equivalent for ψ. To this end, one has to show two properties:

1. The computation of C-resolvents and C-factors terminates when applied to the set $\mathrm{Cls}(\psi)$ of clauses obtained from ψ, i.e. SCAN can generate only finitely many new clauses in the process.
2. The resulting first-order formula, which in general contains Skolem functions, can be successfully unskolemized.

Since every Sahlqvist formula is equivalent to a conjunction of basic Sahlqvist formulae, it suffices to prove the result for every basic Sahlqvist formula φ.

We first consider the case when φ is obtained from definite Sahlqvist implications by only applying boxes but not disjunctions.

Theorem 2. *Given any definite Sahlqvist implication φ,* SCAN* *effectively computes a first-order formula α_φ logically equivalent to φ.*

Proof (Sketch). We show that the clauses in $N = \mathrm{Cls}(\mathrm{ST}(\neg\varphi, a))$ contain either only negative **Q**-literals or are exactly one positive **Q**-literal. Any inference step then has a conclusion with strictly less negative **Q**-literal than the negative premise of the inference step. This implies the termination of any C-resolution derivation from N. By an inductive argument we can show that the inequations and terms occurring in derived clauses can be successfully unskolemized.

The theorem can be extended to the case of Sahlqvist formulae obtained from definite Sahlqvist implications by applying boxes and conjunctions.

In contrast to definite Sahlqvist antecedents, Sahlqvist antecedents can include disjunction as a connective. This makes the proof of completeness of SCAN with respect to Sahlqvist implications much more involved. The cornerstone of our proof is the notion of a *chain*.

Let $(t_1, \ldots, t_n)$ be an ordered sequence of pairwise distinct terms. A *chain* C over $(t_1, \ldots, t_n)$ is a clause containing only literals of the form $(\neg)R_{k_i}(s,t)$ and $(\neg)Q_j(u)$ such that the following three conditions are satisfied:

(1) for every i, $1 \leq i \leq n{-}1$, either $\neg R_{k_i}(t_i, t_{i+1})$ or $R_{k_i}(t_i, t_{i+1})$ is in C;
(2) for every $(\neg)R_{k_i}(u,v) \in C$, $u = t_j$ and $v = t_{j+1}$ for some j, $1 \leq j \leq n-1$;
(3) for every $(\neg)Q_j(u) \in C$, $u = t_j$ for some j, $1 \leq j \leq n$.

Lemma 1. *Let C be a chain over $(t_1, \ldots, t_n)$. Then there does not exist an ordered sequence $(s_1, \ldots, s_m)$ of pairwise distinct terms which is distinct from $(t_1, \ldots, t_n)$ such that C is also a chain over $(s_1, \ldots, s_m)$.*

The *length* of a chain C over $(t_1, \ldots, t_n)$ is n. Note that by Lemma 1 the chain C uniquely determines $(t_1, \ldots, t_n)$. So, the length of a chain is a well-defined notion.

Lemma 2. *Let Σ be a finite vocabulary of unary and binary predicate symbols, constant symbols and unary function symbols. If there is a bound l_b on the length of a chain and a bound d_b on the depth of terms that can occur in a chain, then the number of chains over Σ which are distinct modulo variable renaming is finitely bounded.*

Theorem 3. *Given any Sahlqvist implication φ,* SCAN* *effectively computes a first-order formula α_φ logically equivalent to φ.*

Proof (Sketch). Let $N = \mathrm{Cls}(\mathrm{ST}(\neg\varphi, a))$. All the clauses that we can derive from N consist of (not necessarily variable-disjoint) chains over the vocabulary of N. The length of chains is bounded by the length of chains in N (modulo some small polynomial) and the depth of terms is bounded the depth of terms in N. We show that whenever we derive a clause with 'too many' chains in it, then it can be condensed. The specification of 'too many' uses the bound established in the proof of Lemma 2.

As to reverse Skolemization we again show by an inductive argument that the inequations and terms occurring in derived clauses can successfully unskolemized. Consequently, restoration of the quantifiers can always be successfully accomplished during reverse Skolemization.

Finally, the general case of a basic Sahlqvist formula φ obtained from defi nite Sahlqvist implications by applying boxes, as well as disjunctions to formulae sharing no common variables can be dealt with by induction of the number of applications of such disjunctions. The basis of the induction (with no such applications) has been established by Theorem 3.

We can now state the main theorem.

Theorem 4 (Completeness with respect to Sahlqvist formulae).
Given any Sahlqvist formula φ, SCAN* *effectively computes a first-order formula α_φ logically equivalent to φ.*

References

1. W. Bibel and P. H. Schmitt, editors. *Automated Deduction – A Basis for Applications, Vol. I–III*. Kluwer, 1998.
2. P. Blackburn, M. de Rijke, and V. Venema. *Modal Logic*. Cambridge Univ. Press, 2001.
3. A. Chagrov and M. Zakharyaschev. *Modal Logic*, volume 35 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1997.
4. M. de Rijke. How not to generalize Sahlqvist's theorem, 1992. Technical Note, ILLC.
5. P. Doherty, W. Lukaszewics, and A. Szalas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.
6. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7:35–43, 1992.
7. V. Goranko and D. Vakarelov. Sahlqvist formulas unleashed in polyadic modal languages. To appear in *Advances in Modal Logic*, vol. 3, Stanford Univ. Press., 2001.
8. B. Jónsson. On the canonicity of Sahlqvist identities. Preprint 94-012, Department of Mathematics, Vanderbilt University, Nashville, 1994.
9. M. Kracht. How completeness and correspondence theory got married. In M. de Rijke, editor, *Diamonds and Defaults*, pages 175–214. Kluwer, 1993.
10. M. Kracht. *Tools and Techniques in Modal Logic*, volume 142 of *Studies in Logic*. Elsevier, 1999.
11. A. Nonnengart. Strong skolemization. Research Report MPI-I-96-2-010, Max-Planck-Institut für Informatik, Saarbrücken, 1996.
12. A. Nonnengart, H. J. Ohlbach, and A. Szalas. Quantifier elimination for second-order predicate logic. To appear in *Logic, Language and Reasoning: Essays in honour of Dov Gabbay*, Part I, Kluwer, 1997.
13. A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In Robinson and Voronkov [14], chapter 6, pages 335–367.
14. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier Science, 2001.
15. H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In S. Kanger, editor, *Proc. of the 3rd Scandinavian Logic Symposium, 1973*, pages 110–143. North-Holland, 1975.
16. G. Sambin and V. Vaccaro. A new proof of Sahlqvist's theorem on modal definability and completeness. *Journal of Symbolic Logic*, 54(3):992–999, 1989.
17. A. Szałas. On the correspondence between modal and classical logic: An automated approach. *Journal of Logic and Computation*, 3(6):605–620, 1993.
18. J. van Benthem. *Modal Logic and Classical Logic*. The Humanities Press, 1983.
19. M. Zakharyaschev, F. Wolter, and A. Chagrov. *Advanced Modal Logic*. Kluwer, 1998.