Fundamenta Informaticae XX (2009) 1–37

IOS Press

Algorithmic correspondence and completeness in modal logic. III. Extensions of the algorithm SQEMA with substitutions

Willem Conradie
Department of Mathematics,
University of Johannesburg
Johannesburg, South Africa
wconradie@uj.ac.za

Valentin Goranko
Department of Informatics and Mathematical Modelling
Technical University of Denmark
vfgo@imm.dtu.dk

Dimiter Vakarelov
Faculty of Mathematics and Computer Science,
Sofia University
Sofia, Bulgaria
dvak@fmi.uni-sofia.bg

Abstract. In earlier papers we have introduced an algorithm, SQEMA, for computing first-order equivalents and proving canonicity of modal formulae. However, SQEMA is not complete with respect to the so called complex Sahlqvist formulae. In this paper we, first, introduce the class of complex inductive formulae, which extends both the class of complex Sahlqvist formulae and the class of polyadic inductive formulae, and second, extend SQEMA to $\mathsf{SQEMA}^{\mathrm{sub}}$ by allowing suitable substitutions in the process of transformation. We prove the correctness of $\mathsf{SQEMA}^{\mathrm{sub}}$ with respect to local equivalence of the input and output formulae and d-persistence of formulae on which the algorithm succeeds, and show that $\mathsf{SQEMA}^{\mathrm{sub}}$ is complete with respect to the class of complex inductive formulae.

Keywords: SQEMA, correspondence, d-persistence, complex Sahlqvist formulae.

Introduction

This paper is in the field of algorithmic correspondence and completeness theory in modal logic. The first general result in this field was the celebrated *Sahlqvist's theorem* [22]. It introduces a large class of modal formulae (subsequently called *Sahlqvist formulae*) which are first-order definable and canonical. Moreover, the proof of Sahlqvist's definability theorem, also obtained independently by van Benthem [34], provides an effective procedure, viz. the method of *minimal valuations*, for computing the first-order equivalents of the formulae in that class. For a long time the class of Sahlqvist formulae was considered as the optimal syntactically defined class with these two properties. In [9] the class of Sahlqvist formulae was extended to cover polyadic modal languages, but without extending the original Sahlqvist class on monadic languages. Recently, several new effective extensions or analogs of the Sahlqvist class have been obtained:

- the class of inductive formulae [15, 17, 5] for arbitrary polyadic modal languages,
- the class of inductive hybrid formulae, [16] (see also [25], [4]),
- the class of complex Sahlqvist formulae [26] (for the ordinary modal language).
- classes of formulae having equivalents in the first-order logic with least fix points [8, 13, 17, 20, 31, 32, 35, 36].

Because of the undecidability of the class of first-order definable modal formulae [3], the hierarchy of effective extensions of the Sahlqvist class concerning first-order definability is infinite and all further syntactic extensions are bound to be increasingly more complicated. In [6, 7] another approach has been proposed: instead of syntactic extensions of the Sahlqvist class, an algorithm, SQEMA (**S**econd-Order **Q**uantifier **E**limination for **M**odal formulae using **A**ckermann's lemma), was developed to compute first-order equivalents of modal formulae with unary modalities, further extended in [7] to polyadic and hybrid modal languages. It has been proved in [6, 7] that SQEMA is correct with respect to local equivalence of the input and output formulae, and that the formulae for which it succeeds are locally d-persistent (respectively, locally di-persistent for the case of languages with nominals and converse modalities), and hence canonical in the respective senses.

With respect to first-order correspondence, our approach was preceded and influenced by two earlier developed algorithms for the elimination of second-order quantifiers over predicate variables, viz. SCAN [12, 11] and DLS [10, 21, 23, 19]. Each of them, applied to the negation of the standard translation of a modal formula into monadic second-order logic, attempts to eliminate all occurring existentially quantified predicate variables and thus to compute a first-order correspondent. To that aim, SCAN employs a modification of the resolution method, while DLS is based on a result by Ackermann [1] (see also the references above, as well as [6, 7]), allowing explicit elimination, up to logical equivalence, of an existentially quantified second-order predicate variable.

Let us note that both SCAN and DLS use Skolemization of the input and, after the quantifier elimination procedure, a procedure attempting reverse Skolemization (de-Skolemization, or un-Skolemization) is applied. That procedure is not always successful, which may lead to (sometimes unnecessary) failure of the main algorithm. To avoid the necessity for de-Skolemization, SQEMA does not use the standard translation into the first-order logic but works directly on modal formulae and includes only a very restricted form of Skolemization, viz. only Skolem constants, introduced as nominals (an algorithm working directly with modal formulae was also considered in [24]). Thus, SQEMA attempts to eventually transform modal formulae into pure formulae in an appropriate hybrid modal language, from which the local first-order equivalent is extracted. In order to eliminate the propositional variables, SQEMA uses a modal version of Ackermann's lemma, formulated in terms of propositional modal logic, while the original lemma formulated by Ackermann and used in DLS is in terms of second-order logic. Further information about SCAN, DLS, and SQEMA can be found in the recent book on second-order quantifier elimination [13].

An implementation of a variant of SQEMA for monadic languages extended with nominals and universal modality has been realized by Dimiter Georgiev (see [14]) as a master project, and works online at `http://fmi.uni-sofia.bg/fmi/logic/sqema`.

The starting point of the present paper is the fact that none of the versions of SQEMA mentioned above is complete for the class of so called *complex Sahlqvist formulae* [26, 27]. This is an interesting phenomenon, because all complex Sahlqvist formulae can be effectively translated to Sahlqvist formulae for which all current versions of SQEMA succeed. The translation of complex Sahlqvist formulae into Sahlqvist formulae was constructed in [26] by means of quite complex reversible Boolean substitutions (preserving local first-order equivalents and d-persistence), effectively computed from the input complex formula. In the present paper we have extended SQEMA with a mechanism for applying such substitutions, which enables the new extension, denoted by $\mathsf{SQEMA}^{\mathrm{sub}}$, to succeed on all *complex inductive formulae* – a natural polyadic extension of the class of complex Sahlqvist formulae. We prove that all formulae for which $\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds are first-order definable and canonical, thus implying that this is the currently largest effective extension of the Sahlqvist class of first-order definable and canonical modal formulae.

The paper is organized as follows. Section 1 contains an informal introduction to polyadic modal logic, modal algebras over Kripke frames and the Ackermann lemma formulated in terms of modal algebras. It also provides an example of the latter lemma's application which illustrates the intuition upon which SQEMA is based. The section also contains the formal definition of SQEMA and the formulation of its basic meta-properties which will be used later on in the paper. In Section 2 we introduce the notion of reversible substitution and define two large classes of polyadic modal formulae: the class of complex recursive formulae which extends the class of regular formulae introduced in [17], and the class of complex polyadic inductive formulae, extending the class of polyadic inductive formulae [15, 17]. We also give an example of an inductive complex modal formula for which SQEMA does not succeed. Section 3 is devoted to the study of a special class of so called complex substitutions, on which $\mathsf{SQEMA}^{\mathrm{sub}}$ is based. Section 4 is devoted to an effective translation Θ of the class of complex inductive formulae into the class of inductive formulae by means of special type of reversible Boolean substitutions. This implies a generalization of the Sahlqvist Theorem both on its definability and canonicity part to the class of inductive complex modal formulae. Section 5 is preparatory for the definition of $\mathsf{SQEMA}^{\mathrm{sub}}$. Here we introduce the notion of complex normal form and a special translation Σ which is the main tool in $\mathsf{SQEMA}^{\mathrm{sub}}$. Section 6 is devoted to the definition of $\mathsf{SQEMA}^{\mathrm{sub}}$. We first discuss $\mathsf{SQEMA}^{\mathrm{sub}}$ informally, motivating its internal structure, which contains as a subprogram the former algorithm SQEMA and a new block SUB performing some transformations based on reversible Boolean substitutions. We illustrate the algorithm with some examples for which it succeeds but for which SQEMA does not succeed. We prove correctness and canonicity of $\mathsf{SQEMA}^{\mathrm{sub}}$ and its completeness with respect to the class of complex inductive formulae. We conclude in section 7 where we also mention some open problems and future research agenda.

1. Background on polyadic modalities and the algorithm SQEMA

The version of the algorithm SQEMA introduced in [7] is designed to work on polyadic modal formulae. Since the aim of the present paper is to introduce an extension of this algorithm, we invite the reader to consult [7] as well as [2, 15, 17] for all formal definitions and motivating examples concerning polyadic modal languages and inductive modal formulae. In this section we give an informal introduction to polyadic modal logic, fix some notation, and provide some intuitions underlying the algorithm SQEMA and the main results of the paper.

1.1. Polyadic modal logics

Standard polymodal propositional modal languages contain only unary modalities. With each class Σ of relational structures containing only binary relations we may associate such a language, $\mathcal{L}(\Sigma)$, with the modalities interpreted in Σ using the corresponding relations in the structures. One way to extend this parallelism to arbitrary relational structures is to use modal operators with arbitrary arity, called polyadic. Extending some notations from dynamic logic, we present standard polyadic modalities in the form $[\alpha](A_1, \ldots, A_n)$ (generalizing the box modality $[\alpha]A$) and $\langle\alpha\rangle(A_1, \ldots, A_n)$ (generalizing the diamond modality $\langle\alpha\rangle A$). Here α is called a modal term of arity n (notation $\rho(\alpha) = n$, where ρ is an **arity function**) and in the semantics of $[\alpha](A_1, \ldots, A_n)$ and $\langle\alpha\rangle(A_1, \ldots, A_n)$ this term is associated to a certain $n+1$-ary relation $R_\alpha(w, w_1, \ldots, w_n)$. Using the standard notation for the satisfiability relation in modal logic (see for instance [2]) we express the semantics of polyadic modalities as follows:

$(\mathcal{M}, w) \Vdash \langle\alpha\rangle(A_1, \ldots, A_n)$ iff there exist $w_1, \ldots, w_n$ such that $R_\alpha(w, w_1, \ldots, w_n)$ and $(\mathcal{M}, w_i) \Vdash A_i$, for each $1 \leq i \leq n$,

$(\mathcal{M}, w) \Vdash [\alpha](A_1, \ldots, A_n)$ if, for all $w_1, \ldots, w_n$ such that $R_\alpha(w, w_1, \ldots, w_n)$, it is the case that $(\mathcal{M}, w_i) \Vdash A_i$, for some $1 \leq i \leq n$.

Obviously, if $n = 1$ then this semantics coincides with the standard Kripke semantics for the unary modalities. The above semantics shows that the modality $[\alpha](A_1, \ldots, A_n)$ is dual to the modality $\langle\alpha\rangle(A_1, \ldots, A_n)$ and the following equivalences are valid which obviously generalize the corresponding equivalences for the unary case:

$[\alpha](A_1, \ldots, A_n) \leftrightarrow \neg\langle\alpha\rangle(\neg A_1, \ldots, \neg A_n)$ and

$\langle\alpha\rangle(A_1, \ldots, A_n) \leftrightarrow \neg[\alpha](\neg A_1, \ldots, \neg A_n)$,

Note that the case $n = 0$ is also included and in this case the two modal operators $\langle\alpha\rangle$ and $[\alpha]$ have no arguments and are treated as constants and the corresponding relation R_α is an unary relation, i.e., a subset of the universe of the model $\mathcal{M}$. The semantics of these constants is the following:

$(\mathcal{M}, w) \Vdash \langle\alpha\rangle$ iff $w \in R_\alpha$,

$(\mathcal{M}, w) \Vdash [\alpha]$ iff $w \notin R_\alpha$.

Let us denote by ι_n the modal term which in any model has the following interpretation as $n+1$-ary identity: $R_{\iota_n}(w, w_1, \ldots, w_n)$ iff $w = w_1 = \ldots = w_n$. Then the modality $\langle\iota_n\rangle(A_1 \ldots, A_n)$ is semantically equivalent with the conjunction $A_1 \wedge \ldots \wedge A_n$, and the modality $[\iota_n](A_1 \ldots, A_n)$ is semantically equivalents with the disjunction $A_1 \vee \ldots \vee A_n$. This fact allows one to treat classical conjunctions and disjunctions as polyadic modalities which considerably simplifies the theory of polyadic modal logic (see [15, 17]).

As in dynamic logic, modal terms can be composed subject to some obvious arity constraints. We shall illustrate this construction with an example. Let α be a modal term of arity 2 ($\rho(\alpha) = 2$), let β, γ be modal terms of arbitrary arity, say $\rho(\beta) = 2$ and $\rho(\gamma) = 3$ and let $R_\alpha, R_\beta, R_\gamma$ be the corresponding relations in some model. Then we may define a new relation S by the following natural definition:

$S(x, x_1, x_2, x_3, x_4, x_5)$ iff there exist y_1, y_2 such that $R_\alpha(x, y_1, y_2)$, $R_\beta(y_1, x_1, x_2)$ and $R_\gamma(y_2, x_3, x_4, x_5)$.

With this construction in mind, it is natural to consider the relation S corresponding to the composed modal term $\alpha(\beta, \gamma)$, called the composition of α, β and γ in this order. The following equivalence is true for this composition:

$[\alpha(\beta, \gamma)](A_1, A_2, A_3, A_4, A_5) \leftrightarrow [\alpha]([\beta](A_1, A_2), [\gamma](A_3, A_4, A_5)$.

The above considerations show that we may have different modal languages depending on the set τ of modal terms with their predefined arity, called a **modal similarity type**. A modal similarity type τ and a set Θ of propositional variables together uniquely determine (by a simultaneous induction) the set of all (composed) terms MT_τ and the set of all formulae. This language is denoted by $\mathcal{L}_\tau(\Theta)$. If the particular set of proposition letters Θ over which the language is built is not important, we will omit it and simply write $\mathcal{L}_\tau$. We will always assume that modal languages contain the identity modal terms ι_n.

Similarity types are important in the formal definition of the semantics of a given modal language. Namely, given a type τ, we consider the class of τ-frames. These are relational structures of the form $\mathfrak{F} = (W, \{R_\alpha\})_{\alpha \in \tau}$, with R_α a $(\rho(\alpha) + 1)$-ary relation for each $\alpha \in \tau$.

As in dynamic logic with **inverse operations** α^{-1} on modal terms (also called converse operations), we may consider a generalization of this operation in polyadic modal logic. For the binary case we have the following condition between R_α and $R_{\alpha^{-1}}$: $R_{\alpha^{-1}}(x, y)$ iff $R_\alpha^{-1}(x, y)(=_{def} R_\alpha(y, x))$.

For the polyadic case, if $\rho(\alpha) = n$, then we have n inverses α^{-i}, $i = 1, \ldots, n$, with the following semantics: $R_{\alpha^{-i}}(x, y_1, \ldots, y_i, \ldots, y_n)$ iff $R_\alpha^{-i}(x, y_1 \ldots, y_i, \ldots, y_n)$ where $R_\alpha^{-i}(x, y_1 \ldots, y_i, \ldots, y_n)$ is defined as $R_\alpha(y_i, y_1 \ldots, x, \ldots, y_n))$, i.e., the first and $(i + 1)$st arguments are interchanged.

The following equivalence is always true for the inverse modalities, which generalize the unary case in the obvious way. Let $\mathcal{M}$ be any model in which R_α and $R_{\alpha^{-i}}$ are interpreted. Then $\mathcal{M} \Vdash B \vee [\alpha](A_1, \ldots, A_i, \ldots, A_n)$ iff $\mathcal{M} \Vdash [\alpha^{-i}](A_1, \ldots, B, \ldots, A_n) \vee A_i$, $i = 1, \ldots, n$.

The extension of the language $\mathcal{L}_\tau$ with inverse operations is denoted by $\mathcal{L}_{\tau(r)}$ and is called **completely reversive** extension of $\mathcal{L}_\tau$ ([7]).

We will consider also *hybrid modal languages* containing **nominals**, – special variables, true in exactly one point. The hybrid extensions of $\mathcal{L}_\tau$ and $\mathcal{L}_{\tau(r)}$ will be denoted by $\mathcal{L}_\tau^n$ and $\mathcal{L}_{\tau(r)}^n$ ([7]). Hybrid formulae which do not contain any propositional variables but only (possibly) nominals are called **pure formulae**. Hybrid languages are often extended with the **universal modality** corresponding to a special term U such that in the semantics $R_U = W \times W$, i.e., the largest relation in the frame $\mathfrak{F}$. A formula A in a hybrid language is called a **pure formula** if it does not contain propositional variables.

A well known natural translation of a formula A (in any of the above mentioned modal languages) into a first-order formula $ST(A, x)$ with only one free variable x can be defined, and we invite the reader to consult [2] or [7] for its definition. Let us note that the standard translation of a pure formula is a first-order condition, a fact which will be used in the final stage of algorithm SQEMA and its extension SQEMA$^{\text{sub}}$for obtaining the desired local first-order equivalent of the input formula.

Lastly some terminology relating to the various notions of equivalence for formulae. Two $\mathcal{L}_{\tau(r)}^n$-formulas φ and ψ are **semantically equivalent** (denoted $\varphi \equiv \psi$) if they are true at exactly the same states in all τ-models; **locally frame equivalent** if they are valid at exactly the same states in all τ-

frames; **locally equivalent** if they are valid at exactly the same states in all τ-general frames.

1.2. Modal algebras

For a given similarity type τ, let $\mathfrak{F} = (W, \{R_\alpha\}_{\alpha\in\tau})$ be a τ-frame. For any $(n+1)$-place relation R_α on W, not necessarily corresponding to a given modal τ-term, we define two n-ary operations over subsets $A_1, \ldots, A_n$ of W as follows: $\langle\alpha\rangle(A_1, \ldots, A_n) =_{def} \{x \in W : (\exists y_1 \ldots y_n \in W)(R_\alpha(x, y_1, \ldots, y_n)$ and $y_i \in A_i$ for all $1 \leq i \leq n\}$ and $[\alpha](A_1 \ldots, A_n) =_{def} \neg\langle\alpha\rangle(\neg A_1, \ldots \neg A_n)$.

Let $\mathbf{B}(\mathfrak{F})$ be the Boolean algebra of all subsets of W augmented with the operations $\langle\alpha\rangle$, $\alpha \in \tau$. We will use the standard notion for the logical operations of negation $\neg$, conjunction $\wedge$, and disjunction $\vee$, to denote the corresponding Boolean operations of complement, meet and join, and $0 = \varnothing, 1 = W$ will be respectively the zero and the unit of the algebra.

The algebra obtained in this way will be called **modal algebra over** $\mathfrak{F}$. Modal algebras may have richer signatures than the signature of the corresponding modal language. Namely, even if the modal language does not contain inverses, we allow the application of the operations $[\alpha^{-i}]$ and $\langle\alpha^{-i}\rangle$ to subsets of W, having in mind the natural assumption that $R_{\alpha^{-i}} = R_\alpha^{-i}$.

Modal algebras can be used to simplify some semantical definitions, when one regards modal τ-formulae as polynomials over a modal algebra $\mathbf{B}(\mathfrak{F})$ with propositional variables and nominals ranging over subsets and singleton subsets of W, respectively, and modal operations $[\alpha]$ and $\langle\alpha\rangle$ interpreted as the operations corresponding to the relation R_α. In this way modal formulae will denote subsets of W. Now validity of a formula A in $\mathfrak{F}$ is equivalent to the fact that the equation $A = 1$ is identically true in $\mathbf{B}(\mathfrak{F})$, i.e., $A = 1$ for all possible values of the variables and nominals occurring in A. Local validity at a point $x \in W$ is equivalent to the fact that $x \in A$ is true identically.

Let us note that the above treatment of modal formulae as algebraic expressions in modal algebras has some additional features, which will be used subsequently in the algorithm SQEMA and its extension $\mathsf{SQEMA}^{\mathrm{sub}}$. Namely, some modal expressions over modal algebras code, in some sense, local and global first-order conditions of the frame $\mathfrak{F}$. Let us explain this with some examples.

$x \in \{y\}$ means $x = y$.

Let R_α be a binary relation in W. Then:

$x \in \langle\alpha\rangle\{y\}$ means $R_\alpha(x, y)$

$x \in \langle\alpha^{-1}\rangle\{x\}$ means $R_\alpha(x, x)$ - local reflexivity of R_α at x.

$x \in [\alpha][\alpha]\langle\alpha^{-1}\rangle\{x\}$ means $(\forall y, z)(R_\alpha(x, y) \wedge R_\alpha(y, z) \rightarrow R_\alpha(x, z))$ - the local transitivity of R_α at x.

We will also use the following algebraic facts:

$A \rightarrow B = 1$ iff $A \subseteq B$,

$A \subseteq B$ iff $\neg A \vee B = 1$,

$x \in A$ iff $\{x\} \subseteq A$ iff $\neg\{x\} \vee A = 1$,

$x \notin A$ iff $\{x\} \in \neg A$ iff $\neg\{x\} \vee \neg A = 1$,

$A \vee [\alpha](B_1, \ldots, B_i, \ldots, B_n) = 1$ iff $[\alpha^{-i}](B_1, \ldots, A, \ldots, B_n) \vee B_i = 1$,

$x \in [\alpha](B_1, \ldots, B_i, \ldots, B_n)$ iff $[\alpha^{-i}](B_1 \ldots, \neg\{x\}, \ldots, B_n) \vee B_i = 1$.

$(\alpha^{-i})^{-i} = \alpha$.

1.3. Ackermann's lemma and SQEMA, informally

The main transformation rule of the algorithm SQEMA ([6, 7]) is the so called **Ackermann-Rule**, the details of which will be recalled in the next section. The Ackermann-Rule is based on the Ackermann lemma introduced by Ackermann in [1] for elimination of second-order quantifiers. We owe to A. Szałas [23] the idea to apply the Ackermann lemma (in its original formulation) in modal definability theory. In [6, 7] we used a modal version of the Ackermann lemma, whereas here we will give an algebraic version of this lemma. Similar algebraic treatment of Ackermann's lemma and its generalizations can be found also in [28] – [32]. The algebraic reformulation of this lemma, in standard mathematical language, can be seen as the statement of a kind of necessary and sufficient condition for a special system of equations in modal algebras to have a solution. This makes the lemma more readily understandable and illustrates the intuition behind SQEMA. Notwithstanding the extreme simplicity of its proof, this lemma is most fruitfully applicable.

Lemma 1.1. Modal Ackermann lemma: an algebraic form. Let $\mathbf{B}(\mathfrak{F})$ be a modal algebra over a given τ-frame $\mathfrak{F} = (W, R)$. Let A and $B(q)$ be modal formulae over $\mathbf{B}(W)$ such that A does not contain the variable p and $B(q)$ be a formula having only positive occurrences of the variable q. Consider the following system of equations with respect to p:

$$(*) \quad \left\| \begin{array}{lcl} A \vee p & = & 1 \\ B(\neg p) & = & 1. \end{array} \right.$$

Then $(*)$ has a solution for p in $\mathbf{B}(\mathfrak{F})$ iff $B(A) = 1$.

Proof:
$(\Rightarrow)$ Suppose that $(*)$ has a solution for p. Then $A \vee p = 1$, which is equivalent to $\neg p \subseteq A$. Since $B(q)$ has only positive occurrences of q, it is upward monotone with respect to q. Hence $B(\neg p) \subseteq B(A)$ and, since $B(\neg p) = 1$, $B(A) = 1$.
$(\Leftarrow)$ If $B(A) = 1$, then $p = \neg A$ is a solution of $(*)$. □

Now we will show how to apply Lemma 1.1 to obtain local first-order equivalents of modal formulae. As an example, consider the formula $[\alpha]p \to [\alpha][\alpha]p$. Let $\mathfrak{F} = (W, \{R_\alpha\}_{\alpha\in\tau})$ be a τ-frame and $x \in W$. The local condition at x for this formula is (see the previous section) $(\forall p \subseteq W)(x \in ([\alpha]p \to [\alpha][\alpha]p))$.

We will perform the following sequence of equivalent transformations of this condition and at the end we will obtain the desired first-order local equivalent.

(1) $(\forall p \subseteq W)(x \in ([\alpha]p \to [\alpha][\alpha]p))$ iff
(2) $\neg\neg(\forall p \subseteq W)(x \in [\alpha]p \to x \in [\alpha][\alpha]p)$ iff
(3) $\neg(\exists p \subseteq W)(x \in [\alpha]p$ and $x \notin [\alpha][\alpha]p)$ iff
(4) $\neg(\exists p \subseteq W)(\neg\{x\} \vee [\alpha]p = 1$ and $x \in \neg[\alpha][\alpha]p)$ iff
(5) $\neg(\exists p \subseteq W)([\alpha^{-1}]\neg\{x\} \vee p = 1$ and $\neg\{x\} \vee \neg[\alpha][\alpha]p = 1)$ iff
(6) $\neg(\exists p \subseteq W)([\alpha^{-1}]\neg\{x\} \vee p = 1$ and $\neg\{x\} \vee \neg[\alpha][\alpha]\neg\neg p = 1)$ iff (by Lemma 1.1, with $B(q) = \neg\{x\} \vee \neg[\alpha][\alpha]\neg q)$)
(7) $\neg(\neg\{x\} \vee \neg[\alpha][\alpha]\neg[\alpha^{-1}]\neg\{x\} = 1)$ iff
(8) $\neg(x \notin [\alpha][\alpha]\langle\alpha^{-1}\rangle\{x\})$ iff
(9) $x \in [\alpha][\alpha]\langle\alpha^{-1}\rangle\{x\}$ iff

(10) $(\forall y, z)(R_\alpha(x, y) \wedge R_\alpha(y, z) \to R_\alpha(x, z))$ — the local transitivity of R_α at x.

Note that from (2) to (6) we produce only transformations after the negation sign (**the first negation step**) which is needed in order to turn the universal sentence (1) into an existential form and then to prepare the equations for an application of the modal Ackermann lemma (in step (6)). In (8) we apply the **second negation step** and obtain the needed local condition in a "coded" modal form, which in (10) is "decoded" in its first-order format.

The formal versions of SQEMA performs all these steps following strictly defined syntactic formal transformation rules over some systems of "equations" which are analogs of the algebraic equations of the above informal example.

1.4. The algorithm SQEMA

This subsection recalls the high-level description of the algorithm SQEMA and its transformation rules, and also some of its meta-properties.

1.4.1. Description of SQEMA.

Here we present briefly the basic algorithm SQEMA for reader's convenience; for more detail see [6, 7].

First, some terminology — an expression of the form $\varphi \vee \psi$ with $\varphi, \psi \in \mathcal{L}^n_{\tau(r)}$ is called a **SQEMA-equation**. A finite set of SQEMA-equations is called a **SQEMA-system**. For a system Sys, we let $\mathsf{Form}(\mathsf{Sys})$ be the conjunction of all equations in Sys. Given a formula $\varphi \in \mathcal{L}_\tau$ as input, SQEMA processes it in three phases, with the goal to reduce φ first to a suitably equivalent pure, and then first-order formula.

Phase 1 (preprocessing) — The negation of φ is converted into negation normal form, and $\Diamond$ and $\wedge$ are distributed over $\vee$ as much as possible, by applying the equivalences $\Diamond(\psi \vee \gamma) \equiv \Diamond\psi \vee \Diamond\gamma$ and $\delta \wedge (\psi \vee \gamma) \equiv (\delta \wedge \psi) \vee (\delta \wedge \gamma)$. For each disjunct of the resulting formula $\bigvee \varphi'_i$ a system Sys_i is formed consisting of the single equation $\neg\mathbf{i} \vee \varphi'_i$, where $\mathbf{i}$ is a reserved nominal used to denote the state of evaluation in a model, and not allowed to occur in the input formula φ. These are the *initial systems* in the execution.

Phase 2 (elimination) — The algorithm now proceeds separately on each initial system, Sys_i, by applying to it the transformation rules listed below in section 1.4.2 (table 1). The aim is to eliminate from the system all occurring propositional variables. If this is possible for each system, we proceed to phase 3, else the algorithm report failure and terminates. The rules in table 1 are to be read as rewrite rules, i.e., they replace equations in systems with new equations or, in the case of the Ackermann-rule, systems with new systems. Note that each actual elimination of a variable is achieved through an application of the Ackermann-rule while the other rules are used to solve the system for the variable to be eliminated, i.e., to bring the system into the right form for the application of this rule.

Phase 3 (translation) — This phase is reached only if all systems have been reduced to pure systems, i.e., systems Sys_i with $\mathsf{Form}(\mathsf{Sys}_i)$ a pure formula. Let $\mathsf{Sys}_1, \dots, \mathsf{Sys}_n$ be these systems. Recalling that φ was the input to the algorithm, we will write $\mathsf{pure}(\varphi)$ for the formula $(\mathsf{Form}(\mathsf{Sys}_1) \vee \cdots \vee \mathsf{Form}(\mathsf{Sys}_n))$. The algorithm now computes and returns, as local frame correspondent for the input formula φ, the formula $\forall\overline{y}\exists x_0 \mathrm{ST}(\neg\mathsf{pure}(\varphi), x_0)$ where $\overline{y}$ is the tuple of all occurring variables corresponding to nominals, but with $y_\mathbf{i}$ (corresponding to the designated current state nominal $\mathbf{i}$) left free, since a local correspondent is being computed.

1.4.2. The transformation rules of SQEMA

Table 1 lists the transformation rules used by SQEMA. We have added the $\vee$-rule in order to simplify the Ackermann rule from [7] by enabling all equations of the type $A \vee p$ to be put together into one. Note that, for monadic modalities, the $\Box$ and $\Diamond$-rules simplify as follows:

$$\text{(Monadic } \Box\text{-rule)} \quad \frac{A \vee [\alpha]B}{[\alpha^{-1}]A \vee B} \qquad\qquad \text{(Monadic inverse } \Box\text{-rule)} \quad \frac{A \vee [\alpha^{-1}]B}{[\alpha]A \vee B}$$

$$\text{(Monadic } \Diamond\text{-rule)} \quad \frac{\neg \mathbf{j} \vee \langle\alpha\rangle A}{\neg \mathbf{j} \vee \langle\alpha\rangle \mathbf{k}, \ \neg \mathbf{k} \vee A}$$

where α is any unary modal term, and $\mathbf{k}$ is a fresh nominal not occurring in the premise. The algorithm can be strengthened further by adding more transformation rules facilitating some propositional reasoning, as is done in [6, 7].

1.4.3. Some meta-properties of SQEMA

A. Correctness

A formula on which SQEMA succeeds will be called a **SQEMA-formula**.

Theorem 1.1. (Correctness of SQEMA, [7])
Every SQEMA-formula is locally frame-correspondent to the first-order formula returned.

B. Canonicity

For a definition of descriptive frames see e.g., [7].

A formula φ is **locally d-persistent**, if, for every pointed descriptive frame $(\mathfrak{F}, w)$ for the respective language, it is the case that $(\mathfrak{F}_\sharp, w) \Vdash \varphi$ whenever $(\mathfrak{F}, w) \Vdash \varphi$; φ is **d-persistent** if $\mathfrak{F}_\sharp \Vdash \varphi$ whenever $\mathfrak{F} \Vdash \varphi$. Clearly, local d-persistence implies d-persistence.

Theorem 1.2. (D-persistence,[7])

1. Every SQEMA-formula in $\mathcal{L}_\tau$ is locally persistent with respect to the class of all descriptive τ-frames.

2. Every SQEMA-formula in $\mathcal{L}_{r(\tau)}$ is locally persistent with respect to the class of all reversive descriptive τ-frames.

Corollary 1.1. (Canonicity of SQEMA, [7])
All formulae on which SQEMA succeeds are canonical.

C. Completeness

For the definition of polyadic inductive formula we refer to the paper [17] (see also [15, 16, 7]. Later on in the paper these formulae will be discussed as special case of complex inductive formulae.

Table 1. SQEMA Transformation Rules

Rules for connectives

$$\frac{C \vee (A \wedge B)}{C \vee A, C \vee B} \quad (\wedge\text{-rule}) \qquad \frac{A \vee C, B \vee C}{(A \wedge B) \vee C} \quad (\vee\text{-rule})$$

$$\frac{C \vee (A \vee B)}{(C \vee A) \vee B} \quad (\text{left-shift } \vee\text{-rule}) \qquad \frac{(C \vee A) \vee B}{C \vee (A \vee B)} \quad (\text{right-shift } \vee\text{-rule})$$

$$\frac{A \vee [\gamma](B_1, \ldots, B_n)}{[\gamma^{-i}](B_1, \ldots, B_{i-1}, A, B_{i+1}, \ldots, B_n) \vee B_i} \quad (\Box\text{-rule})$$

$$\frac{A \vee [\gamma^{-i}](B_1, \ldots, B_n)}{[\gamma](B_1, \ldots, B_{i-1}, A, B_{i+1}, \ldots, B_n) \vee B_i} \quad (\text{inverse } \Box\text{-rule})$$

$$\frac{\neg \mathbf{j} \vee \langle \gamma \rangle (A_1, \ldots, A_n)}{\neg \mathbf{j} \vee \langle \gamma \rangle (\mathbf{k}_1, \ldots, \mathbf{k}_n),\ \neg \mathbf{k}_1 \vee A_1, \ldots, \neg \mathbf{k}_n \vee A_n} \quad (\Diamond\text{-rule}^*)$$

$*$where the $\mathbf{k}_i$ are new nominals not occurring in the system.

Polarity switching rule

Substitute $\neg p$ for every occurrence of p in the system.

Ackermann-rule

$$\text{The system} \quad \left\|\begin{array}{l} A \vee p \\ B_1(p) \\ \cdots \\ B_m(p) \\ C_1 \\ \cdots \\ C_k \end{array}\right. \quad \text{is replaced by} \quad \left\|\begin{array}{l} B_1(A/\neg p) \\ \vdots \\ B_m(A/\neg p) \\ \\ C_1 \\ \cdots \\ C_k \end{array}\right.$$

where:

1. p does not occur in $A, C_1, \ldots, C_k$;
2. $\mathsf{Form}(B_1) \wedge \cdots \wedge \mathsf{Form}(B_m)$ is negative in p.
3. $B_i(A/\neg p)$ means that $\neg p$ in B_i is replaced by A.

Theorem 1.3. (Completeness of SQEMA w.r.t. inductive formulae, [7])
SQEMA succeeds on all conjunctions of polyadic inductive formulae.

Examples of how SQEMA works on different formulae, including polyadic inductive formulae, can be found in [7]. Later we will present an example of inductive complex modal formula (to be defined in the next section) on which SQEMA fails.

2. Inductive complex modal formulae and reversible substitutions

In this section we introduce two large classes of modal formulae: the **recursive complex modal formulae (RCM-formulae)** and their subclass of **inductive complex modal formulae (ICM-formulae)**.

The class of the ICM-formulae was introduced in [27] under the name 'complex polyadic Sahlqvist formulae'. It simultaneously extends both the class of inductive formulae and the class of complex Sahlqvist formulae [26]. The adjective 'complex' comes from the fact that these formulae are built over some special Boolean formulae, called 'complex variables' in [26]. All ICM-formulae are first-order definable and canonical, but the current version of SQEMA does not succeed on all of them (see examples 2.2 and 2.3, below). One of our main objectives in this paper is to extend SQEMA with an additional module which performs special substitutions which enables it to succeeds on all ICM-formulae.

2.1. Substitutions

We adopt the standard definition of **(uniform) substitution** ([2]) as a mapping in the set of formulae acting on them homomorphically. This means that a substitution S can be defined if we first specify it on propositional variables and then extend it by induction for arbitrary formulae as follows: $S(\neg A) = \neg S(A)$, $S(A \circ B) = S(A) \circ S(B)$ where $\circ$ is any binary Boolean connective, and $S[\alpha](A_1, \dots, A_n) = [\alpha](S(A_1), \dots, S(A_n))$

We will usually denote substitutions by S, T. Sometimes we will be interested in substitutions acting on a fixed set of propositional variables. In such a case we assume that they act on all other variables identically. The following observation is immediate.

Fact 2.1. Local frame validity is preserved by uniform substitutions and by modus ponens.

If $A, B(p) \in \mathcal{L}^n_{r(\tau)}$ we will write $B(A/p)$, or simply $B(A)$, for the formula obtained from $B(p)$ by uniform substitution of A for all occurrences of p.

Definition 2.1. Let $\mathbf{p} = \langle p_1, \dots, p_n \rangle$ and $\mathbf{q} = \langle q_1, \dots, q_m \rangle$ be two disjoint lists of different propositional variables, and S a substitution which maps the variables in $\mathbf{p}$ to formulae built over $\mathbf{q}$, and acts identically on variables not in $\mathbf{p}$. We say that S is a **reversible substitution** if there is a substitution T that maps the variables in $\mathbf{q}$ to formulae built over $\mathbf{p}$, acts on variables not in $\mathbf{q}$ identically, and is such that $T(S(p_i)) \equiv p_i$, for $i = 1, \dots, n$ (and, consequently, $T(S(A)) \equiv A$ for any formula A containing only propositional variables in $\mathbf{p}$). We then say that $\mathbf{p}$ is the **domain** of S, $\mathbf{q}$ is a **co-domain** of S, and T **reverses** S.

Note that if a substitution T reverses a substitution S, then T need not be reversible itself, because of its action on variables not in the range of S. The following lemma follows immediately from fact 2.1 and the definition of a reversible substitution.

Lemma 2.1. Let S be a reversible substitution with a domain $\mathbf{p}$ and a co-domain $\mathbf{q}$. Then A is locally equivalent to $S(A)$ for every formula A.

Clearly, the requirement for S to be a reversible substitution is essential, e.g., consider S such that $S(p) = q \vee \neg q$ and take $A = p$.

A simple example of reversible substitutions, as employed by SQEMA, is **polarity change**: $S(p) := \neg p$. Non-trivial examples of reversible Boolean substitutions can be found in [26, 29]; more such examples are provided further in the paper.

2.2. Substitutions producing inductive formulae: an informal discussion

Since the definitions of recursive and inductive complex modal formulae are complicated, we will begin with some concrete motivating examples. For simplicity we will start with an example in the basic mono-modal language with the usual box and diamond modalities $\Box$ and $\Diamond$. Consider the formula

$$A = \Diamond\Box(p_1 \vee p_2) \wedge \Diamond\Box(p_1 \vee \neg p_2) \wedge \Diamond\Box(\neg p_1 \vee p_2) \to \Box\Diamond(p_1 \wedge p_2).$$

It is not a Sahlqvist formula, nor even an inductive one, and the standard Sahlqvist-van Benthem substitution method does not work on it. However, note that $p_1 \wedge p_2 \equiv (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2)$, hence A can be obtained, up to local equivalence, from the formula

$$A' = \Diamond\Box q_1 \wedge \Diamond\Box q_2 \wedge \Diamond\Box q_3 \to \Box\Diamond(q_1 \wedge q_2 \wedge q_3).$$

by applying the substitution:

$$\begin{aligned} T(q_1) &= p_1 \vee p_2, \\ T(q_2) &= p_1 \vee \neg p_2, \\ T(q_3) &= \neg p_1 \vee p_2. \end{aligned}$$

The formula A' is a Sahlqvist formula and it locally corresponds to the following Church-Rosser-like first-order property of a binary relation R:

$$xRy_1 \wedge xRy_2 \wedge xRy_3 \wedge xRy_4 \to (\exists z)(y_1Rz \wedge y_2Rz \wedge y_3Rz \wedge y_4Rz). \qquad (\mathrm{CR}_4)$$

Thus, A is a local consequence from A'. Conversely, A' can be obtained, up to local equivalence, from A by means of the following substitution:

$$\begin{aligned} S(p_1) &= q_1 \wedge q_2, \\ S(p_2) &= (q_1 \wedge \neg q_2) \vee (q_1 \wedge q_3). \end{aligned}$$

After simple Boolean transformations one can obtain the formula

$$A'' = \Diamond\Box q_1 \wedge \Diamond\Box(\neg q_1 \vee q_2) \wedge \Diamond\Box(\neg q_1 \vee \neg q_2 \vee q_3) \to \Box\Diamond(q_1 \wedge q_2 \wedge q_3).$$

Using the valid implications $q_2 \to \neg q_1 \vee q_2$ and $q_3 \to \neg q_1 \vee \neg q_2 \vee q_3$, and the monotonicity of $\Box$ and $\Diamond$, one can then easily obtain A' as a local consequence from A''.

Thus, the two formulae A and A' are locally equivalent. In particular, the formula A locally corresponds to the first-order formula (CR_4), too [1]. Furthermore, note that the substitution T reverses the substitution S: $T(S(p_i)) \equiv p_i$, for every variable p_i in the domain of S.

In order to see the general pattern of transformation between A and A', let us denote $D_1 = p_1 \vee p_2$, $D_2 = p_1 \vee \neg p_2$ and $D_3 = \neg p_1 \vee p_2$. Then A can be presented in the following way:

$$A = \Diamond\Box D_1 \wedge \Diamond\Box D_2 \wedge \Diamond\Box D_3 \to \Box\Diamond(D_1 \wedge D_2 \wedge D_3).$$

Notationally, A and A' look quite similar, the only difference being that the elementary disjunctions D_1, D_2 and D_3 in A replace the variables q_1, q_2 and q_3 in A'. In [26] such elementary disjunctions are called **complex variables**, because they code in some way ordinary variables, and the respective extension of the Sahlqvist class defined in [26] — **complex Sahlqvist formulae**. It is not, however, true in general that complex formulae can be obtained from Sahlqvist formulae simply by replacing their different variables by different elementary disjunctions as in the above example, because non-first-order definable modal formulae can be obtained in such a way, too. Consider, for instance, the Sahlqvist formula $\Diamond\Box q_1 \to \Diamond\Box(q_1 \wedge q_2) \vee \Diamond\Box(q_1 \wedge q_3)$ and replace q_1, q_2, q_3 by D_1, D_2, D_3, respectively. After some Boolean simplifications we obtain the formula $\Diamond\Box(p_1 \vee p_2) \to \Diamond\Box p_1 \vee \Diamond\Box p_2$, which is not first-order definable [33].

The next example of complex formula is in a polyadic language, where α, β, and γ are modal terms of suitable arities:

$$B = [\alpha](\neg[\beta](\neg p_1 \vee p_2), \neg[\beta](p_1 \vee \neg p_2), \neg[\beta](p_1 \vee p_2), \langle\gamma\rangle((\neg p_1 \vee p_2), (p_1 \leftrightarrow p_2), (p_1 \wedge p_2))).$$

Modulo semantic equivalences the formula B can be rewritten, using the elementary disjunctions D_1, D_2, D_3, as:

$$B = [\alpha](\neg[\beta]D_3, \neg[\beta]D_2, \neg[\beta]D_1, \langle\gamma\rangle(D_3, D_3 \wedge D_2, D_3 \wedge D_2 \wedge D_1)).$$

That formula can be obtained by an obvious substitution from

$$B' = [\alpha](\neg[\beta]q_1, \neg[\beta]q_2, \neg[\beta]q_3, \langle\gamma\rangle(q_1, q_1 \wedge q_2, q_1 \wedge q_2 \wedge q_3)).$$

Here B is a complex formula and B' is an inductive formula with headed boxes $[\beta]q_1$, $[\beta]q_2$ and $[\beta]q_3$ and a negated headless box (positive part) $\langle\gamma\rangle(q_2, q_2 \wedge q_3, q_2 \wedge q_3 \wedge q_1)$. Note that B and B' are again related by means of a pair of reversible substitutions S' and T', where:

$$T'(q_1) = D_3,\ T'(q_2) = D_2,\ T'(q_3) = D_1.$$

$$S'(p_1) = \neg q_1 \vee (q_2 \wedge q_3),\ S'(p_2) = q_1 \wedge (\neg q_2 \vee q_3).$$

From the example above one can see that the complex variables D_1, D_2, D_3 appear in B only as the heads of the headed boxes and in the positive part $\langle\gamma\rangle$ in special 'complex blocks' based on a given order

[1] This was first established by using the algorithm SCAN, thanks to a suggestion of Andreas Herzig, before complex formulae were introduced.

of the complex variables. This special structure of the complex formulae is needed to guarantee their translation into inductive formulae by means of the substitutions T, S, respectively T', S'.

The question of how to find such suitable substitutions arises. For instance, the definitions of T and T' are obvious, but how to find S and S'? We will show later that substitutions like S and S' can be effectively computed from the form of the given complex formula as a solution of a special system of Boolean equations corresponding to that formula. In order to give some preliminary intuition we present that system of equations for the case of the second example. Looking at the formula B' we see that the substitution S' should satisfy the following equations:

$$S'(D_3) \equiv q_1,\ S'(D_3 \wedge D_2) \equiv q_1 \wedge q_2,\ S'(D_3 \wedge D_2 \wedge D_1) \equiv q_1 \wedge q_2 \wedge q_3.$$

In this system S' is an unknown substitution, which has to be extracted from the given equations. Using the fact that S' should be a substitution, the system can be transformed equivalently to the following one which has to be solved with respect to $S'(p_1), S'(p_2)$:

$$\left\| \begin{array}{l} q_1 \equiv (\neg S'(p_1) \vee S'(p_2)) \\ q_1 \wedge q_2 \equiv (\neg S'(p_1) \vee S'(p_2)) \wedge (S'(p_1) \vee \neg S'(p_2)) \\ q_1 \wedge q_2 \wedge q_3 \equiv (\neg S'(p_1) \vee S'(p_2)) \wedge (S'(p_1) \vee \neg S'(p_2)) \wedge (S'(p_1) \vee S'(p_2)) \end{array} \right. \tag{1}$$

By easy Boolean manipulations (negate both part of the first equation and add disjunctively to the second equation, and do the same with the second and third equations) this system can be transformed into the following equivalent one:

$$\left\| \begin{array}{l} q_1 \equiv \neg S'(p_1) \vee S'(p_2) \\ \neg q_1 \vee q_2 \equiv S'(p_1) \vee \neg S'(p_2) \\ \neg q_1 \vee \neg q_2 \vee q_3 \equiv S'(p_1) \vee S'(p_2) \end{array} \right. , \tag{2}$$

Now, (2) can be easily solved with respect to $S'(p_1)$ and $S'(p_2)$: by taking conjunctions on the left and on the right in the second and third equations and simplifying, one can obtain $S'(p_1) \equiv \neg q_1 \vee (q_2 \wedge q_3)$; then, by taking the conjunctions on the left and on the right in the first and third equations, one can obtain $S'(p_2) \equiv q_1 \wedge (\neg q_2 \vee q_3)$ — just what was expected.

The system for the substitution T' is the following one:

$$T'(q_1) \equiv D_1,\ T'(q_1 \wedge q_2) \equiv D_1 \wedge D_2,\ T'(q_1 \wedge q_2 \wedge q_3) \equiv D_1 \wedge D_2 \wedge D_3.$$

An obvious solution of this system with respect to $T'(q_1), T'(q_2), T'(q_3)$ is: $T'(q_1) = D_1, T'(q_2) = D_2$ and $T'(q_3) = D_3$.

To conclude this informal discussion, let us mention that the polyadic version of SQEMA from [7] does not succeed on the formula B, but it succeeds on its equivalent inductive formula B' (because SQEMA succeeds on all inductive formulae). So, the intuitive idea for the extension of SQEMA is to supply it with a sub-procedure based on substitutions like S and S', whence the name of this extension: "SQEMA with substitutions", hereafter denoted SQEMA$^{\text{sub}}$.

2.3. Formal definitions of the classes of RCM- and ICM-formulae

Let p be a propositional variable. Denote $p^0 =_{def} \neg p$ and $p^1 =_{def} p$. Let $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ be a list of different variables. Formulae of the form $D = p_1^{i_1} \vee \ldots \vee p_n^{i_n}$ (in this order of the variables) will be called

elementary disjunctions. If $\mathbf{p} = \langle p_1 \rangle$ then we have only two (degenerate) elementary disjunctions, p_1 and $\neg p_1$. There are of course exactly 2^n non-equivalent elementary disjunctions of $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$. Sometimes we will consider a fixed order of all elementary disjunctions: $D_1, \ldots, D_{2^n}$. If we consider $\{0, 1\}$-strings $(i_1, \ldots, i_n)$ as binary codes of the integers then there are two natural orderings of the sequence of D_i's: the **increasing order** – starting from $(0, \ldots, 0) = 0$ and ending with $(1, \ldots, 1) = 2^n - 1$, and the **decreasing order** which is just the opposite of the increasing order.

Let $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ and let the elementary disjunctions built from $\mathbf{p}$ have the same order of the variables. Let $D(\mathbf{p}) = \langle D_1, \ldots, D_{2^n-1} \rangle$ be a fixed sequence of different elementary disjunctions (the last one D_{2^n} is missing), and let $D^*(\mathbf{p}) = \langle D_1, D_1 \wedge D_2, \ldots, D_1 \wedge \cdots \wedge D_{2^n-1} \rangle$. The pair $\langle D(\mathbf{p}), D^*(\mathbf{p}) \rangle$ will be called a **propositional complex (of dimension** n**)**. The disjunctions in $D(\mathbf{p})$ will be called **complex variables (of dimension** n), and the elements from $D^*(\mathbf{p})$ will be called **complex blocks (of dimension** n**)**. We say that the a complex $\langle D(\mathbf{p}), D^*(\mathbf{p}) \rangle$ is disjoint from a complex $\langle D(\mathbf{q}), D^*(\mathbf{q}) \rangle$ if the strings $\mathbf{p}$ and $\mathbf{q}$ do not share variables.

As we have seen in section 2.2, elementary disjunctions D_i can be used to "code" in some sense ordinary variables. This suggests that one could see the elementary disjunctions D_i as a new kind of "complex" variable.

Let Σ be a set of pairwise disjoint propositional complexes. Let $A(p_1, ..., p_k)$ be a formula and $B_1, \ldots, B_k$ be a list of complex blocks from Σ. Then $A(B_1, \ldots, B_k)$ is called a **complex atom in** Σ. If p_i occurs in $A(p_1, ..., p_k)$ only positively (negatively) then B_i occurs in $A(B_1, \ldots, B_k)$ positively (negatively). If $A(p_1, ..., p_k)$ is a positive (negative) formula then $A(B_1, ..., B_k)$ is a **positive (negative) complex atom in** Σ. A **complex essentially box-formula in** Σ is any formula of the type

$$B = [\beta](D, N_1, ..., N_m) = [\beta](D, \overrightarrow{N}),$$

where β is a modal term of arity $m + 1$, $\overrightarrow{N} = N_1, ..., N_m$ is a string of negative complex atoms in Σ and D is a complex variable from Σ.[2] A formula of this type is called a **headed complex box**, where D is the **head** of B and $\overrightarrow{N}$ is the **negative part** of B.

Note that in $\neg B = \neg[\beta](D, N_1, ..., N_m)$ the head D has a negative occurrence and all complex blocks in $N_1, ..., N_m$ have positive occurrences. Also note that β can be a composed modal term. If, for instance, α and β are unary terms, then the formula $[\alpha]p \vee [\beta]q$ can be represented as $[\gamma](p, q)$, where γ is the following composition $\gamma = \iota_2(\alpha, \beta)$.

Recall that a **constant formula** is a formula not containing propositional variables.

Definition 2.2. A **recursive complex modal formula** (RCM-formula for short) in Σ is any constant formula or a formula $A = [\alpha](\neg B_1, \ldots, \neg B_{m,} C_1, \ldots, C_n)$ where $B_1, \ldots, B_n$ are complex essentially box formulae in Σ, $C_1, \ldots, C_n$ are positive complex atoms in Σ, and where both m and n may be zero. The formulae B_i, $1 \leq i \leq m$, are called **the headed boxes of** A, while the formulae C_j, $1 \leq j \leq n$, are called **the positive components of** A. Note that all heads in A have only negative occurrences and all complex blocks (other than heads) in A have only positive occurrences.

In the case that all propositional complexes of A are of the form $P = \langle \langle p \rangle, \langle p \rangle \rangle$ (i.e., of dimension one and with the decreasing order of the elementary disjunctions), A is called a **recursive modal formula**, or RM-formula for short. Thus, RM-formulae have no (non-degenerate) complex variables.

[2]Here D need not be only in the first argument place, but we put it first for simplicity of notation.

Remark 2.1. We can generalize definition 2.2 slightly by allowing simplifications of the complex blocks. For example, the complex block $(p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2)$ can be simplified to $p_1 \wedge p_2$. However, when proving theorems for RCM-formulae, we will always assume that they have not been simplified.

Let $A = [\alpha](\neg B_1, ..., \neg B_{m,} C_1, ..., C_n)$ be a RCM-formula and P be a propositional complex of A. We say that P **is essential for** A if some of the heads of A belong to P.

Definition 2.3. The **dependency digraph** of an RCM-formula $A = [\alpha](\neg B_1, ..., \neg B_{m,} C_1, ..., C_n)$ is the digraph $G = (V_{A,} E_A)$, where the vertex set $V_A = \{P_1, ..., P_k\}$ is the set of all essential propositional complexes of A, and $P_i E_A P_j$ holds if a complex block from the propositional complex P_i occurs in a negative component of a formula B_l from $B_1, ..., B_m$ such that the head of B_l is from the propositional complex P_j. A digraph is **acyclic** if it contains no directed cycles or loops.

Definition 2.4. An RCM-formula A is called an **inductive complex modal formula**, or an ICM-formula for short, if the dependency digraph of A is acyclic. If the dependency digraph of A has no arcs at all, then A is called a **simple ICM-formula**. In the case that all propositional complexes of A are of the form $P = \langle\langle p\rangle, \langle p\rangle\rangle$ (i.e., of dimension one and with the decreasing order of the elementary disjunctions), A is called an **inductive modal formula**, or an IM-formula for short.

The formulae A, B in the examples from subsection 2.2 are simple ICM-formulae, while the formulae A', B' are inductive formulae. In order to see this for A, we rewrite it in the following box form:

$$\Box\neg\Box D_1 \vee \Box\neg\Box D_2 \vee \Box\neg D_3 \vee \Box\Diamond(D_1 \wedge D_2 \wedge D_3).$$

If we denote by α the unary modal term corresponding to $\Box$, the formula above can be presented as:

$$[\beta](\neg[\alpha]D_1, \neg[\alpha]D_2, \neg[\alpha]D_3, \langle\alpha\rangle(D_1 \wedge D_2 \wedge D_3)),$$

where $\beta = \iota_4(\alpha, \alpha, \alpha, \alpha)$. In this form the formula is obviously a simple ICM-formula with heads D_1, D_2, D_3 and only one complex block $D_1 \wedge D_2 \wedge D_3$.

Here is an example of ICM-formula which is not a simple one:

Example 2.1. $C = [\alpha](\neg[\beta](p_1 \vee p_2), \neg[\beta](p_1 \vee \neg p_2), \neg[\beta](\neg p_1 \vee p_2), \neg[\gamma](q, N((p_1 \vee p_2), ((p_1 \vee p_2) \wedge (p_1 \vee \neg p_2)), ((p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2)), \mathsf{Pos}(q))$ where α, β, and γ are modal terms of respective arities, $N(\cdot, \cdot, \cdot)$ is a negative formula built from three different variables, and $\mathsf{Pos}(q)$ is a positive formula built from the variable q. C has two propositional complexes, namely, P, containing the complex variables $(p_1 \vee p_2), (p_1 \vee \neg p_2), (\neg p_1 \vee p_2)$, and Q, containing only the degenerate complex variable q. The dependency digraph has only one arc from P to Q.

Remark 2.2. Inductive complex modal formulae are obvious generalizations of inductive modal formulae introduced in [17] and, under another name, in [26]. The adjective **inductive** comes from the fact that their first-order equivalents can be computed by a procedure using simple induction. Recursive complex formulae are obvious generalization of recursive modal formulae which were introduced in [17] under the name **regular formulae**. It was proven in [17] (see also [8]) that regular formulae have equivalents in the extension of first-order logic with least fix points which are solutions of systems of recursive equations, hence the name **recursive modal formulae**.

2.4. Examples of ICM-formulae for which SQEMA fails

SQEMA succeeds on some complex modal formulae. For instance, the formula $A = \Diamond\Box(p_1 \vee p_2) \wedge \Diamond\Box(p_1 \vee \neg p_2) \wedge \Diamond\Box(\neg p_1 \vee p_2) \rightarrow \Box\Diamond(p_1 \wedge p_2)$ discussed in Section 2.2 is one of the examples in [6] for which the monadic SQEMA succeeds. However, this is not always the case, as the following examples illustrate:

Example 2.2. Consider the ICM-formula:

$$\varphi = [\mathbf{3}](\neg[\mathbf{1}](p_1 \vee \neg p_2), \neg[\mathbf{1}](\neg p_1 \vee p_2), \langle\mathbf{2}\rangle((p_1 \vee \neg p_2), (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2))),$$

where $\mathbf{1}, \mathbf{2}, \mathbf{3}$ are modal terms of arities 1, 2, and 3, respectively. Let us see that SQEMA fails on φ.

Step 1 Negating and moving the negation inside, we obtain

$$\neg\varphi \equiv \langle\mathbf{3}\rangle([\mathbf{1}](p_1 \vee \neg p_2), [\mathbf{1}](\neg p_1 \vee p_2), [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2))).$$

Step 2 The initial system of SQEMA-equations:

$$\left\| \; \neg\mathbf{i} \vee \langle\mathbf{3}\rangle([\mathbf{1}](p_1 \vee \neg p_2), [\mathbf{1}](\neg p_1 \vee p_2), [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2))) \right. \quad .$$

Step 3 Applying the $\Diamond$ rule yields:

$$\left\| \begin{array}{l} \neg\mathbf{i} \vee \langle\mathbf{3}\rangle(\mathbf{j}_1, \mathbf{j}_2, \mathbf{j}_3) \\ \neg\mathbf{j}_1 \vee [\mathbf{1}](p_1 \vee \neg p_2) \\ \neg\mathbf{j}_2 \vee [\mathbf{1}](\neg p_1 \vee p_2) \\ \neg\mathbf{j}_3 \vee [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2)) \end{array} \right. \quad .$$

Step 4 We choose to try and eliminate p_1 first. We apply the $\Box$-rule to the second equation:

$$\left\| \begin{array}{l} \neg\mathbf{i} \vee \langle\mathbf{3}\rangle(\mathbf{j}_1, \mathbf{j}_2, \mathbf{j}_3) \\ \left[\mathbf{1}^{-1}\right] \neg\mathbf{j}_1 \vee (p_1 \vee \neg p_2) \\ \neg\mathbf{j}_2 \vee [\mathbf{1}](\neg p_1 \vee p_2) \\ \neg\mathbf{j}_3 \vee [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2)) \end{array} \right. \quad .$$

Step 5 After applying commutativity of disjunction, and then the Left-shift rule to the second equation we obtain:

$$\left\| \begin{array}{l} \neg\mathbf{i} \vee \langle\mathbf{3}\rangle(\mathbf{j}_1, \mathbf{j}_2, \mathbf{j}_3) \\ ([\mathbf{1}^{-1}]\neg\mathbf{j}_1 \vee \neg p_2) \vee p_1 \\ \neg\mathbf{j}_2 \vee [\mathbf{1}](\neg p_1 \vee p_2) \\ \neg\mathbf{j}_3 \vee [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2)) \end{array} \right. \quad .$$

Now we obtain p_1 separated in the second equation and negative in the third equation, but neither negative nor positive in the fourth equation, so the Ackermann-rule cannot be applied.

Step 6 Now, we try to eliminate p_2 and apply the $\Box$-rule to the third equation:

$$\left\|\begin{array}{l} \neg\mathbf{i} \vee \langle\mathbf{3}\rangle(\mathbf{j}_1, \mathbf{j}_2, \mathbf{j}_3) \\ ([\mathbf{1}^{-1}]\neg\mathbf{j}_1 \vee \neg p_2) \vee p_1 \\ \left[\mathbf{1}^{-1}\right]\neg\mathbf{j}_2 \vee (\neg p_1 \vee p_2) \\ \neg\mathbf{j}_3 \vee [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2)) \end{array}\right. .$$

Step 7 After applying the Left-shift rule to the third equation we obtain:

$$\left\|\begin{array}{l} \neg\mathbf{i} \vee \langle\mathbf{3}\rangle(\mathbf{j}_1, \mathbf{j}_2, \mathbf{j}_3) \\ ([\mathbf{1}^{-1}]\neg\mathbf{j}_1 \vee \neg p_2) \vee p_1 \\ ([\mathbf{1}^{-1}]\neg\mathbf{j}_2 \vee \neg p_1) \vee p_2 \\ \neg\mathbf{j}_3 \vee [\mathbf{2}]((\neg p_1 \wedge p_2), (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2)) \end{array}\right. .$$

Now, p_2 is separated in the third equation, but the Ackermann-rule cannot be applied because of the fourth equation. The only step which can be taken is to change polarity of p_1 or of p_2 and then to try the elimination procedure again (we invite the reader to do this). But, again, we will reach similar situation, because the fourth equation will be neither positive nor negative with respect to p_1 and p_2. So, the algorithm returns FAIL.

There are many examples of ICM-formulae for which SQEMA fails for similar reasons. We mention two more, without proof.

Example 2.3. Firstly, the formula B discussed in section 2.2:

$$B = [\alpha](\neg[\beta](\neg p_1 \vee p_2), \neg[\beta](p_1 \vee \neg p_2), \neg[\beta](p_1 \vee p_2), \langle\gamma\rangle((\neg p_1 \vee p_2), (p_1 \Leftrightarrow p_2), (p_1 \wedge p_2))).$$

Secondly, the following (simplified) ICM-formula in the standard monomodal language (see also [30]):

$$\Diamond\Box(p_1 \vee \neg p_2) \wedge \Diamond\Box(\neg p_1 \vee p_2) \wedge \Diamond\Box(p_1 \vee p_2) \to \Box(p_1 \vee \neg p_2) \vee \Box\Diamond(p_1 \leftrightarrow p_2) \vee \Box\Diamond\Box(p_1 \wedge p_2).$$

3. Complex substitutions

In this section we will prove some results that guarantee the existence of reversible substitutions for complex formulae.

Lemma 3.1. Let $\mathbf{p} = \langle p_1, \ldots, p_n\rangle$ be a list of different propositional variables and let $D_1, \ldots, D_{2^n}$ be a list of all (different) elementary disjunctions from $\mathbf{p}$. Then:

(i) if $i \neq j$ then $D_i \vee D_j \equiv \top$,

(ii) $\bigwedge_{i=1}^{2^n} D_i \equiv \bot$,

(iii) $\neg D_j \equiv \bigwedge_{i=1, i\neq j}^{2^n} D_i$, $j = 1, \ldots, 2^n$,

(iv) $\neg D_1 \vee \cdots \vee \neg D_i \vee D_{i+1} \equiv D_{i+1}$, $i = 1, \ldots, 2^n - 1$.

(v) $\bigwedge\{D_l : p_k \in D_l\} \equiv p_k$, $k = 1, \ldots, n$.

Proof:
Claim (i) is obvious, (ii) is a well-known, (iii) and (iv) follow from (i) and (ii) by easy Boolean manipulations. For (v) note that $\bigwedge\{D_l : p_k \in D_l\} \equiv p_k \vee \bigwedge_{i=1}^{2^{n-1}} D_i' \equiv p_k \vee \bot \equiv p_k$, where D_i' are all elementary disjunctions built from the variables of $\mathbf{p}$ different from p_k, which by (ii) is equivalent to $\bot$. $\Box$

Lemma 3.2. (First substitution lemma , [26])
Let $\mathbf{p} = \langle p_1, \ldots, p_n\rangle$ be a list of different propositional variables and $D_1, \ldots, D_{2^n}$ be a fixed list of all elementary disjunctions of them. Let $A_1 \ldots A_{2^n}$ be an arbitrary list of propositional formulae not containing variables from $\mathbf{p}$. Then, the following two conditions are equivalent:

1. There exists a substitution S acting on the variables $p_1, \ldots, p_n$ such that the following equations hold:

$$(\#1) \quad \left\| \begin{array}{lll} A_1 & \equiv & S(D_1) \\ A_2 & \equiv & S(D_2) \\ \ldots & & \ldots \\ A_{2^n} & \equiv & S(D_{2^n}). \end{array} \right.$$

2. The following two conditions hold for any $i, j \leq 2^n$:

   (a) If $i \neq j$ then $A_i \vee A_j \equiv \top$,

   (b) $\bigwedge_{i=1}^{2^n} A_i = \bot$.

Moreover, if the condition 2 is fulfilled, then the substitution S is uniquely determined by the equations

$$(\#2) \quad S(p_k) = \bigwedge\{A_l \mid p_k \in D_l, l \leq 2^n\}, k = 1, \ldots, n.$$

An example of how to solve a system like $(\#1)$ is given in Section 2.2.

Lemma 3.3. (Second substitution lemma , [26])
Let $\mathbf{p} = \langle p_1, \ldots, p_n\rangle$, be a sequence of different propositional variables, let $D_1, \ldots, D_{2^n}$ be a sequence of all elementary disjunctions of these variables, and let $\mathbf{q} = \langle q_1, \ldots, q_{2^n-1}\rangle$ be a sequence of propositional formulae not containing variables from $\mathbf{p}$. Then:

1. There exists a substitution S (depending on the given order of the disjunctions D_i) acting only on the variables $p_1, \ldots, p_n$ and satisfying the following conditions:

$$(*) \quad \left\| \begin{array}{lll} q_1 & \equiv & S(D_1) \\ q_1 \wedge q_2 & \equiv & S(D_1 \wedge D_2) \\ \ldots & & \ldots \\ q_1 \wedge q_2 \wedge \ldots \wedge q_{2^n-1} & \equiv & S(D_1 \wedge D_2 \wedge \ldots \wedge D_{2^n-1}). \end{array} \right.$$

2. The conditions $(*)$ uniquely determine S (up to Boolean equivalence) and S can be effectively computed from them.

3. The substitution S also satisfies the following conditions:

$$(**)\quad \left\| \begin{array}{lcl} q_1 & \models & S(D_1) \\ q_2 & \models & S(D_2) \\ \dots & & \dots \\ q_{2^n-1} & \models & S(D_{2^n-1}). \end{array} \right.$$

In [26] lemma 3.3 was proven by an application of Lemma 3.2 and the formulae defining S are given in that proof. The proof contains the following fact, which we formulate now explicitly as Lemma 3.4, from which Lemma 3.3 can be easily derived.

Lemma 3.4. Let the assumptions of Lemma 3.3 be fulfilled, and let S be a substitution acting on the variables from the list $\mathbf{p}$. Also let $A_1 = q_1$, $A_2 = \neg q_1 \vee q_2$, ..., $A_{i+1} = \neg q_1 \vee \ldots \vee \neg q_i \vee q_{i+1}$, ..., $A_{2^n} = \neg q_1 \vee \ldots \vee \neg q_{2^n-1}$.

1. The following two conditions (a) and (b) are equivalent:

$$\text{(a)}\quad \left\| \begin{array}{lcl} q_1 & \equiv & S(D_1) \\ q_1 \wedge q_2 & \equiv & S(D_1 \wedge D_2) \\ \dots & & \dots \\ q_1 \wedge q_2 \wedge \ldots \wedge q_{2^n-1} & \equiv & S(D_1 \wedge D_2 \wedge \ldots \wedge D_{2^n-1}). \end{array} \right.$$

$$\text{(b)}\quad \left\| \begin{array}{lcl} A_1 & \equiv & S(D_1) \\ A_2 & \equiv & S(D_2) \\ \dots & & \dots \\ A_{2^n} & \equiv & S(D_{2^n}). \end{array} \right.$$

2. If S satisfies condition (a) then it is determined by the following formulae

$$S(p_k) = \bigwedge\{A_l : p_k \in D_l, l \leq 2^n\}, k = 1, \ldots, n. \qquad (S(p_k))$$

Proof:
(Sketch)

1. An idea for the proof of $(a) \Rightarrow (b)$ is given by some examples in Section 2.2, using Lemma 3.1. The implication $(b) \Rightarrow (a)$ can be proved by direct Boolean calculations on the left hand side and then using Lemma 3.1 for the right hand side.

2. It is easy to see that the formulae A_i satisfy the conditions of Lemma 3.2, so we may apply it. From that lemma and (1) we obtain (2).

□

Lemma 3.5. Let S be the substitution from Lemma 3.4 with the assumption that the propositional variables in $\mathbf{q} = \langle q_1, \dots, q_{2^n-1} \rangle$ are different. Then S is reversible with domain $\mathbf{p} = \langle p_1, \dots, p_n \rangle$ and co-domain $\mathbf{q}$. In particular, the substitution T, defined by putting $T(q_i) = D_i$, $i = 1, \dots, 2^n - 1$, reverses S.

Proof:
The explicit definition of S from Substitution lemma 3.4 is

$$S(p_k) = \bigwedge \{\neg q_1 \vee \ldots \vee \neg q_{l-1} \vee q_l \mid p_k \in D_l, l \leq 2^n\}, \quad k = 1, \dots, n.$$

Then, by Lemma 3.1 we obtain

$$\begin{aligned} T(S(p_k)) &= T(\bigwedge \{\neg q_1 \vee \ldots \vee \neg q_{l-1} \vee q_l \mid p_k \in D_l, l \leq 2^n\}) \\ &= \bigwedge \{\neg D_1 \vee \ldots \vee \neg D_{l-1} \vee D_l \mid p_k \in D_l, l \leq 2^n\} \\ &= \bigwedge \{D_l \mid p_k \in D_l, l \leq 2^n\} = p_k. \end{aligned}$$

□

Remark 3.1. Lemmas 3.3 and 3.5 together guarantee the existence of reversible substitutions. The substitutions of the form S from Lemma 3.3 will be used in $\mathsf{SQEMA}^{\mathsf{sub}}$. Let us note that S depends on the given list of the variables $\mathbf{p} = \langle p_1, \dots, p_n \rangle$ and the sequence $\langle D_1, \dots, D_{2^n} \rangle$ of the elementary disjunctions built from $\mathbf{p}$. Another feature of the specific substitution S is that its codomain contains $2^n - 1$ variables where n is the number of variables of the domain of S. So, S produces formulae with exponentially many more variables than the input formula. Let us also mention that we cannot claim that the substitution T from Lemma 3.5 (which reverses S) is reversible.

4. The translation θ

The following theorem is the main technical statement in this section.

Theorem 4.1. (Translation Theorem)
For every RCM-formula A there exists an effectively computable translation θ such that the following hold:

(i) $\theta(A)$ is an RM-formula, and if A is an ICM-formula, then $\theta(A)$ is an inductive modal formula.

(ii) A is locally equivalent to $\theta(A)$.

Proof:
Definition of the translation θ: The translation θ has a very simple definition – it replaces all complex variables in A with new propositional variables ("new" here means "not appearing in A"), replacing different complex variables by different new variables.

Consequently, θ transforms all complex blocks into positive formulae, hence it transforms A into a recursive modal formula $\theta(A)$. Furthermore, if A is an ICM-formula, then it is easy to see that the dependency digraph of $\theta(A)$ does not contain cycles, and therefore $\theta(A)$ is an inductive formula. Thus, (i) is proved.

(ii) Since the propositional complexes in A are disjoint, we may realize the translation θ step-by-step, substituting the complex variables one by one (in any arbitrary order). More precisely, we will consider a 'partial' translation θ_P for each propositional complex P, by just substituting in A only the occurrences of complex variables from P. So, it is sufficient to prove that A is locally equivalent to $\theta_P(A)$.

Let $\mathfrak{F} = (W, \{R_\alpha\}_{\alpha \in \mathrm{MT}_\tau}, \mathbb{W})$ be a **general τ-frame** (for the corresponding definition see [2] or [7]), $x \in W$ and A a be an RCM-formula of type τ. We have to prove that A is locally valid at x iff $\theta(A)$ is. First, note that A can be obtained from $\theta_P(A)$ by ordinary substitution, so the direction from $\theta_P(A)$ to A is immediate.

For the direction from A to $\theta_P(A)$ we proceed as follows. Let $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ and let $D(\mathbf{p}) = \langle D_1, \ldots, D_{2^n-1} \rangle$ be the string of complex variables built from $\mathbf{p}$. For the translation we need a string of different new variables $\langle q_1, \ldots, q_{2^n-1} \rangle$. Then, by the Substitution lemma 3.3 there exists a substitution S satisfying $(*)$ and $(**)$. (To be more precise, we should denote S by S_P, but we keep the notation S for simplicity).

Internal Lemma. *If Q is a complex positive or negative atom then $S(Q) \equiv \theta_P(Q)$.*

The proof is by a simple induction on the formation of Q and by application of the conditions $(*)$ in the Substitution lemma 3.3.

Since A is an RCM-formula it has the following form $A = [\alpha](\neg B_1, ..., \neg B_k, C_1, ..., C_l)$. Then, applying S and θ_P to A we obtain:

$$S(A) = [\alpha](\neg S(B_1), ..., \neg S(B_k), S(C_1), ..., S(C_l))$$

and

$$\theta_P(A) = [\alpha](\neg \theta_P(B_1), ..., \neg \theta_P(B_k), \theta_P(C_1), ..., \theta_P(C_l)).$$

By the Internal Lemma , for all C_i, $i \leq l$, we have that $S(C_i) \equiv \theta_P(C_i)$, and therefore

$$\theta_P(A) \equiv [\alpha](\neg \theta_P(B_1), ..., \neg \theta_P(B_k), S(C_1), ..., S(C_l)). \tag{1}$$

Using (1) we have to show that $(\mathfrak{F}, x) \Vdash [\alpha](\neg B_1, ..., \neg B_k, C_1, ..., C_l)$ implies $(\mathfrak{F}, x) \Vdash [\alpha](\neg \theta_P(B_1), ..., \neg \theta_P(B_k), S(C_1), ..., S(C_l))$. Suppose for the sake of contradiction that this is not true, i.e.,

$$(\mathfrak{F}, x) \Vdash [\alpha](\neg B_1, ..., \neg B_k, C_1, ..., C_l) \tag{2}$$

but that

$$(\mathfrak{F}, x) \nVdash [\alpha](\neg \theta_P(B_1), ..., \neg \theta_P(B_k), S(C_1), ..., S(C_l)).$$

Then, there is a pointed model $(\mathcal{M}, x)$ over $\mathfrak{F}$ such that

$$(\mathcal{M}, x) \nvDash [\alpha](\neg \theta_P(B_1), ..., \neg \theta_P(B_k), S(C_1), ..., S(C_l)). \tag{3}$$

Then, there exist $y_1, \ldots, y_k, z_1, \ldots, z_l \in W$ such that

$$R_\alpha x y_1 \ldots y_k z_1 \ldots z_l, \tag{4}$$

$$(\mathcal{M}, y_i) \Vdash \theta_P(B_i), \quad i = 1, \ldots, k, \tag{5}$$

and

$$(\mathcal{M}, z_j) \nVdash S(C_j), \quad j = 1, \ldots, l. \tag{6}$$

Let $B = [\beta](D, N_1, ..., N_m) = [\beta](D, \overrightarrow{N})$ be any of the headed boxes B_i, $1 \leq i \leq k$, with head D and negative part $\overrightarrow{N}$.

Case 1: D is not a complex variable from $D(\mathbf{p})$. Then $\theta_P(D) = S(D) = D$. Also, by the Internal Lemma we have $S(\overrightarrow{N}) \equiv_l \theta_P(\overrightarrow{N})$. Consequently $S(B) \equiv \theta_P(B)$. So in this case

$$(\mathcal{M}, y_i) \Vdash S(B_i). \tag{7}$$

Case 2: The head D of B is a complex variable from $D(\mathbf{p})$ and $D = D_m$ for some $m \leq 2^n - 1$. Then $\theta_P(D_m) = q_m$, and consequently $\theta_P(B_i) = [\beta](q_m, \theta_P(\overrightarrow{N})) = [\beta](q_m, S(\overrightarrow{N}))$. So, by (5), we have

$$(\mathcal{M}, y_i) \Vdash [\beta](q_m, S(\overrightarrow{N})). \tag{8}$$

We will show in this case that $(\mathcal{M}, y_i) \Vdash [\beta](S(D_m), S(\overrightarrow{N}))$. Suppose that this is not true. Then there exist $t_1, \ldots, t_{\rho(\beta)}$ such that

$$R_\beta y_i t_1 \ldots t_{\rho(\beta)}, \tag{9}$$

and

$$(\mathcal{M}, t_1) \nVdash S(D_m), \tag{10}$$

and for all $1 < j \leq \rho(\beta)$,

$$(\mathcal{M}, t_j) \nVdash S(N_j). \tag{11}$$

By the Substitution lemma 3.3, condition $(**)$, we have $q_m \models S(D_m)$, and by (10) we obtain

$$(\mathcal{M}, t_1) \nVdash q_m. \tag{12}$$

Then by (9), (12) and (11) we obtain $(\mathcal{M}, y_i) \nVdash [\beta](q_m, S(\overrightarrow{N}))$, which contradicts (8). Therefore, we have that $(\mathcal{M}, y_i) \Vdash [\beta](S(D_m), S(\overrightarrow{N}))$.

Thus, in both cases we have that

$$(\mathcal{M}, y_i) \Vdash S(B_i), \quad i = 1, \ldots k. \tag{13}$$

Now by (4), (6) and (13) we obtain $(\mathcal{M}, x) \nVdash S([\alpha](\neg B_1, \ldots, \neg B_k, C_1, \ldots, C_l))$, which contradicts (2), since local validity is preserved by substitutions. $\square$

As a corollary we obtain the following important generalization of the Sahlqvist theorem for ICM-formulae:

Theorem 4.2. (Sahlqvist theorem for ICM-formulae)
Every ICM-formula is locally first-order definable and locally d-persistent. Moreover, its local first-order correspondent can be effectively computed.

Proof:
By theorem 4.1 every ICM-formula A is locally equivalent to the inductive formula $\theta(A)$. By Corollary 60 of [17], every inductive formula B is locally first-order definable and locally d-persistent and the local correspondent of B can be effectively computed. The claim now follows since local equivalence preserves local first-order definability and local d-persistence. The second claim of the theorem follows since θ is an effective translation. $\square$

5. Complex normal forms and the translation Σ

5.1. The substitution σ

The translation $\theta(A)$, which we defined in the previous section, is the simplest one that translates an RCM-formula A directly into an RM-formula $\theta(A)$ with the property that if A is an ICM-formula then $\theta(A)$ is an inductive modal formula. $\theta(A)$ is realized by a sequence of translations θ_P corresponding to all propositional complexes P of A, i.e., $\theta(A) = \theta_{P_1}(\theta_{P_2}(\ldots(\theta_{P_n}(A))\ldots))$, where $P_1, \ldots, P_n$ are the propositional complexes of the formula A. The proof of the translation theorem (Theorem 4.1) shows that A and $\theta_P(A)$ are locally equivalent, which implies that A and $\theta(A)$ are locally equivalent, too. In fact, the proof uses the substitutions S_P, the existence of which is established by the second substitution lemma (Lemma 3.3), such that $S_P(A)$ and $\theta_P(A)$ are locally equivalent.

This motivates the introduction of a **substitution** σ, obtained by applying consecutively all substitutions of the type S_P to A for all different propositional complexes of A, namely:

$$\sigma(A) = S_{P_1}(S_{P_2}(\ldots(S_{P_n}(A))\ldots)),$$

So the translation theorem implies that A, $\sigma(A)$, and $\theta(A)$ are locally equivalent. The difference between θ and σ is that θ is *not a substitution*, while σ is a reversible substitution which guarantees the local equivalence between A and $\sigma(A)$ for arbitrary formula A. Another difference between θ and σ is that the image of an ICM-formula under θ is always an inductive modal formula, while its image under σ is generally not in the format of inductive formulae. However, the latter can always be post-processed into an inductive formula. Let us illustrate this post-processing on the formula from example 2.1:

$C = [\alpha](\neg[\beta](p_1 \vee p_2), \neg[\beta](p_1 \vee \neg p_2), \neg[\beta](\neg p_1 \vee p_2), \neg[\gamma](q, N((p_1 \vee p_2), ((p_1 \vee p_2) \wedge (p_1 \vee \neg p_2)), ((p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2)), \mathsf{Pos}(q))$.

Let P be the propositional complex corresponding to the three complex variables $D_1 = (p_1 \vee p_2)$, $D_2 = (p_1 \vee \neg p_2)$ and $D_3 = (\neg p_1 \vee p_2)$, in this order. To define S_P we need three new variables q_1, q_2, q_3. Using the corresponding formulae from Lemma 3.4 we obtain: $S_P(p_1) = q_1 \wedge (\neg q_1 \vee q_2)$ and $S_P(p_2) = q_1 \wedge (\neg q_1 \vee \neg q_2 \vee \neg q_3)$.

By applying this substitution to C we do not automatically obtain an inductive formula, but using the properties of S_P from Substitution lemmas 3.3 and 3.4 we obtain the following equivalences:

$S_P(D_1) \equiv q_1$,
$S_P(D_2) \equiv \neg q_1 \vee q_2$,
$S_P(D_3) \equiv \neg q_1 \vee \neg q_2 \vee q_3$,
$S_P(D_1 \wedge D_2) \equiv q_1 \wedge q_2$,
$S_P(D_1 \wedge D_2 \wedge D_3) \equiv q_1 \wedge q_2 \wedge q_3$.

Using these simplifications we obtain that $S_P(C)(= \sigma(C))$ is semantically equivalent to the formula:

$$C' = [\alpha](\neg[\beta]q_1, \neg[\beta](\neg q_1 \vee q_2), \neg[\beta](\neg q_1 \vee \neg q_2 \vee q_3), \neg[\gamma](q, N(q_1, (q_1 \wedge q_2), (q_1 \wedge q_2) \wedge q_3)), \mathsf{Pos}(q)).$$

This is an inductive formula with 6 arcs in the dependency digraph: $q_1 \longrightarrow q_2$, $q_1 \longrightarrow q_3$, $q_1 \longrightarrow q$, $q_2 \longrightarrow q_3$, $q_2 \longrightarrow q$, and $q_3 \longrightarrow q$. The result of the application of θ to C, however, is different:

$$C'' = [\alpha](\neg[\beta]q_1, \neg[\beta]q_2, \neg[\beta]q_3, \neg[\gamma](q, N(q_1, (q_1 \wedge q_2), (q_1 \wedge q_2) \wedge q_3)), \mathsf{Pos}(q)).$$

C'' is an inductive formula, simpler than C'. It differs from C' only in the heads. By the translation theorem 4.1, C' and C'' are locally equivalent and hence, although they are different inductive formulae, they define the same first-order conditions.

The above discussion leads to the conclusion that in the extension of SQEMA to SQEMA$^{\text{sub}}$ it is better to use substitutions like S_P, because they are reversible and therefore will guarantee the correctness of the extension. But, we would have to modify S_P (and consequently, σ) in order to be able to obtain inductive formulae **without the need for additional post-processing**, in the way that, for instance, θ does. One way to do this is to make it possible to apply σ not to variables, but to some Boolean subformulae, for instance, to the complex variables and complex blocks for which the Substitution lemma 3.3 and Lemma 3.4 guarantee a better result. For instance, in the example above, the better result for $S_P(D_1)$ is q_1 and for $S_P(D_1 \wedge D_2)$ it is $q_1 \wedge q_2$.

To this end, in order for such extended σ (denoted later on by Σ) to be applicable to arbitrary formulae, we have to transform the Boolean sub-formulae of the input of S_P into some 'complex normal form', which is the topic of the next subsection.

5.2. Complex normal forms

By a **Boolean formula** we mean any formula of the classical propositional language.

Let B be a subformula of a modal formula A. This means that B may have several occurrences in A. Note that any two such occurrences are either identical or disjoint. The positions in A where the occurrences of B are placed will be called **the locations of** B **in** A. If C is a subformula of A and B is a subformula of C, we say that C is a **proper extension of** B if B is a proper subformula of C. An occurrence of a Boolean subformula B at a given location in a modal formula A is called a **maximal Boolean subformula of** A **at this location** if B has no proper Boolean extensions that occurs at that location. For example, in the formula $A = \Box(\neg(p \wedge q) \vee \Diamond(p \wedge q)) \vee (r \wedge s)$ the Boolean subformula $(p \wedge q)$ has two occurrences. It is not maximal at the first location, because it has as a proper Boolean extension $\neg(p \wedge q)$ which occurs there, but it is maximal at the second location.

Let $B_1, \ldots, B_m$, be the list of all maximal Boolean subformulae at their respective locations, listed in the order of their occurrence in A. The formulae $B_1, \ldots, B_m$ will be called the **Boolean blocks** of A. Then we can regard A as being built from its Boolean blocks, and write $A = A(B_1, \ldots, B_m)$. Note that all B_i have different locations and are disjoint, so replacement of each block with some formula will not destroy the other blocks. The list of Boolean blocks in the above example A is $B_1 = \neg(p \wedge q)$, $B_2 = (p \wedge q)$, $B_3 = (r \wedge s)$ and, accordingly, $A = \Box(B_1 \vee \Diamond B_2) \vee B_3$.

Note that each occurrence of a variable in a formula $A = A(B_1, \ldots, B_m)$ is in one of its Boolean blocks $B_1, \ldots B_m$. Further, note that we can partition the set of Boolean blocks of a given formula A into disjoint non-empty clusters, called **neighbourhood classes**, satisfying the following conditions:

(1) formulae from different groups have no common variables, and
(2) each group is minimal with this property.

This partitioning corresponds to the equivalence relation obtained as the transitive closure of the relation of *sharing a common variable* between Boolean blocks. It also divides the set of variables in A into disjoint sets.

For instance, the neighbourhood classes for the above example A are $\mathrm{I} = \{B_1, B_2\}$ and $\mathrm{II} = \{B_3\}$, where the variables for the group I are $\{p, q\}$ while those for the group II are $\{r, s\}$.

Let $A = A(B_1, \ldots B_m)$ be a formula with a list of Boolean blocks $B_1, \ldots B_m$, and let them be divided into neighbourhood classes $C_1, \ldots, C_k$. Let C be any neighbourhood class and let $\mathbf{p}_C = \langle p_1, \ldots, p_n \rangle$ be a fixed sequence of all different variables occurring in the formulae from C and let $D(\mathbf{p}_C)$ be the set of all different elementary disjunctions built from the sequence $\mathbf{p}_C$ in which the variables are ordered as in $\mathbf{p}_C$. Now, in each set $D(\mathbf{p}_C)$ we fix an order of the elementary disjunctions $\langle D_1, \ldots, D_{2^n} \rangle$ and denote the resulting vector by $\overrightarrow{D(\mathbf{p}_C)}$. Note that $\overrightarrow{D(\mathbf{p}_C)}$ also determines the propositional complex $P(\overrightarrow{D(\mathbf{p}_C)}) = \langle \overrightarrow{D(\mathbf{p}_C)}, \overrightarrow{D^*(\mathbf{p}_C)} \rangle$ with $\overrightarrow{D^*(\mathbf{p}_C)} = \langle D_1, D_1 \wedge D_2, \ldots, D_1 \wedge D_2 \wedge \ldots \wedge D_{2^n - 1} \rangle$.

Further, we can replace each formula B_i from C by its conjunctive normal form B_i' using the disjunctions from $D(\mathbf{p}_C)$, so that the disjuncts in this normal form are ordered as in $\overrightarrow{D(\mathbf{p}_C)}$. In this way, by replacing every Boolean block B_i of A with its respective conjunctive normal form B_i' we obtain a formula A' called a **complex normal form of** A corresponding to the sequence $\overrightarrow{D(\mathbf{p}_{C_1})}, \ldots, \overrightarrow{D(\mathbf{p}_{C_k})}$. Propositional complexes $P_i = P(\overrightarrow{D(\mathbf{p}_{C_i})})$, $i = 1 \ldots, k$ are called **propositional complexes of** A'. So, A has many complex normal forms, all of which are equivalent to A, and the difference between them is only in the order of the elementary disjunctions in the conjunctive normal forms of its Boolean blocks. This order is inessential for the formula A but is essential for the substitutions of the form S_{P_i} corresponding to each propositional complex P_i occurring in the substitution σ. The following lemma is immediate:

Lemma 5.1. If A is a non-simplified RCM-formula, then A is itself in a complex normal form.

5.3. The translation Σ

Let A be a modal formula in a complex normal form and let $P_1, \ldots, P_k$ be the propositional complexes of A. Note that each P from the above sequence is determined by the corresponding vector $\overrightarrow{D(\mathbf{p}_C)} = \langle D_1, \ldots, D_{2^n} \rangle$, where n is the number of the propositional variables from $\mathbf{p}_C$. With each P we associate a sequence of new propositional variables $\langle q_1, \ldots, q_{2^n} \rangle$ such that the variables associated to $P_1, \ldots, P_k$ are all different. Then we consider the unique substitutions S_{P_i}, $i = 1, \ldots, k$, guaranteed to exist by the Substitution lemma 3.3. By the definition of the substitution σ introduced in Section 5.1 we have:

$$\sigma(A) = S_{P_1}(S_{P_2}(\ldots(S_{P_k}(A))\ldots)).$$

Note that σ coincides with the composition $S_{P1} \circ \ldots \circ S_{P_k}$. Note also that the order of the components in the above composition is inessential, because different propositional complexes of A have disjoint sets of propositional variables and each of the components S_{P_i} acts only on the Boolean blocks built from the variables in P_i. Because the S_{P_i} are reversible substitutions we immediately obtain the following lemma.

Lemma 5.2. Let A be a modal formula and A' a complex normal form of A. Then the formulae A, A' and $\sigma(A')$ are locally equivalent.

Now, we will transform σ component-wise into a new translation Σ, which, when applied to ICM-formulae, will, without any post-processing, produce inductive modal formulae.

The idea is the following. Let S_P be a fixed component of σ and let $\mathbf{q} = \langle q_1, \ldots, q_{2^n} \rangle$ be the sequence of the new variables associated to P. As a substitution S_P acts on an arbitrary formula A homomorphically. So, let $A = A(B_1, \ldots, B_m)$, where $B_1, \ldots, B_m$ are its Boolean blocks in the respective conjunctive normal form. Then we have:

$$S_P(A) = A(S_P(B_1), \ldots, S_P(B_m))$$

and

$$\sigma(A) = S_{P_1}(S_{P_2}(\ldots(S_{P_k}(A))\ldots)).$$

We want to define Σ to act on A per propositional complex, like σ does, i.e., we first want to define the translations Σ_P depending on each propositional complex P of A and then define $\Sigma(A)$, again in analogy to the definition of σ, as

$$\Sigma(A) =_{def} \Sigma_{P_1}(\Sigma_{P_2}(\ldots\Sigma_{P_k}(A))\ldots),$$

where $P_1, \ldots, P_k$ are all propositional complexes of A.

Definition 5.1. (The translation Σ)

Definition of $\Sigma_P(A)$. Let P be any of the propositional complexes of A. First we define Σ_P on the Boolean blocks of A.

Case 1: If B is a Boolean block of A not containing variables from P, then $\Sigma_P(B) = B$.

Case 2: B is a Boolean block of A built over variables from P.

Subcase 2.1: B is in the form D_i, for some $1 \leq i \leq 2^n$. Then, in accordance with Lemmas 3.3 and 3.4, we define $\Sigma_P(D_1) = q_1$, and for $1 < i < 2^n$, put $\Sigma_P(D_i) = \neg q_1 \vee \ldots \vee \neg q_{i-1} \vee q_i$. Finally in this case we put $\Sigma_P(D_{2^n}) = \neg q_1 \vee \ldots \vee \neg q_{2^n-1}$.

Subcase 2.2: B is in the form $D_1 \wedge D_2 \wedge \ldots \wedge D_i$, for some $1 \leq i \leq 2^n - 1$. Then, again in accordance with Lemmas 3.3 and 3.4, we define $\Sigma_P(B) = q_1 \wedge q_2 \wedge \ldots q_i$. If $i = 2^n$ then we put $\Sigma_P(B) = \perp$.

Subcase 2.3: B falls in neither of the previous two subcases. Then $B = D_{i_1} \wedge \ldots \wedge D_{i_j}$, $j > 1$, and in this case we put $\Sigma_P(B) = \Sigma_P(D_{i_1}) \wedge \ldots \wedge \Sigma_P(D_{i_j})$, considering conjuncts as in case 2.1.

Lastly we define $\Sigma_P(A) =_{def} A(\Sigma_P(B_1), \ldots, \Sigma_P(B_m))$.

Definition of $\Sigma(A)$ $\Sigma(A)$ is given by means of the translations Σ_{P_i}, $i = 1, \ldots, k$, as

$$\Sigma(A) = \Sigma_{P_1}(\Sigma_{P_2}(\ldots \Sigma_{P_k}(A)\ldots).$$

Note that Σ can act only on modal formulae in complex normal form. Also, the result $\Sigma(A)$ depends not only on A itself but it also depends on the propositional complexes of A (which are determined during the construction of A), and on the new propositional variables associated to the propositional complexes of A. So, when we write $\Sigma(A)$ we will also have in mind all this additional information needed to compute $\Sigma(A)$. The following lemma lists some useful properties of Σ.

Lemma 5.3. Let A be a modal formula and A' be a complex normal form of A. Then:

(i) $\sigma(A') \equiv \Sigma(A')$.

(ii) Let $A' = A(A_1, \ldots, A_i)$, where all subformulae $A_1, \ldots, A_i$ are composed from Boolean blocks of A'. Then $\Sigma(A') = A(\Sigma(A_1), \ldots, \Sigma(A_i))$.

(iii) A, A', and $\Sigma(A')$ are locally equivalent.

Proof:
The proofs of conditions (i) and (ii) are straightforward, because Σ is defined so as to capture the properties of σ on the complex variables and complex atoms. As for (iii), by Lemma 5.2, A, A', and $\sigma(A')$ are locally equivalent. By (i) $\sigma(A')$ and $\Sigma(A')$ are (semantically) equivalent, which implies the local equivalence of A, A' and $\Sigma(A')$. □

Now, we are ready to establish the following important proposition.

Proposition 5.1. Let A be an ICM-formula. Then:

(i) A is in a complex normal form and $\Sigma(A)$ is an inductive formula.

(ii) A and $\Sigma(A)$ are locally equivalent.

Proof:

(i) Let A be an ICM-formula. Then $A = [\alpha](\neg B_1, \ldots, \neg B_k, C_1, \ldots, C_l)$, where the B_i are the headed boxes of A and the C_j are the positive components of A. By Lemma 5.1 A is in a complex normal form. Then we may apply Σ to A, and by Lemma 5.3(ii) we obtain:

$$\Sigma(A) = [\alpha](\neg\Sigma(B_1), \ldots, \neg\Sigma(B_k), \Sigma(C_1), \ldots, \Sigma(C_l)).$$

Since all $C_1, \ldots C_l$ are built from complex atoms of the form $D_1 \wedge D_2 \wedge \ldots \wedge D_j$, taken from the propositional complexes of A, then Σ acts on them exactly as θ. The same holds for the negative parts of the headed boxes $B_1, \ldots, B_k$. The only difference between the actions of Σ and θ on headed boxes is how they act on their heads. If D_{i+1} is the head of some headed box $B = [\beta](D_{i+1}, N_1, \ldots, N_j)$ and D_{i+1} is a complex variable from some propositional complex P, then $\theta(D_{i+1}) = q_{i+1}$, while $\Sigma(D_{i+1}) = \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_i \vee q_{i+1}$. Observe that now $\Sigma(B) = [\beta](\neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_i \vee$

$q_{i+1}, \Sigma(N_1), \ldots, \Sigma(N_j))$, which is also a headed box with a head q_{i+1}. Indeed, this can be seen after composing β with the disjunction (which is a box modality) $\neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_i \vee q_{i+1}$. Then $\neg q_1$, $\neg q_2$,...,$\neg q_i$ go into the negative part of the headed box and q_{i+1} becomes a head. So, we see that $\Sigma(A)$ is an RM-formula in which all complex variables are of dimension 1.

In order to prove that $\Sigma(A)$ is an inductive formula, it remains to show that its dependency digraph has no cycles. Let A be an ICM-formula, G be the dependency digraph of A and denote by $\Sigma(G)$ the dependency digraph of $\Sigma(A)$. By definition, the vertices of G are the essential propositional complexes of A, and those of $\Sigma(G)$ are the heads of $\Sigma(A)$. Let P and P' be two vertices of G with corresponding dimensions n and n' and let a be an arc with source P and target P'. Note that P and P' are different, otherwise G will have a loop. Let $\mathbf{q} = \langle q_1, \ldots, q_{2^n-1}\rangle$ and $\mathbf{q}' = \langle q'_1, \ldots, q_{2^{n'}-1}\rangle$ be the strings of new variables needed by Σ for P and P', respectively. Some of these new variables occur in $\Sigma(A)$ as heads, and we call them the 'real heads'; the rest will be called 'potential heads'. The ideal case is when all these variables are real heads. Let us now see what kind of arcs correspond to the arc a in the graph $\Sigma(G)$. Note that the positive components of $\Sigma(A)$ and the negative components of the headed boxes are composed from blocks of the form $r_1 \wedge \ldots \wedge r_i$, where all conjuncts are real or potential heads of $\Sigma(A)$. Having this in mind we will associate to the arc a the following sets of arcs from $\Sigma(G)$.

Inherited arcs. Let q_i be a real head which occurs in the negative part of a headed box with a real head q'_j. Then in $\Sigma(G)$ there exists an arc from q_i to q'_j. All such arcs are called **inherited arcs** corresponding to the arc a.

New arcs. Since each head D_i becomes a real head q_i after the transformation Σ, namely $\Sigma(D_i) = \neg q_1 \vee \ldots \vee \neg q_{i-1} \vee q_i$ with additional negative parts $\neg q_1 \vee \ldots \vee \neg q_{i-1}$, then for every $i > 1$, q_i may receive new arcs starting from real heads from the sequence $q_1, \ldots, q_{i-1}$ with targets in q_i. Similarly for the real heads q'_i — they, too, may receive arcs from real heads q'_j , with $j < i$. These arcs have no analogs in G, and that is why we call them **new arcs**.

Thus, we have seen that the inherited and the new arcs are the only arcs in $\Sigma(G)$ related to the arc a. Note that it is possible that each of these sets to be empty, as the following example shows.

Let $D_1 = p \vee q$, $D_2 = p \vee \neg q$, $D_3 = \neg p \vee q$ and let A be the following ICM-formula: $\neg[\alpha]D_2 \vee \neg[\beta](r, \neg D_1)$. A has two propositional complexes, P_1, of dimension 2, built from the variables p, q, and P_2, of dimension 1, built from the variable r. Both complexes are essential, and the dependency digraph contains only one arc a from P_1 to P_2. Then $\Sigma(A)$ is the following inductive formula: $\neg[\alpha](\neg q_1 \vee q_2) \vee \neg[\beta](r, \neg q_1)$ with real heads q_2 and r. One can see that the dependency digraph of $\Sigma(A)$ has no arcs at all. So, the sets of inherited and the new arcs of a are empty.

It is easy to see that the following facts are true.

Fact 1. (i) If there is a new arc in $\Sigma(G)$ from q_i to q_j then $i < j$.

(ii) If there exists a path consisting only of new arcs then the vertices of this path are real heads q_i from $\Sigma(G)$ corresponding to one essential propositional complex of the formula A.

Fact 2. (i) If there exists an inherited arc in $\Sigma(G)$ from q_i to q'_j then there is an arc in G from P to P'.

(ii) Let $q_{i_1} \longrightarrow q_{i_2} \longrightarrow \ldots \longrightarrow q_{i_k}$ be a path of new arcs in $\Sigma(G)$ with vertices corresponding to the essential complex P of A; let $q'_{j_1} \longrightarrow q'_{j_2} \longrightarrow \ldots \longrightarrow q'_{j_{k'}}$ be a path of new arcs corresponding to the essential complex P' of A and let $q_{i_k} \longrightarrow q'_{j_1}$ be an inherited arc from q_{i_k} to q'_{j1}. Then there is an arc from P to P' in G.

The rest of the proof of (i) follows from the following

Internal lemma. The digraph $\Sigma(G)$ has no cycles.

Proof of the Internal lemma. Suppose that there is a cycle C in $\Sigma(G)$. We proceed to arrive at a contradiction.

Case 1: All arcs of C are new arcs. Then it is easy to see by Fact 1(ii) that the vertices of $C = q_{i_1} \longrightarrow q_{i_2} \longrightarrow \ldots \longrightarrow q_{i_k} \longrightarrow q_{i_1}$ consists of real heads corresponding to the new variables of one essential propositional complex P of the formula A. Then by Fact 1(i) we obtain $i_1 < i_2 < \ldots < i_k < i_1$. This implies $i_1 < i_1$ which is a contradiction.

Case 2: The cycle C contains inherited arcs. Consider the following 3 sub-paths of C: $q_{i_1} \longrightarrow q_{i_2} \longrightarrow \ldots \longrightarrow q_{i_k}$ of new arcs, the inherited arc $q_{i_k} \longrightarrow q'_{j_1}$, and the new arcs $q'_{j_1} \longrightarrow q'_{j_2} \longrightarrow \ldots \longrightarrow q'_{j_{k'}}$. By Fact 2(ii) this implies that there exists an arc between the corresponding propositional complexes P and P' in G. Going further consider the 3 sub-paths $q'_{j_1} \longrightarrow q'_{j_2} \longrightarrow \ldots \longrightarrow q'_{j_{k'}}$, the inherited arc $q'_{j_{k'}} \longrightarrow q''_{m_1}$, and the new arcs $q''_{m_1} \longrightarrow q''_{m_2} \longrightarrow \ldots \longrightarrow q''_{i_{k''}}$. Again by fact 2(ii) this implies that there exists an arc between the corresponding propositional complexes P' and P'' in G. Reasoning in this way we obtain by induction a path $P \longrightarrow P' \longrightarrow P'' \longrightarrow \ldots \longrightarrow P$ which is a cycle in G. Since G has no cycles, this is the intended contradiction — the lemma is proved.

(ii) is an immediate corollary of Lemma 5.3(iii). $\Box$

6. The algorithm $\mathsf{SQEMA}^{\mathrm{sub}}$

In this section we give an informal introduction to $\mathsf{SQEMA}^{\mathrm{sub}}$, followed by a more formal description. We illustrate the algorithm with some examples and prove some theorems about its meta-properties.

6.1. The algorithm $\mathsf{SQEMA}^{\mathrm{sub}}$, informally

In Example 2.2 we considered the formula

$$\varphi = [\mathbf{3}](\neg[\mathbf{1}](p_1 \vee \neg p_2), \neg[\mathbf{1}](\neg p_1 \vee p_2), \langle 2\rangle((p_1 \vee \neg p_2), (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2))),$$

on which SQEMA fails. The reason for this failure was that, in any attempt to eliminate the variables p_1 and p_2 *one by one*, the algorithm was unable to apply the Ackermann-rule since one of the equations was neither positive nor negative with respect to the chosen variable. This example indicates a weakness of the Ackermann lemma and suggests that it could be strengthened in order to handle such cases. One possible solution would be to strengthen the Ackermann-rule. Indeed, in [28, 29, 30] some generalizations of the Ackermann lemma, by means of which one can eliminate several variables at once, were considered. Particularly, a generalized Ackermann lemma was given which is sufficient to eliminate all ordinary variables of one given complex variable in an ICM-formula at once, and finally to find the corresponding first-order frame condition. Now, in step 7 of example 2.2 both variables are ready to be simultaneously eliminated according to (a rule based upon) one such generalization of Ackermann lemma. Following this route would, however, require a serious reconstruction of the algorithm SQEMA. Moreover, all theorems relating to SQEMA, e.g., its correctness and the canonicity of the reducible formulae, would have to be re-proved for the new algorithm so obtained.

That is why we opt for another, rather more modular approach, based on the method of reversible substitutions. We have already shown how, with this method, each ICM-formula can be transformed by a suitable substitution-like transformation (viz. the transformation Σ developed in section 5.3) into an inductive formula. We know from [7] that SQEMA succeeds on all inductive formulae, so the main idea is to introduce an extension of SQEMA, called $\mathsf{SQEMA}^{\mathrm{sub}}$, composed of two subprograms. The original (polyadic) algorithm SQEMA itself constitutes the first of these subprograms, while the second, to be called SUB, deals with the substitution-like transformation Σ. $\mathsf{SQEMA}^{\mathrm{sub}}$ will inherit all the important properties of SQEMA, viz. the local first-order definability and d-persistence (hence, canonicity) of the formulae reducible by it. Moreover, it will succeed not only on all inductive formulae, but also on all ICM-formulae.

Let us now describe informally how $\mathsf{SQEMA}^{\mathrm{sub}}$ works. The input formula A is first sent to the subprogram SQEMA. If SQEMA succeeds it reports SUCCESS and outputs the corresponding first-order condition of A. If SQEMA does not succeed then $\mathsf{SQEMA}^{\mathrm{sub}}$ runs SUB to find (nondeterministically) a complex normal form A' of the input formula A, then applies Σ to A', and then send the result back to SQEMA. This cycle is repeated until SQEMA succeeds or SUB cannot produce any new complex normal form of A. If all (finitely many) possible complex normal forms have been generated and SQEMA has not succeeded on any of them, then $\mathsf{SQEMA}^{\mathrm{sub}}$ reports FAIL and halts.

6.2. Description of $\mathsf{SQEMA}^{\mathrm{sub}}$

Here is a formal description of $\mathsf{SQEMA}^{\mathrm{sub}}$.

Algorithm $\mathsf{SQEMA}^{\mathrm{sub}}(\varphi)$ This is the main body of the algorithm. It takes as input an $\mathcal{L}_\tau$-formula φ, for which it either returns a local first-order frame correspondent, or reports failure.

(1) Call $\mathsf{SQEMA}(\varphi)$.
If $\mathsf{SQEMA}(\varphi)$ returns a first-order formula F
then return F and terminate,
else if $\mathsf{SQEMA}(\varphi)$ returns FAIL, proceed to step 2.

(2) Call procedure $\mathsf{SUB}(\varphi)$.
If $\mathsf{SUB}(\varphi)$ returns a first-order formula F
then return F and terminate,
else if $\mathsf{SUB}(\varphi)$ returns, return FAIL and terminate.

Procedure $\mathsf{SUB}(\varphi)$ This procedure takes as input an $\mathcal{L}_\tau$-formula φ, for which it either returns a local first-order frame correspondent, or reports failure.

(1) Initialize procedure Complex normal form with φ.

(2) Repeat

(2.1) Request a complex normal form φ' of φ from Complex normal from. Procedure Complex normal from returns such a complex normal form, together with an associated list

$$\overline{\mathsf{StringDVar}} = \langle \mathsf{StringDVar}(C_1), \ldots, \mathsf{StringDVar}(C_l) \rangle$$

encoding the propositional complexes $P_1, P_2, \ldots, P_l$ of φ', and a list

$$\overline{\mathsf{StringNewVar}} = \langle \mathsf{StringNewVar}(C_1), \ldots, \mathsf{StringNewVar}(C_l) \rangle$$

of strings of new variables.
If procedure Complex normal from returns 'DONE'
then return FAIL and terminate,
else proceed to (2.2).

(2.2) Call procedure Translation $\Sigma(\varphi', \overline{\mathsf{StringDVar}}, \overline{\mathsf{StringNewVar}})$.
This procedure returns a $\mathcal{L}$-formula φ''.

(2.3) Call $\mathsf{SQEMA}(\varphi'')$.
If $\mathsf{SQEMA}(\varphi)$ returns a first-order formula F
then return F and terminate.

Procedure Complex normal form This procedure is initialized with a formula A, for which it then successively produces all possible complex normal forms, as requested.

Initialization The procedure initializes with a formula A by performing the following four steps:

(1.1) Determine the list of Boolean blocks $B_1, \ldots, B_k$ of A and represent A as built from these blocks: $A = A(B_1 \ldots, B_k)$.

(1.2) Partition the set of Boolean blocks into neighbourhood classes $C_1, \ldots, C_l$ and determine the sets of variables $\mathsf{Var}(C_i)$, $i = 1, \ldots, l$, where each set $\mathsf{Var}(C_i)$ is considered as a string of different variables, in a fixed order.

(1.3) For each set of variables $\mathsf{Var}(C) = \langle p_1, \ldots, p_n \rangle$ produce all elementary disjunctions $\mathsf{DVar}(C)$ over $\mathsf{Var}(C)$ and order the variables in each disjunction in the same way as the order of the variables in $\mathsf{Var}(C)$.

(1.4) For each set of variables $\mathsf{Var}(C) = \langle p_1, \ldots, p_n \rangle$ choose a new set of variables $\mathsf{NewVar}(C)$ with cardinality 2^n, such that all these sets, and the sets $\mathsf{Var}(C)$, are pairwise disjoint.

Complex normal forms When a complex normal form for the formula A initialized with is requested, the following steps are performed:

(1.5.1) Choose nondeterministically a *new order* of the elementary disjunctions in each set $\mathsf{DVar}(C)$ and produce the string $\mathsf{StringDVar}(C) = \langle D_1, \ldots, D_{2^n} \rangle$ according to the chosen order.

(1.5.2) Reorder the variables in the set $\mathsf{NewVar}(C)$ and produces the string $\mathsf{StringNewVar}(C)$ in order to obtain a one-one correspondence between the strings $\mathsf{StringDVar}(C)$ and $\mathsf{StringNewVar}(C)$. (If D_i is the i-th element of $\mathsf{StringDVar}(C)$ then the i-th element of $\mathsf{StringNewVar}(C)$ will be denoted by q_i.)
If the new order cannot be generated, i.e., all possible orders have been attempted
then return DONE,
else proceeds to (1.6.1).

(1.6.1) For each Boolean block B of A produce its conjunctive normal form B' as follows: choose the neighbourhood class C such that $B \in C$ and define the conjunctive normal form by the elementary disjunctions from the string $\mathsf{StringDVar}(C)$, such that the conjuncts follow the order of $\mathsf{StringDVar}(C)$.

(1.6.2) Produce the complex normal form A' of A, replacing each Boolean block B of A by its conjunctive normal form B', i.e., $A' = A(B'_1, \ldots, B'_k)$.

(1.6.3) Return the triple $(A', \overline{\mathsf{StringDVar}}, \overline{\mathsf{StringNewVar}})$.

Remark 6.1. Each time (1.5.1) is called, it produces a new order of the elementary disjunctions of all sets $\mathsf{DVar}(C) = \langle D_1, \ldots, D_{2^n} \rangle$ and (1.5.2) produces the corresponding order of the new variables in the sets $\mathsf{NewVar}(C)$. The execution of the next steps of the procedure depends on that new order. Note also that the number N of possible orders of the sets of disjunctions is finite and can be obtained as $N = \prod_{i=1}^{l} 2^{n_i}!$ where n_i is the cardinality of the set $\mathsf{Var}(C_i)$, for $i = 1, \ldots, l$.

Procedure Translation Σ This procedure receives a triple $(A', \overline{\mathsf{StringDVar}}, \overline{\mathsf{StringNewVar}})$, such as is produced by procedure Complex normal form, as input. This triple contains all the needed data to compute the new modal formula $\Sigma(A')$.

Proceed recursively top-down:

$\Sigma(A') = A(\Sigma(B'_1), \ldots, \Sigma(B'_k))$, computing $\Sigma(B')$ for each component $B' = B'_i$, as follows:

Choose the neighbourhood class C such that $B' \in C$.

Choose the corresponding string $\mathsf{StringNewVar}(C)$.

Compute $\Sigma(B')$ according to the following rules (see the definition of Σ in Section 5.3):

(2.1) If B' is in the form D_i, $i = 1, \ldots, 2^n - 1$, then $\Sigma(D_1) = q_1$, for $1 < i < 2^n$, $\Sigma(D_i) = \neg q_1 \vee \ldots \vee \neg q_{i-1} \vee q_i$, and $\Sigma(D_{2^n}) = \neg q_1 \vee \ldots \vee \neg q_{2^n-1}$.

(2.2) If B' is in the form $D_1 \wedge D_2 \wedge \ldots \wedge D_i$, $i = 1, \ldots, 2^n - 1$, then $\Sigma(B') = q_1 \wedge q_2 \wedge \ldots q_i$, and $\Sigma(B') = \perp$ if $i = 2^n$.

(2.3) If B is in neither of these forms, then $B = D_{i_1} \wedge D_{i_2} \ldots D_{i_j}$, $j > 1$, and in this case we put $\Sigma(B) = \Sigma(D_{i_1}) \wedge \ldots \wedge \Sigma(D_{i_j})$, considering conjuncts as in (2.1).

Return the obtained formula $\Sigma(A')$.

6.3. Examples

We now illustrate $\mathsf{SQEMA}^{\mathrm{sub}}$ with two examples.

Example 6.1. Let $\varphi := \Diamond\Box(p \to q) \wedge \Diamond\Box(q \to p) \to \Diamond\Box(p \leftrightarrow q)$. When φ is given to $\mathsf{SQEMA}^{\mathrm{sub}}$ as input, it is first passed to the subroutine SQEMA. As the reader can check, SQEMA will fail on φ, as it will be unable to solve for either p or q. Thus φ is passed to procedure SUB. SUB calls and initializes Complex normal form with φ. When Complex normal form initializes with φ, step (1.1) determines the boolean blocks $B_1 = (p \to q)$, $B_2 = (q \to p)$ and $B_3 = (p \leftrightarrow q)$ of φ and represents φ as $\varphi = \Diamond\Box B_1 \wedge \Diamond\Box B_2 \to \Diamond\Box B_3$. Step (1.2) determines that $C_1 = \{B_1, B_2, B_3\}$ and that $\mathsf{Var}(C_1) = \langle p, q \rangle$. Step (1.3) determines that $\mathsf{DVar}(C_1) = \langle (p \vee q), (p \vee \neg q), (\neg p \vee q), (\neg p \vee \neg q) \rangle$. Step (1.4) determines $\mathsf{NewVar}(C_1) = \langle q_1, q_2, q_3, q_4 \rangle$.

When SUB requests a complex normal form of φ from Complexnormalform, the latter procedure will, in response to one of these requests, return a triple $(A', \overline{\mathsf{StringDVar}}, \overline{\mathsf{StringNewVar}})$ with $A' = \Diamond\Box(\neg p \vee q) \wedge \Diamond\Box(p \vee \neg q) \to \Diamond\Box((\neg p \vee q) \wedge (p \vee \neg q))$, $\overline{\mathsf{StringDVar}} = \langle\langle (\neg p \vee q), (p \vee \neg q), (p \vee q), (\neg p \vee \neg q) \rangle\rangle$, and $\overline{\mathsf{StringNewVar}} = \langle\langle q_1, q_2, q_3, q_4 \rangle\rangle$. SUB passes this triple to Translation Σ, which produces and returns the formula $\Diamond\Box q_1 \wedge \Diamond\Box(\neg q_1 \vee q_2) \to \Diamond\Box(q_1 \wedge q_2)$. This is an inductive formula, so when SUB passes it to SQEMA, the latter subroutine will successfully compute a first-order frame correspondent for it, namely $Rxy \wedge Rxz \to \exists u(Rxu \wedge \forall v(Ruv \to (Ryv \wedge Rzv)))$. Thus $\mathsf{SQEMA}^{\mathrm{sub}}$ will return this first-order formula and terminate successfully.

Example 6.2. Consider the formula $\varphi_2 = [\mathbf{2}](\neg[\mathbf{1}](\neg[\mathbf{1}]p \vee p), p \wedge [\mathbf{1}]\bot)$. As was shown in example 2.4 in [7], SQEMA fails on this formula, despite its being locally first-order definable. Thus, when $\mathsf{SQEMA}^{\mathrm{sub}}$ is run on this formula, the initial execution of SQEMA on it will fail. Note that φ_2 is already in complex normal form and that, in fact, this is its only complex normal form. Thus, passing it to Complex normal form and sending the result to Translation Σ will not produce a different formula. Hence $\mathsf{SQEMA}^{\mathrm{sub}}$ will fail on φ_2.

6.4. Properties of $\mathsf{SQEMA}^{\mathrm{sub}}$

Theorem 6.1. (Correctness)
If $\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds on an input formula A, then A is a local frame-correspondent of the returned first-order condition.

Proof:
Let $F(x)$ be the first-order formula returned by $\mathsf{SQEMA}^{\mathrm{sub}}$ when run on A. If the subroutine SQEMA succeeded on A without the procedure SUB being called, then the claim follows from Theorem 1.1. On the other hand, if SUB was called, then it returned a formula $\Sigma(A')$ such that A' is a complex normal form of A, which was then passed again to SQEMA, which, in turn, returned $F(x)$. Again by theorem 1.1, $F(x)$ is a local first-order equivalent of the formula $\Sigma(A')$. By Lemma 5.3 we have that A, A', and $\Sigma(A)$ are locally equivalent, which implies that $F(x)$ is a local first-order equivalent of A. □

Theorem 6.2. (Canonicity)

1. If $\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds on an $\mathcal{L}_\tau$-formula A, then A is locally persistent with respect to the class of all descriptive τ-frames.

2. If $\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds on an $\mathcal{L}_{r(\tau)}$-formula A, then A is locally persistent with respect to the class of all reversive descriptive τ-frames.

Proof:
We will prove case 1; case 2 is analogous. If $\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds on A without calling the subprogram SUB, then it must be the case that SQEMA succeeds on A, and hence the result follows from Theorem 1.2. On the other hand, if SUB is called, then at some stage subroutine SQEMA succeeds on a formula $\Sigma(A')$, such that A' is a complex normal form of A. Then, again by Theorem 1.2 $\Sigma(A')$ is locally persistent with respect to the class of all descriptive τ-frames. By Lemma 5.3 A, A' and $\Sigma(A)$ are locally equivalent, which implies that A is locally persistent with respect to the class of all descriptive τ-frames. □

Theorem 6.3. (Completeness)
$\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds on all ICM-formulae.

Proof:
Let A be an ICM-formula and suppose that the subprogram SQEMA does not succeed on A. Hence procedure SUB is called. Since A is itself in a complex normal form, A will at some stage be passed to

procedure Translation Σ without change, and hence $\Sigma(A)$ will be sent to SQEMA. By proposition 5.1, $\Sigma(A)$ is an inductive formula. Then, by theorem 1.3, the subprogram SQEMA will succeed on $\Sigma(A)$, and hence $\mathsf{SQEMA}^{\mathrm{sub}}$ will succeed on A. □

7. Concluding remarks

We have explored the use of reversible substitutions for transforming a modal formula into a locally equivalent Sahlqvist-like one, for which a suitable adaptation of the algorithm SQEMA can compute a local first-order equivalent and prove d-persistence, and hence canonicity. In particular, we have extended the algorithm SQEMA with a special module for computing such substitutions and have shown that the resulting algorithm $\mathsf{SQEMA}^{\mathrm{sub}}$ succeeds on all inductive complex formulae, thus extending essentially the scope of SQEMA. An implementation of $\mathsf{SQEMA}^{\mathrm{sub}}$ is currently being developed.

The algorithm $\mathsf{SQEMA}^{\mathrm{sub}}$ does not exhaust the potential of the method of reversible substitutions. Finding a suitable substitution, however, is generally a rather non-trivial task and little is known about its applicability beyond the class of complex inductive formulae. Furthermore, even if one can guess a suitable substitution for a given formula, the question if it is reversible seems difficult, and at present we do not know if it is algorithmically decidable. Nevertheless the problem of finding broader effective classes of reversible substitutions which may extend the applicability of SQEMA seems sensible, and we formulate it as one of the important open problems. In the present version $\mathsf{SQEMA}^{\mathrm{sub}}$ transforms the input formula into formula with an exponentially greater number of propositional variables, which is a rather expensive task. One of our future plans is to modify $\mathsf{SQEMA}^{\mathrm{sub}}$ such that the subprogram SUB preserves the number of variables of the input formula. Another open problem is the comparison of $\mathsf{SQEMA}^{\mathrm{sub}}$ with other algorithms like SCAN and DLS and even with SQEMA: we know that SQEMA fails on some ICM-formulae and the problem is to find a large subclass of ICM-formulae to which SQEMA succeeds and to see if the whole class of ICM-formulae can be reduced by suitable reversible substitutions to this subclass. As for SCAN we know that it succeeds on all Sahlqvist formulae [18] and also on some ICM-formulae (for instance the formula A from Section 2.2) but it is not known whether, for instance, SCAN is complete for the inductive (ordinary, or complex) modal formulae. In [6] there are examples of formulae for which SQEMA succeeds but neither SCAN nor DLS does, which show that SQEMA, and hence $\mathsf{SQEMA}^{\mathrm{sub}}$ are incomparable with SCAN and DLS in terms of the scope of application.

Acknowledgements

Thanks are due to the anonymous referees for the very careful reading of the paper and pointing out some inaccuracies, and especially for their useful suggestions to make some technical parts more readable and intuitive. The work of Dimiter Vakarelov is supported by the contract IM/1510 with the Bulgarian Ministry of Science and Education. Also, part of this research was done during his visit to Johannesburg, funded by the University of Johannesburg and the National Research Foundation of South Africa, which have also supported financially the research of the first two authors.

References

[1] W. Ackermann. Untersuchung über das Eliminationsproblem der mathematischen Logic. *Mathematische Annalen*, 110:390-413, 1935.

[2] P. Blackburn, M. de Rijke, and Y. Venema. **Modal Logic**. Cambridge University Press, 2001.

[3] L. A. Chagrova. An undecidable problem in correspondence theory. *Journal of Symbolic Logic*, 56:1261-1272, 1991.

[4] W. Conradie. Completeness and correspondence in hybrid logic via an extension of SQEMA. *Electronic Notes in Theoretical Computer Science*, 231:175 – 190, 2009. Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007).

[5] W. Conradie, V. Goranko, and D. Vakarelov: Elementary canonical formulae: a survey on syntactic, algorithmic, and model-theoretic aspects, In: R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing (editors). **Advances in Modal Logic, vol. 5**, Kings College London, 2005, pages 17-51.

[6] W. Conradie, V. Goranko, and D. Vakarelov. Algorithmic correspondence and completeness in modal logic I : The core algorithm SQEMA. *Logical Methods in Computer Science*, 2(1:5), 2006.

[7] W. Conradie, V. Goranko, and D. Vakarelov. Algorithmic correspondence and completeness in modal logic II. Polyadic and hybrid extensions of the algorithm SQEMA. *Journal of Logic and Computation*, 16:579–612, 2006.

[8] W. Conradie, V. Goranko, and D. Vakarelov. Algorithmic correspondence in modal logic. V : Recursive extensions of the algorithm SQEMA, 2008, submitted.

[9] 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.

[10] P. Doherty, W. Łukaszewicz, and A. Szałas, Computing circumscription revisited, *Journal of Automated Reasoning*, 1997, 18(3):297–336.

[11] T. Engel. Quantifier Elimination in Second-Order Predicate Logic, Diploma Thesis, MPI, Saarbrücken, 1996.

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

[13] D. M. Gabbay, R. Schmidt, and A. Szałas. **Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications**. Studies in Logic, vol. 12, College Publications, 2008.

[14] D. Georgiev, T. Tinchev and D. Vakarelov. SQEMA - an algoritm for computing first-order equivalents in modal logic: a computer realization. In: **Pioneers of Bulgarian Mathematics**, International Conference dedicated to Nicola Obrechkoff and Lubomir Tschakaloff, Sofia, July 8–10, 2006. Abstracts.

[15] V. Goranko and D. Vakarelov. Sahlqvist Formulae Unleashed in Polyadic Modal Languages, In: F. Wolter et al (eds.), **Advances in Modal Logic, vol. 3**, World Scientific, Singapore, 2002, pp. 221–240.

[16] V. Goranko, and D. Vakarelov. Sahlqvist Formulae in Hybrid Polyadic Modal Languages. *Journal of Logic and Computation*, 11(5):737–254, 2001.

[17] V. Goranko, and D. Vakarelov. Elementary Canonical Formulae: Extending Sahlqvist Theorem. *Annals of Pure and Applied Logic*, 141(1-2):180–217, 2006.

[18] V. Goranko, U. Hustadt, R. Schmidt, and D. Vakarelov. SCAN is complete for all Sahlqvist formulae, in: **Relational and Kleene-Algebraic Methods in Computer Science (Proc. of RelMiCS 7)**, LNCS 3051, Springer, 2004, 149–162.

[19] J. Gustafsson. An Implementation and Optimization of an Algorithm for Reducing Formulae in Second-Order Logic. Technical Report LiTH-MAT-R-96-04. Dept. of Mathematics, Linkoping University, Sweden, 1996.

[20] A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory, in: E. Orlowska (ed.), **Logic at Work, Essays dedicated to the memory of Helena Rasiowa**, Springer Physica-Verlag, 89–108, 1998.

[21] A. Nonnengart, H.-J. Ohlbach, and A. Szałas, Elimination of Predicate Quantifiers, in: H. J. Ohlbach and U. Reyle (eds.), **Logic, Language and Reasoning: Essays in honour of Dov Gabbay**, Part I, Kluwer, 1999, 159–181.

[22] H. Sahlqvist. Correspondence and completeness in the first and second-order semantics for modal logic, in: S. Kanger (ed.), **Proc. of the 3rd Scandinavial Logic Symposium, Uppsala 1973**, North-Holland, Amsterdam, 1975, 110–143.

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

[24] A. Szałas, Second-order quantifier elimination in modal contexts, in: S. Flesca and G. Ianni (eds.), **JELIA'02**, LNAI 2424, Springer-Verlag, 2002, pp. 223–232.

[25] B. ten Cate, M. Marx, and P. Viana. Hybrid logics with Sahlqvist axioms. *Logic Journal of the IGPL*, 13(3): 293-300, 2005.

[26] D. Vakarelov, Modal definability in languages with a finite number of propositional variables, and a new extention of the Sahlqvist class. In: P. Balbiani, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev (eds), **Advances in Modal Logic, vol. 4**, King's College Publications, London, 2003, pp. 495–518.

[27] D. Vakarelov. Exstended Sahlqvist Formulae and Solving Equations in Modal Algebras, Abstract, 12-th International Congress of Logic Methodology and Philosophy of Science, August 7–13, 2003, Oviedo, Spain,Program, p. 33.

[28] D. Vakarelov, On a generalization of the Ackermann lemma for computing first-order equivalents of modal formulae, abstract, TARSKI Workshop, March 11 - 13, 2003, Toulouse, France.

[29] D. Vakarelov. Modal definability, solving equations in modal algebras and generalization of the Ackermann lemma. In the Proceedings of 5th Panhellenic Logic Symposium, Athens, July 25-28, 2005, 182–189.

[30] D. Vakarelov. On a generalization of the Ackermann lemma for computing first-order equivalents of modal formulae. In: Logic Colloquium 2005, July 28- August 5, Athens, Greece, Abstracts, p. 123.

[31] D. Vakarelov. Solving recursive equations in complete modal algebras with applications to modal definability. In: **Pioneers of Bulgarian Mathematics**, International Conference dedicated to Nicola Obrechkoff and Lubomir Tschakaloff, Sofia, July 8–10, 2006. Abstracts.

[32] D. Vakarelov, A recursive generalizations of Ackermann lemma with applications to μ-definability. In 6th Panhellenic Logic Symposium PLS-2007, 5–8 July, Volos, Greece, Extended abstacts, G.Kaouri and S. Zahos Eds, pp. 133–137.

[33] J. F. A. K. van Benthem. **Modal Logic and Classical Logic**. Bibliopolis, Napoli, 1983.

[34] J. F. A. K. van Benthem. Some Correspondence Results in Modal Logic, Tech. Report 74-05, Mathematisch Instituut, Univ. van Amsterdam, 1974.

[35] J. F. A. K. van Benthem. Minimal predicates, fixed-points, and definability. *Journal of Symbolic Logic*, 70:3:696-712, 2005.

[36] J. F. A. K. van Benthem. Modal frame correspondence and fixed-points. *Studia Logica*, 83: 133-155, 2006.