SCAN Is Complete for All Sahlqvist Formulae*

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

[1] Rand Afrikaans University, South Africa
vfg@na.rau.ac.za
[2] University of Liverpool, UK
U.Hustadt@csc.liv.ac.uk
[3] University of Manchester, UK
schmidt@cs.man.ac.uk
[4] Sofia University, Bulgaria
dvak@fmi.uni-sofia.bg

Abstract. SCAN is an algorithm for reducing 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 successful for every second-order logic formula which has an equivalent first-order 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 results on first-order definability and completeness in modal logic is Sahlqvist's theorem [16] 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 [19, 17, 11, 4, 10, 9]).

Computing the first-order equivalent of a modal formula (if it exists) amounts to the 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

* This work was supported by EU COST Action 274, and research grants GR/M88761 and GR/R92035 from the UK EPSRC. The first author's work was supported by research grants from Rand Afrikaans University. Part of the work by the third author was done while on sabbatical leave at the Max-Planck-Institut für Informatik, Germany, in 2002. We would also like the thank the referees for their helpful comments.

formula. A well-known algorithm for eliminating existential second-order quantifiers is SCAN [8]. SCAN does not succeed for all first-order definable modal formulae, for example, if we take the K axiom $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$ and replace ϕ and ψ by two distinct instances of the McKinsey axiom, e.g. $\Box\Diamond p \to \Diamond\Box p$ and $\Box\Diamond q \to \Diamond\Box q$, respectively, then SCAN will not terminate on the resulting formula, although it is equivalent to the first-order formula $\top$. However, SCAN has been proved correct whenever it gives an answer [8].

An alternative algorithm to SCAN is DLS [5, 13, 18]. SCAN and DLS are incomparable concerning their ability to compute first-order correspondences for modal formulae. For example, while SCAN successfully computes a first-order equivalent formula for $(\Diamond\Box(p\vee q)\wedge\Diamond\Box(p\vee\neg q)\wedge\Diamond\Box(\neg p\vee q)) \to (\Box\Diamond(p\vee q)\vee\Box\Diamond p\vee\Box\Diamond(p\wedge q))$, DLS fails. On the other hand, there are also examples where DLS succeeds and SCAN fails.

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 a quantifier elimination algorithm.

We assume basic knowledge of the syntax and semantics of modal logic and first-order resolution. State-of-the-art references on modal logic and on automated deduction including resolution are [2, 3] and [1, 15], respectively.

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, e.g. $\varphi \to \psi$ is defined as $\neg(\varphi \wedge \neg\psi)$. 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 $\Box_{k_1}\ldots\Box_{k_n}q$, where $\Box_{k_1}, \ldots, \Box_{k_n}$ 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. It can be shown [9]) that that every Sahlqvist formula is semantically equivalent to a conjunction of basic Sahlqvist formulae.

Example 1.

i. $\Diamond(\neg\Box(p \vee q) \wedge \Diamond\Box\Box q) \to \Box\Diamond(p \wedge q)$ is a Sahlqvist formula.
ii. $\Box\Diamond p \to \Box p$ and $\Box(p \vee q) \to \Box p$ are not Sahlqvist formulae, but can be converted into Sahlqvist formulae defining the same semantic conditions by taking their contrapositions and reversing the signs of p and q.
iii. $\Box\Diamond p \to \Diamond\Box p$ and $\Box(\Box p \to p) \to \Box p$ are not Sahlqvist formulae, and cannot be converted into Sahlqvist formulae defining the same semantic conditions, because both are known not to be first-order definable.

Theorem 1 ([16]). *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).*

The problem whether a given modal formula is frame-equivalent to some Sahlqvist formula is not known to be decidable, and most probably it is not. However, syntactical transformations like the one used in Example 1.ii can extend the applicability of the result we present in this paper.

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 (inner) Skolemization;
(ii) C-resolution;
(iii) reverse Skolemization (unskolemization).

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 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 [12, 14]. 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.

To allow for a more concise proof of the main result, our presentation of C-resolution differs slightly from [8] in that we have explicitly included purity deletion, subsumption deletion, and condensation in the calculus.

In the following *clauses* are considered to be multisets of literals. A *literal* is either an *atom* $P(t_1, \ldots, t_n)$ or $t_1 \approx t_2$ (also called *positive literal*) where P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, or a *negative literal* $\neg P(t_1, \ldots, t_n)$, or $t_1 \not\approx t_2$. An atom $t_1 \approx t_2$ is also called an *equation*, while a negative literal $t_1 \not\approx t_2$ is called an *inequation*. 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 equal. A *subclause* of a clause C is a submultiset of C. A *variable indecomposable clause* is a clause that cannot be split into non-empty subclauses which do not share variables. The finest partition of a clause into variable indecomposable subclauses is its *variable partition*. 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, a literal with a binary predicate symbol (not including equality) is an **R**-literal.

A subclause D of a clause 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 C is *condensed* iff $\mathrm{cond}(C)$ is identical to C. A *clause set* is a set of clauses. With $\uplus$ we 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 *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: $$\frac{N \uplus \{C \vee Q(s_1, \dots, s_n)\}}{N}$$

if all inferences with respect to Q with $C \vee Q(s_1, \dots, s_n)$ 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 of clause sets $N_0, N_1, \dots$ 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*. In the following we assume that the condensation rule is applied eagerly, that is, whenever a clause set N_i in a derivation contains a clause C which is not condensed, the condensation rule is applied to N_i to derive N_{i+1} in which C is replaced by $\mathrm{cond}(C)$.

The algorithm generates all possible C-resolvents and C-factors with respect to the predicate variables $Q_1, \dots, Q_k$. 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. The subsumption deletion rule is optional for the sake of soundness, but helps simplify clause sets in the derivation.

If the C-resolution stage terminates, it yields a set N of clauses in which the specified second-order variables are eliminated. This set is satisfiability-equivalent to the original second-order formula. If no clauses remain after purity deletion, 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 restore the quantifiers from the Skolem functions by reversing Skolemization. This is not always possible, for instance 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 Skolem functions occur and the reverse Skolemization typically produces Henkin quantifiers. If SCAN terminates and reverse Skolemization is successful, then the result is a first-order formula logically equivalent to the second-order input formula.

The SCAN algorithm as described above differs in two details from the SCAN implementation.[1] First, the implementation does not restrict C-factoring inferences to positive literals, although this is sufficient for the completeness of the

[1] `http://www.mpi-sb.mpg.de/units/ag2/projects/SCAN/index.html`

algorithm. Concerning this aspect the implementation also differs from the SCAN algorithm as described in [8]. Second, the implementation of reverse Skolemization does not take into account that variable-disjoint subclauses can be unskolemized separately, which is crucial for our results. For example, unskolemizing the clause $r(x, f(x)) \vee r(y, g(y))$ should result in the first-order formula $(\forall x \exists u R(x, u)) \vee (\forall y \exists v R(y, v))$. Instead the SCAN implementation will produce a formula involving a Henkin quantifier. Obviously, these two deviations between the SCAN algorithm and its implementation are minor and could be easily incorporated into the implementation.

Since we intend to apply SCAN to Sahlqvist formulae, we now define a translation of modal formulae into second-order logic. 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 itself is inductively defined as follows:

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

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 representing the accessibility relation associated with $\Diamond_{k_i}$, and x is a first-order variable. The important property of the standard translation is that it preserves the truth of a modal formula at any state w of a Kripke model M, i.e. $M, w \models \phi$ iff $M \models ST(\phi, x)(w/x)$ where M is regarded as a first-order structure for the language of the standard translation. Thus, validity (resp. satisfiability) of a modal formula is expressed by a universal (resp. existential) monadic second-order formula.

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 of SCAN, if it terminates.

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 obtained by preprocessing from a Sahlqvist formula φ, SCAN can compute a first-order equivalent for ψ. To this end, we have to show two properties:

1. The computation of C-resolvents and C-factors terminates when applied to the set $\mathrm{Cls}(\psi)$ of clauses associated with 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.

We first consider the case when φ is a definite Sahlqvist implication.

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

Proof (Sketch). Let $\varphi = A \to P$. In the preprocessing stage, SCAN* computes $\Pi(\varphi)$ and negates the result. Since $\Pi(\varphi) = \forall Q_1 \ldots \forall Q_m \forall x\, \mathrm{ST}(\varphi, x)$, the negation $\neg\Pi(\varphi)$ is equivalent to $\exists Q_1 \ldots \exists Q_m \exists x\, \mathrm{ST}(\neg\varphi, x) = \exists Q_1 \ldots \exists Q_m \exists x\, \mathrm{ST}(A \wedge \neg P, x)$. So, the initial clause set N we obtain after clausification and Skolemization is given by $\mathrm{Cls}(\mathrm{ST}(A \wedge \neg P, a))$ where a is a Skolem constant. The definite Sahlqvist antecedent A is constructed from propositional constants, boxed atoms and negative formulae by applying only $\wedge$ and diamonds. In addition $\neg P$ is also a negative formula, since P is a positive formula. Thus, $A \wedge \neg P$ is itself a definite Sahlqvist antecedent. Note that

$$\begin{aligned} \mathrm{ST}(\alpha \wedge \beta, a_i) &= \mathrm{ST}(\alpha, a_i) \wedge \mathrm{ST}(\beta, a_i) \\ \mathrm{ST}(\Diamond_{k_i} \alpha, a_i) &= \exists y (a_i R_{k_i} y \wedge \mathrm{ST}(\alpha, y)). \end{aligned}$$

Skolemization will replace the existentially quantified variable y by a new constant a_{i+1} which replaces any occurrence of y in $a_i R_{k_i} y \wedge \mathrm{ST}(\alpha, y)$. Consequently,

$$\begin{aligned} \mathrm{Cls}(\mathrm{ST}(\alpha \wedge \beta, a_i)) &= \mathrm{Cls}(\mathrm{ST}(\alpha, a_i)) \cup \mathrm{Cls}(\mathrm{ST}(\beta, a_i)) \\ \mathrm{Cls}(\mathrm{ST}(\Diamond_{k_i} \alpha, a_i)) &= \{a_i R_{k_i} a_{i+1}\} \cup \mathrm{Cls}(\mathrm{ST}(\alpha, a_{i+1})) \end{aligned}$$

It follows by a straightforward inductive argument that we can divide the clause set $N = \mathrm{Cls}(\mathrm{ST}(A \wedge \neg P, a))$ into a set N_n of clauses which stems from negative formulae occurring in $A \wedge \neg P$ and a set N_p of clauses which stems from the translation of the propositional constants $\top$ and $\bot$, and boxed atoms.

The translation of boxed atoms with respect to a constant a_i is given by

$$\begin{aligned} \mathrm{ST}(\Box_{k_1} \ldots \Box_{k_n} q_j, a_i) = \forall x_1 (a_i R_{k_1} x_1 \to & \\ & \forall x_2 (x_1 R_{k_2} x_2 \to \cdots \\ & \quad \forall x_n (x_{n-1} R_{k_n} x_n \to Q_j(x_n)) \ldots)) \end{aligned}$$

where $n \geq 0$. Clausification transforms $\mathrm{ST}(\Box_{k_1} \ldots \Box_{k_n} q_j, a_i)$ into a single clause of the form

$$\neg a_i R_{k_1} x_1 \vee \bigvee_{l=1}^{n-1} \neg x_l R_{k_l} x_{l+1} \vee Q_j(x_n),$$

for $n \geq 0$. In the case of $n = 0$, the clause we obtain consists of a single positive ground literal $Q_j(a_i)$. Besides clauses of this form, N_p can only contain the empty clause, which is the result of translation of the propositional constant $\bot$, while the translation of $\top$ will be eliminated during clausification.

Thus, every clause in N_p contains at most one predicate symbol Q_j. Moreover, all clauses in N_p will only contain positive occurrences of unary predicate symbols Q_j. In contrast, by definition, all occurrences of propositional variables q_j in the negative formulae in $A \wedge \neg P$ are negative. So, the corresponding occurrences of unary predicate symbols Q_j in N_n are all negative as well.

We have to establish the following.

The derivation always terminates for the formulae (clauses) under consideration. We define a function μ_1 that assigns to each clause C a triple $\mu_1(C) = \langle n_C^Q, n_C^R, d_C \rangle$ of natural numbers such that n_C^Q is the number of Q_i-literals in C, n_C^R is the number of all remaining literals in C, and d_C is the depth of C. We call $\mu_1(C)$ the *complexity* of clause C. It is straightforward to show that for a given triple $c = \langle n^Q, n^R, d \rangle$ of natural numbers the preimage of c under μ_1 contains only finitely many clauses (up to renaming of variables). We also define an ordering $\succ$ on $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ by the lexicographic combination of the ordering $>$ on the natural numbers with itself. Obviously, the ordering $\succ$ is well-founded.

We have already established that no clause in N contains a positive Q_i-literal as well as a negative Q_j-literal and the clauses in N_p have the property that each clause which contains a positive Q_i-literal contains exactly one such literal. It follows that in any C-resolution derivation from N no inference steps by C-factoring are possible. Furthermore, any C-resolvent D obtained by C-resolution with positive premise C_p and negative premise C_n will not contain a positive Q_i-literal, and D contains less Q_i-literals than C_n. Thus, $\mu_1(C_n) \succ \mu_1(D)$. Since no other inference steps are allowed in the C-resolution calculus, we have established that the conclusion of any inference step in a derivation from N is of strictly smaller complexity than one of its premises. The application of a deletion rule will only replace a clause C of complexity $\mu_1(C)$ be a clause D with smaller complexity $\mu_1(D)$. It follows that any derivation from N terminates.

The restoration of the quantifiers does not fail. To ensure that the restoration of quantifiers does not fail once the derivation from N has terminated, we can show by induction that for any clause C in a derivation from N, (i) C contains only inequations of the form

$$b \not\approx z, \quad b \not\approx c, \quad b \not\approx f(z), \quad y \not\approx z, \quad y \not\approx c, \quad \text{or} \quad y \not\approx f(z),$$

(ii) there are no two inequations in C of the form $y \not\approx f(z)$ and $y \not\approx g(z)$ with $f \neq g$, and (iii) if C contains negative Q_i-literals then these are of the form $\neg Q_i(z)$, $\neg Q_i(c)$ or $\neg Q_i(f(z))$, where c is a Skolem constant and f a unary Skolem function. An alternative formulation of property (ii) is that for any two inequations $x_1 \not\approx f(y)$ and $x_2 \not\approx g(z)$ in C with $f \neq g$ we have $x_1 \neq x_2$.

Inspection of the unskolemization procedure defined in [6] shows that properties (i) and (ii) are sufficient to ensure that unskolemization is successful, property (iii) enables us to show that the other two properties are preserved in inference steps. □

The theorem can be extended to the case of basic Sahlqvist formulae, obtained from definite Sahlqvist implications by applying boxes and disjunctions to formulae not sharing predicate variables.

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

(1) for every i, $1 \leq i \leq n-1$, either $\neg t_i R_{k_i} t_{i+1}$ or $t_i R_{k_i} t_{i+1}$ is in C;
(2) for every $(\neg)uR_{k_i}v \in C$, $u = t_j$ 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 $(t_1, \dots, t_n)$. Then there does not exist an ordered sequence $(s_1, \dots, s_m)$ of pairwise distinct terms which is distinct from $(t_1, \dots, t_n)$ such that C is also a chain over $(s_1, \dots, s_m)$.*

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

The link between the clauses we obtain from translating Sahlqvist formulae or modal formulae, in general, and chains is not as straightforward as one may hope. For example, consider the Sahlqvist antecedent $\neg q_1 \vee \Box q_2 \vee \Box q_3$. The clausal form of its translation consists of the single clause

$$\neg Q_1(a) \vee \neg a\ Rx \vee Q_2(x) \vee \neg a\ Ry \vee Q_3(y).$$

It is straightforward to check that we cannot arrange the terms a, x, and y in an ordered sequence S such that the whole clause would be a chain over S. Instead the clause consists of at least two chains: $\neg Q_1(a) \vee \neg a\ Rx \vee Q_2(x)$ over (a, x) and $\neg a\ Ry \vee \neg Q_3(y)$ over (a, y), or alternatively, $\neg a\ Rx \vee Q_2(x)$ over (a, x) and $\neg Q_1(a) \vee \neg a\ Ry \vee \neg Q_3(y)$ over (a, y). However, we could also divide the clause into three or more chains, for example, $\neg Q_1(a)$ over (a), $Q_2(x)$ over (x), $Q_3(y)$ over (y), $\neg a\ Rx$ over (a, x) and $\neg a\ Ry$ over (a, y).

In the following we will only consider *maximal chains*. A chain C over $(t_1, \dots, t_n)$ is *maximal* with respect to a clause D iff C is a variable indecomposable subclause of D and there is no chain C' over $(s_1, \dots, s_m)$, $m > n$, such that C' is a subclause of D and for every i, $1 \leq i \leq n$, $t_i \in \{s_1, \dots, s_m\}$.

Under this definition our example clause can only be partitioned into three maximal chains, namely, $\neg Q_1(a)$ over (a), $\neg a\ Rx \vee Q_2(x)$ over (a, x), and $\neg a\ Ry \vee Q_3(y)$ over (a, y). We can see the obvious link between the modal subformula of $\neg q_1 \vee \Box q_2 \vee \Box q_3$ and these three maximals. So, it makes sense to say that the first chain *is associated with* the negative formula $\neg q_1$, the second *is associated with* from the boxed atom $\Box q_2$, and the third from the boxed atom $\Box q_3$. In general, more than one maximal chain can be associated with a single negative formula, while exactly one maximal chain is associated with a boxed atom. We call a maximal chain which is associated with a boxed atom a *positive chain*, while all the chains associated with a negative formula are called *negative chains*.

It turns out that in the case of boxed atoms, the clauses we obtain have another important property: The clauses consist of a single maximal chain which is *rooted*. A chain C over $(t_1, \dots, t_n)$ is *rooted* iff t_1 is a ground term.

Lemma 2. *Let φ be a Sahlqvist implication. Then any clause C in $\mathrm{Cls}(\neg\Pi(\varphi))$ can be partitioned into a collection $\mathcal{D}$ of maximal chains. For any two maximal chains D and D' in $\mathcal{D}$, either D and D' are identical or they share at most one variable. In addition, if a maximal chain D in $\mathcal{D}$ is associated with a boxed atom in φ, then D is rooted and shares no variables with the other maximal chains in $\mathcal{D}$.*

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

Proof (Sketch). Let $\varphi = A \rightarrow P$. We know that $\mathrm{Cls}(\neg\Pi(\varphi)) = \mathrm{Cls}(\mathrm{ST}(A, a)) \cup \mathrm{Cls}(\mathrm{ST}(\neg P, a))$, where a is a Skolem constant. We also know that all clauses in $N_0 = \mathrm{Cls}(\neg\Pi(\varphi))$ satisfy the conditions stated in Lemma 2, in particular, any clause C in $\mathrm{Cls}(\neg\Pi(\varphi))$ can be partitioned into a collection $\mathcal{D}$ of maximal chains.

We introduce some additional notation for chains. We know that a chain associated with a boxed atom contains exactly one positive **Q**-literal $Q_i(t_i)$, where t_i is either a Skolem constant or a variable, and in the following we denote such a chain by $C^+[Q_i(t_i)]$ or $C_j^+[Q_i(t_i)]$. Chains associated with a negative formula may contain one or more negative **Q**-literals $\neg Q_1(t_1), \dots, \neg Q_n(t_n)$, where each t_i is either a Skolem constant, a variable, or a Skolem term of the form $f(x)$. We denote these chains by $C^-[\neg Q_1(t_1),\ \dots, \neg Q_n(t_n)]$ or $C_j^-[\neg Q_1(t_1),\ \dots, \neg Q_n(t_n)]$. By $C^+[\top]$ we denote the clause we obtain by removing the **Q**-literal $Q_i(t_i)$ from the chain $C^+[Q_i(t_i)]$. Analogously, $C^-[\top,\ \dots, \neg Q_n(t_n)]$ denotes the clause obtained by removing $\neg Q_1(t_i)$ from the chain $C^-[\neg Q_1(t_1),\ \dots, \neg Q_n(t_n)]$.

Unlike in the case of definite Sahlqvist implications, since a clause in N_0 can contain more than one positive **Q**-literal, inference steps by C-factoring are possible. Such an inference step would derive a clause $D_1 \vee C_1^+[Q(t_1)] \vee C_2^+[\top] \vee t_1 \not\approx t_2$ from a clause $D_1 \vee C_1^+[Q(t_1)] \vee C_2^+[Q(t_2)]$. Since t_1 and t_2 are either variables or constants, the constraint $t_1 \not\approx t_2$ can take the forms

$$b \not\approx z, \quad b \not\approx c, \quad y \not\approx z, \quad \text{or} \quad y \not\approx c.$$

In all cases, except where b and c are distinct constants, a most general unifier σ of t_1 and t_2 exists, and constraint elimination replaces $D_1 \vee C_1^+[Q(t_1)] \vee C_2^+[\top] \vee t_1 \not\approx t_2$ by $(D_1 \vee C_1^+[Q(t_1)] \vee C_2^+[\top])\sigma$. Note that this clause is identical to $D_1 \vee (C_1^+[Q(t_1)] \vee C_2^+[\top])\sigma$, that is, the subclause D_1 is not affected by the inference step nor does it influence the result of the inference step. A problem occurs in the following situation: The clause $Q(a) \vee \neg R(a, x) \vee Q(x)$ is a chain over (a, x) and a C-factoring step is possible which derives $Q(a) \vee \neg R(a, x) \vee a \not\approx x$. This C-factor is simplified by constraint elimination to $Q(a) \vee \neg R(a, a)$. However, an **R**-literal like $\neg R(a, a)$ with two identical arguments is not allowed in a chain. We could modify the definition of a chain to allow for these literals, but it is simpler to consider a clause like $Q(a) \vee \neg R(a, a)$ as shorthand for $Q(a) \vee \neg R(a, x) \vee a \not\approx x$.

It is important to note that the condition that a maximal chain associated with a boxed atom does not share any variables with other chains may no longer be true for C-factors. For example, consider the clause $\neg R(a, x) \vee$

$Q(x) \vee \neg R(a,u) \vee \neg R(u,v) \vee Q(v)$, obtained from $\neg \Pi(\Box q \vee \Box\Box q \rightarrow \top)$, which can be partitioned into two maximal chains, $\neg R(a,x) \vee Q(x)$ over (a,x) and $\neg R(a,u) \vee \neg R(u,v) \vee Q(v)$ over (a,u,v). This clause has the C-factor $\neg R(a,x) \vee Q(x) \vee \neg R(a,u) \vee \neg R(u,v) \vee v \not\approx x$. Constraint elimination replaces this C-factor by $\neg R(a,x) \vee Q(x) \vee \neg R(a,u) \vee \neg R(u,x)$ which can be partitioned into two maximal chains $\neg R(a,x) \vee Q(x)$ over (a,x) and $\neg R(a,u) \vee \neg R(u,x)$ over (a,u,x) that share the variable x. Let us call such clauses *factored positive chains.*

Note that the length of chains in a C-factor is the length of chains in the premise clause. Also, the depth of terms in a C-factor is the same as in the premise clause. Let there be c_p positive chains in the clauses of N_0. Then we can potentially derive $2^{c_p} - 1$ factored positive chains.

A C-resolvent can only be derived from a clause $D_1 \vee C^+[Q_i(t_i)]$ and a clause $D_2 \vee C^-[\neg Q'_1(t'_1), \ \dots, \neg Q'_n(t'_n)]$ where one of the Q'_j, $1 \leq j \leq n$, is identical to Q_i. Without loss of generality, we assume that $Q_i = Q'_1$. Then the resolvent is $D_1 \vee C^+[\top] \vee D_2 \vee C^-[\top, \neg Q'_2(t'_2), \ \dots, \neg Q'_n(t'_n)] \vee t_i \not\approx t'_1$. The term t_i will either be a Skolem constant b or a variable y, while the term t'_1 can either be a Skolem constant c, a variable z or a Skolem term $f(z)$. Thus, $t_i \not\approx t'_1$ has one of the following forms:

$$b \not\approx z, \quad b \not\approx c, \quad b \not\approx f(z), \quad y \not\approx z, \quad y \not\approx c, \quad \text{or} \quad y \not\approx f(z),$$

If t_i and t'_1 are unifiable by a most general unifier σ, then constraint elimination replaces the C-resolvent by $(D_1 \vee C^+[\top] \vee D_2 \vee C^-[\top, \neg Q'_2(t'_2), \ \dots, \neg Q'_n(t'_n)])\sigma$, which is identical to $D_1 \vee D_2 \vee C^+[\top] \vee (C^-[\top, \neg Q'_2(t'_2), \ \dots, \neg Q'_n(t'_n)])\sigma$. If t_i is a variable, then the t_i and t'_1 must be unifiable, since t_i cannot occur in t'_1. Furthermore, if t_i and t'_1 are unifiable, then the most general unifier is either the identity substitution or a substitution replacing t_i by t'_1. However, if t_i and t'_1 are not unifiable, then the constraint $t_i \not\approx t'_1$ cannot be eliminated. In this case t_i must be a Skolem constant b and t'_1 is either a Skolem constant c distinct from b or a Skolem term $f(z)$. Again, no terms deeper than terms in N_0 occur.

We will focus on the union of negative chains that occur within a single clause and share variables. We call these *joined negative chains.* Joined negative chains are variable indecomposable subclauses of the clauses in which they occur and they are variable-disjoint from the rest of the clause. A C-resolution inference step involves one such joined negative chain and one factored positive chain, and the result is again a joined negative chain. Let c_n be the number of joined negative chains in N_0 and let there be at most n_Q occurrences of $\mathbf{Q}$-literals in any joined negative chain. Then at most $c_n \times n_Q \times 2^{c_p}$ joined negative chains can be derived by one or more C-resolution inference steps.

Each clause in a derivation $N_0, N_1, \dots$ is a collection of factored positive chains and joined negative chains without duplicates modulo variable renaming. Any clause containing duplicates modulo variable renaming would immediately be replaced by its condensation. Given there are $2^{c_p} - 1$ factored positive chains and $c_n \times n_Q \times 2^{c_p}$ joined negative chains. We can derive at most $2^{c_n \times n_Q \times 2^{c_p} + 2^{c_p} - 1}$ distinct clauses. This ensures the termination of the derivation.

As to reverse Skolemization, again the Q_i-literals in N have one of the forms

$$Q_i(b), \quad Q_i(y), \quad \neg Q_i(z), \quad \neg Q_i(c) \quad \text{or} \quad \neg Q_i(f(z)),$$

where b denotes any Skolem constant, f denotes any Skolem function, and y and z arbitrary variables. The possible forms of inequality literals in all C-resolvents and C-factors are then

$$b \not\approx z, \quad b \not\approx c, \quad b \not\approx f(z), \quad y \not\approx z, \quad y \not\approx c, \quad \text{or} \quad y \not\approx f(z).$$

I.e. the form of inequality literals is $s \not\approx t$, where s and t are variable-disjoint, s is either a variable or a constant, and t is either a variable, a constant or a Skolem term of the form $f(x)$. What is again crucial is that no derived clause contains two inequations $y \not\approx s$ and $y \not\approx t$, where s and t are compound terms. This cannot happen. Consequently, restoration of the quantifiers can always be successfully accomplished during reverse Skolemization [6].

Finally, the general case of a Sahlqvist formula φ obtained from Sahlqvist implications by freely applying boxes and conjunctions, as well as disjunctions to formulae sharing no common propositional 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.*

5 Conclusion

Sahlqvist's theorem [16] can be seen as a milestone for a particular kind of classification problem in modal logics, namely, for which modal formulae can we establish correspondence and completeness results.

At first sight the work on SCAN seems to be unrelated to this classification problem, since SCAN tries to provide a general automated approach to establishing correspondence results while, as already noted, it is not complete for all modal formulae which have a first-order correspondent.

It is interesting to compare this situation to that in first-order logic. The classification problem in first-order logic is to identify fragments of first-order logic which are decidable/undecidable. It has been shown in recent years that general automated approaches to the satisfiability problem in first-order logic, in particular approaches based on the resolution principle, also provide decision procedures for many decidable fragments [7]. They can also provide a good starting point for establishing new decidable fragments of first-order logic.

In this paper we have applied similar approach to SCAN, showing that this general automated approach computes first-order correspondents for the class of Sahlqvist formulae. In fact, the class of modal formulae for which SCAN is successful is strictly larger than the class of Sahlqvist formulae (a witness being the formula given in the introduction). An important open problem is to find a purely logical characterization of that class.

Interestingly, since SCAN is based on a variation of the resolution principle, we were able to employ techniques used for establishing decidability results based on the resolution principle in first-order logic, to obtain our result.

It can be expected that corresponding results can also be obtained for other classes of modal formulae, and even classes for which until now no correspondence and completeness results have been established.

References

[1] W. Bibel and P. H. Schmitt, editors. *Automated Deduction – A Basis for Applications, Vol. I–III.* Kluwer, 1998. 150

[2] P. Blackburn, M. de Rijke, and V. Venema. *Modal Logic.* Cambridge Univ. Press, 2001. 150

[3] A. Chagrov and M. Zakharyaschev. *Modal Logic*, volume 35 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1997. 150

[4] M. de Rijke, and Y. Venema Sahlqvist's Theorem For Boolean Algebras with Operators with an Application to Cylindric Algebras, *Studia Logica*, 54:61–78, 1995. 149

[5] P. Doherty, W. Lukaszewics, and A. Szalas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997. 150

[6] T. Engel. Quantifier Elimination in Second-Order Predicate Logic. MSc thesis, Saarland University, Saarbrücken, Germany, 1996. 156, 160

[7] C. G. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution decision procedures. In *Handbook of Automated Reasoning*, pp. 1791–1849. Elsevier, 2001. 160

[8] D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7:35–43, 1992. 150, 151, 154

[9] V. Goranko and D. Vakarelov. Sahlqvist formulas unleashed in polyadic modal languages. In *Advances in Modal Logic*, vol. 3, pp. 221-240. World Scientific, 2002. 149, 151

[10] B. Jónsson. On the canonicity of Sahlqvist identities. *Studia Logica*, 53:473–491, 1994. 149

[11] M. Kracht. How completeness and correspondence theory got married. In *Diamonds and Defaults*, pp. 175–214. Kluwer, 1993. 149

[12] A. Nonnengart. Strong skolemization. Research Report MPI-I-96-2-010, Max-Planck-Institut für Informatik, Saarbrücken, 1996. 151

[13] A. Nonnengart, H. J. Ohlbach, and A. Szalas. Quantifier elimination for second-order predicate logic. In *Logic, Language and Reasoning: Essays in honour of Dov Gabbay*. Kluwer, 1999. 150

[14] A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In *Handbook of Automated Reasoning*, pp. 335–367. Elsevier, 2001. 151

[15] A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier Science, 2001. 150

[16] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In *Proc. of the 3rd Scandinavian Logic Symposium, 1973*, pp. 110–143. North-Holland, 1975. 149, 151, 160

[17] 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. 149

[18] A. Szałas. On the correspondence between modal and classical logic: An automated approach. *Journal of Logic and Computation*, 3(6):605–620, 1993. 150

[19] J. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, 1983. 149