On Correspondence Between Modal and Classical Logic: Automated Approach

Andrzej SZAŁAS

MPI–I–92–209 March 1992

Authors' Addresses

until May 1992:

Max–Planck–Institut für Informatik
Im Stadtwald
D-W-6600 Saarbrücken
Germany
szalas@cs.uni-sb.de

later:

Institute of Informatics
University of Warsaw
02-097 Warsaw 59
ul. Banacha 2
Poland
aszalas@plearn.bitnet

Acknowledgements

I would like to thank A. Nonnengart and H. J Ohlbach for many discussions on automated deduction in modal logics as well as for many helpful remarks directly concerning this paper.

Abstract

The current paper is devoted to automated techniques in correspondence theory. The theory we deal with concerns the problem of finding classical first-order axioms corresponding to propositional modal schemas. Given a modal schema and a semantics based method of translating propositional modal formulas into classical first-order ones, we try to derive automatically classical first-order formula characterizing precisely the class of frames validating the schema. The technique we consider can, in many cases, be easily applied even without any computer support.

Although we mainly concentrate on Kripke semantics, the technique we apply is much more general, as it is based on elimination of second-order quantifiers from formulas. We show many examples of application of the method. Those can also serve as new, automated proofs of considered correspondences.

We essentially strengthen the considered elimination technique. Thus, as a side-effect of this paper we get a stronger elimination based method for proving a subset of second-order logic.

Keywords

automated theorem proving, correspondence axioms, modal logics, semantics based translation

1 Introduction

A great deal of attention has been devoted to automated theorem proving during the last two decades. The revolutionary resolution method introduced by Robinson in 1968 focussed the interest of researchers and software designers mainly on classical first-order logic. This resulted in huge amount of implementation-oriented optimizations and improvements of the resolution method and had also influence on other theorem proving techniques. No wonder than that the classical first-order logic occupies now the central position in the field of automated theorem proving. On the other hand, classical first-order logic has some disadvantages, widely discussed by philosophers and computer scientists. The following ones are perhaps the most serious from the point of view of artificial intelligence techniques:

- classical implication does not perfectly correspond to natural language implication
- classical first-order logic is an extensional one, i.e. the logical value of complex formulas depend solely on values of their atomic components.

Thus classical-first order logic is not very natural when one uses implication and/or wants to express intentional natural language phenomena.

Let us discuss this more precisely. Consider the following two sentences:

1. if water is dry then John will be a president of USA
2. it is possible that John has walked to his office today.

The first sentence is true (in classical logic), since its assumption is false, whilst, in natural language, one usually considers similar sentences senseless. The truth of the second sentence does not only depend on truth of the sentence *John has walked to his office today*, as we do not intend to say that this fact indeed took place. Even if John has not walked to his office today, this still might be possible. That, however, depends on many factors, like whether John lives in walking distance from the office or whether John is able to walk at all.

Modal logics have been proposed as a remedy to those difficulties[1]. They were intensively studied during our century. There were some successful attempts to mechanize the reasoning in those logics (cf. e.g. methods described in [3, 5, 7], just to mention a few of them). When attempting to mechanize modal reasoning, one faces the choice between designing the whole system from the beginning or, alternatively, using existing theorem provers for classical first-order logic. The first way usually requires much effort and is time consuming. On the other hand, as motivated above, there are many good reasons to use the existing theorem provers whenever possible. One of the best ways to achieve this goal is to interpret modal logics in terms of classical first-order logic. Such an interpretation can be done by translation of modal logic into classical logic (see e.g. [7, 8]).

Let us denote classical first-order logic by $\mathcal{L}_{\mathcal{I}}$. Suppose that we are going to mechanize theorem proving in some modal logic, say $\mathcal{L}_{\mathcal{M}}$. According to above discussion, in what

[1] there is a rich literature on modal logics - for their history, further motivations and references see e.g. [2]

follows we shall consider methods of translating $\mathcal{L}_{\mathcal{M}}$ into $\mathcal{L}_{\mathcal{I}}$. In order to make such a translation useful, one has to ensure the following two properties:

- if the translated formula is valid, then also original modal formula is valid[2] (*soundness* of translation)
- if modal formula is valid, then it remains valid after translation (*completeness* of translation).

We thus need a translation which associates classical first-order formula to a given modal formula. This, however, is only a part of the problem. Namely properties of modal operators are usually given by means of a set of postulates, i.e. axiom schemas. For instance, if one says that modal operator satisfies $\Box p \to p$, what he in fact has in mind is that *for any* modal formula p, $\Box p$ implies p. Those postulates must also have their counterparts in translation mechanism. Such counterparts are usually called correspondence axioms (cf. e.g. [4, 8]). Our paper is then devoted to automatic search for classical first-order axioms corresponding to modal postulates. Before we formulate our goal more formally, let us consider the following example (for a survey of correspondence theory see e.g. [8]).

Example 1.1 Let us consider modal logic with modality $\Box$. The modality can have various possible readings, like *always, necessary, known*, etc. According to the reading, the modality has some properties expressed by postulates. E.g. if we read $\Box$ as *necessary*, we can have the following postulates, where, as usual, $\Diamond$ is defined as $\neg\Box\neg$:

(i) $\Box p \to p$

(ii) $\Box p \to \Box\Box p$

(iii) $p \to \Box\Diamond p$.

Let us now consider Kripke-like translation of modal formulas, which translates $\Box q$ into

$$\forall x.\forall y.R(x,y) \to Q(y),$$

where R represents the accessibility relation provided by Kripke structures. Then postulates (i) - (iii) correspond to the following classical first-order properties of relation R:

(i') $\forall x.R(x,x)$ - reflexivity

(ii') $\forall x,y,z.R(x,y) \to (R(y,z) \to R(x,z))$ - transitivity

(iii') $\forall x,y.R(x,y) \to R(y,x)$ - symmetry.

Thus in order to prove that modal formula, say q, is a theorem of the least normal logic[3] containing schemas (i) - (iii), one can translate the formula into classical first-order formula q' and try to prove that q' follows from (i') - (iii') in classical logic. □

[2] one can also consider dual case of translations preserving satisfiability of formulas. We shall, however, prefer translations preserving validity

[3] for definition of normal logic see the next section

An algorithm (called SCAN) for automatic synthesis of correspondence axioms, based on first-order resolution principle, was recently proposed by D. Gabbay and H. J. Ohlbach in [4]. The SCAN algorithm is a break through in mechanizing the theory of correspondence between modal and classical logic. On the other hand, SCAN is partial, i.e. does not always stop even if the answer exists (cf. [4]). Since we are concerned with automated techniques, we find this feature a disadvantage of the SCAN algorithm[4]. In this paper we shall present another algorithm. The new algorithm always halts. Moreover, one can construct examples[5] when our algorithm gives some first-order formula as an answer and, for the same input, SCAN algorithm loops.

The algorithm we present essentially differs from SCAN, as it is based on techniques discovered mainly by Ackermann in connection to classical elimination problem - cf. [1]. Those techniques allow, in many cases, to eliminate second-order quantifiers from formulas of second-order logic. On the other hand, when considering schemas of formulas, we basically deal with second-order quantification. For instance, schema $\Box p \rightarrow p$ corresponds to second-order formula $\forall p.\Box p \rightarrow p$. Thus the main problem to be solved is to eliminate second-order quantification.

For the sake of readability of the algorithm we shall first present it in a simplified form (cf. section 4). The algorithm of section 4 can still be strengthened so to accept more formulas. The possible directions of strengthening the algorithm are discussed in section 5. Note that we introduce there a factorization principle that essentially strengthens the elimination technique of Ackermann.

2 Modal logics

Let us now define modal logics. It should be emphasized here that, for the sake of simplicity, we make the following restrictions:

- we do not consider multimodal logics
- we consider one-argument modalities only
- we consider Kripke semantics only
- we do not consider first-order versions of modal logics
- we consider only normal logics.

However, as it will follow from our considerations, the extension of our approach to cover the multimodal case and the case of many-argument modalities as well as first-order quantifiers and non-normal logics is straightforward. The method is also applicable to other kinds of semantics as well. For a discussion of mentioned extensions see also final remarks.

In what follows by $\top$ and $\bot$ we shall denote truth values, *true* and *false*, respectively. Let us now define the notion of Kripke frame and Kripke model.

[4]note, however, that when SCAN loops, it is sometimes possible to observe some patterns that, put together, give us a correspondence axiom of infinitary logic $L_{\omega_1\omega}$

[5]we admit however, that examples we know are rather artificial ones

Definition 2.1 Let V be an enumerable set of propositional variables.

- By *Kripke frame* (frame, in short) we shall mean any pair $\langle W, R\rangle$, where
  - W is any set, called the set of *worlds*
  - R is a binary relation defined on W, called the *accessibility relation*.
- By *Kripke model* we shall mean any triple $\langle \mathcal{K}, w, v\rangle$, where
  - $K = \langle W, R\rangle$ is a Kripke frame
  - $w \in W$ is a distinguished element of W, called the *actual* world
  - $v : V \times W \longrightarrow \{\top, \bot\}$ is a mapping associating truth value to a given propositional variable in a given world.

We shall say that class $\mathcal{B}$ of frames *validates* modal formula p iff any Kripke model with frame in $\mathcal{B}$ satisfies p. □

We are now ready to define the notion of modal logic.

Definition 2.2 By *(propositional) modal logic* we shall mean triple $\mathcal{L}_M = \langle \mathcal{F}, \mathcal{C}, \models\rangle$, where

- $\mathcal{F}$ is the set of formulas, i.e. expressions built from propositional variables by applying classical propositional connectives $\neg$, $\vee$, $\wedge$[6] and one-argument modal operators $\Box$, $\Diamond$
- $\mathcal{C}$ is a subclass of the class of Kripke models
- $\models$ is a *satisfiability relation* defined as follows:
  - $\mathcal{K}, w, v \models p$ iff $v(p, w) = \top$, where p is a propositional variable
  - $\mathcal{K}, w, v \models \neg p$ iff not $\mathcal{K}, w, v \models p$
  - $\mathcal{K}, w, v \models p \vee q$ iff $\mathcal{K}, w, v \models p$ or $\mathcal{K}, w, v \models q$
  - $\mathcal{K}, w, v \models p \wedge q$ iff $\mathcal{K}, w, v \models p$ and $\mathcal{K}, w, v \models q$
  - $\mathcal{K}, w, v \models \Box p$ iff for any w' such that $R(w, w')$, $\mathcal{K}, w', v \models p$
  - $\mathcal{K}, w, v \models \Diamond p$ iff there is w' such that $R(w, w')$ and $\mathcal{K}, w', v \models p$.

By *normal* modal logic we shall mean any modal logic which is an extension of classical propositional logic and such that

- $\models \Box(p \to q) \to (\Box p \to \Box q)$
- $\models p$ implies $\models \Box p$. □

[6] the choice of just those connectives allows us to simplify the presentation. We shall, of course, use also other connectives – those should always be understood as abbreviations.

The following fact is well-known (cf. e.g. [2]).

Proposition 2.3 The set of all tautologies of the least normal logic is precisely the set of those formulas which are valid in any Kripke model. □

We are now ready to define the notion of correspondence axioms.

Definition 2.4 Let S be a modal schema. We shall say that classical first-order formula A is a *correspondence axiom* for S iff the class of classical first-orders models of A is precisely the class of frames validating any formula obtained from schema S by substituting propositional variables by modal formulas. □

Note that, according to the above definition, we shall consider only normal logics. We also need the following definition.

Definition 2.5 We shall say that occurrence of predicate P in formula A is *positive* iff P appears under no negation sign (in the place in question). Dually, we shall say that occurrence of P in A is *negative* iff the occurrence P has the form $\neg P$ and (this) $\neg P$ appears under no negation sign in A. □

3 Semantics based translation

As one can easily notice, the definition 2.2 suggests translations summarized in the table below.

Modal formula	Translated formula	
	preserving satisfiability	preserving validity
$\Box p$	$\exists x.\forall y.R(x,y) \rightarrow P(y)$	$\forall x.\forall y.R(x,y) \rightarrow P(y)$
$\Diamond p$	$\exists x.\exists y.R(x,y) \wedge P(y)$	$\forall x.\exists y.R(x,y) \wedge P(y)$

For our algorithm we shall chose the second kind of translation, preserving validity of formulas. We shall use the translation suggested above. On the other hand, translation function will be an input to our algorithm. This makes our method independent of any particular translation.

The translation presented above is so called *relational translation*. Since it is well known, we shall not define it here precisely, giving some examples instead (for precise definition see e.g. [7]).

Modal formula	Translated formula
$\Box p \rightarrow p$	$\forall x.(\forall y.R(x,y) \rightarrow P(y)) \rightarrow P(x)$
$\Box p \rightarrow \Box\Box p$	$\forall x.(\forall y.R(x,y) \rightarrow P(y)) \rightarrow \forall y.(R(x,y) \rightarrow \forall z.(R(y,z) \rightarrow P(z)))$
$p \rightarrow \Box\Diamond p$	$\forall x.P(x) \rightarrow (\forall y.R(x,y) \rightarrow \exists z.(R(y,z) \wedge P(z))$

In what follows we shall always consider translations satisfying the following requirement:

(3.1) modal formula is valid (in sense of the least normal modal logic) iff translated formula is valid (in sense of classical logic).

Proposition 2.3 justifies the requirement, for no additional conditions on relation R are necessary here. In case of other kinds of semantics one, of course, has to change the requirement as to normality of the logic. In general, requirement (3.1) can be formulated as follows:

(3.1') modal formula is valid (in sense of some basic modal logic) iff translated formula is valid (in sense of classical logic),

where "basic modal logic" depends on chosen semantics.

Note that the above translation can easily be extended to the case of modal schemas, using second-order quantification as follows (cf. also [8]).

Modal schema	Translated schema
$\Box p$	$\forall P.\forall x.\forall y.R(x,y) \rightarrow P(y)$
$\Diamond p$	$\forall P.\forall x.\exists y.R(x,y) \wedge P(y)$

The following table illustrates the translation of modal schemas.

Modal schema	Translated schema
$\Box p \rightarrow p$	$\forall P.\forall x.(\forall y.R(x,y) \rightarrow P(y)) \rightarrow P(x)$
$\Box p \rightarrow \Box\Box p$	$\forall P.\forall x.(\forall y.R(x,y) \rightarrow P(y)) \rightarrow \forall y.(R(x,y) \rightarrow \forall z.(R(y,z) \rightarrow P(z)))$
$p \rightarrow \Box\Diamond p$	$\forall P.\forall x.P(x) \rightarrow (\forall y.R(x,y) \rightarrow \exists z.(R(y,z) \wedge P(z))$

However, the above formulas are useless from the point of view of automated theorem proving, as the set of second-order tautologies even with one second-order quantification is, in general, totally undecidable (not even arithmetical). In order to put our translation into good use, we should then try to eliminate second-order quantification and find suitable first-order conditions on R that could serve as basis in proving theorems.

Observe that the elimination technique we consider preserves logical equivalence of formulas. This means that the resulting formula is equivalent over all interpretations. Thus, whenever one considers semantics of modal logics which is based on some interpreted formalism (like neighbourhood semantics), one has to work in some theory that approximates the formalism. We shall not consider the question of interpreted formalisms in this paper. Note that the requirement (3.1) (as well as (3.1')) excludes the possibility of dealing with interpreted formalisms directly.

4 The algorithm

The algorithm we shall present is based on Ackermann's techniques developed in connection to elimination problem (cf. [1]). Below we shall follow a simplified presentation of elimination techniques given in [6], slightly modifying it whenever necessary. Note also that we present the algorithm in its simplest (and thus not strongest) form. Possible directions of strengthening the algorithm are discussed in section 5.

The elimination techniques we consider are based on proposition 4.1 and lemma 4.2, where by $A(\sigma \leftarrow \delta)$ we denote the formula obtained from A by replacing every occurrence of expression σ by expression δ. Proposition 4.1 summarizes well-known facts (cf. e.g. [1]).

Proposition 4.1 The following pairs of formulas are equivalent, where Q stands for any quantifier and A, B, C are formulas such that C does not contain free occurrence of variable x

(1)	$\neg\neg A$	and	A
(2)	$\neg(A \wedge B)$	and	$\neg A \vee \neg B$
(3)	$\neg(A \vee B)$	and	$\neg A \wedge \neg B$
(4)	$\neg\forall x.A(x)$	and	$\exists x.\neg A(x)$
(5)	$\neg\exists x.A(x)$	and	$\forall x.\neg A(x)$
(6)	$\exists x.(A(x) \vee B(x))$	and	$\exists x.A(x) \vee \exists x.B(x)$
(7)	$\forall x.(A(x) \wedge B(x))$	and	$\forall x.A(x) \wedge \forall x.B(x)$
(8)	$Qx.(A(x)) \wedge C$	and	$Qx.(A(x) \wedge C)$
(9)	$C \wedge Qx.A(x)$	and	$Qx.(C \wedge A(x)))$
(10)	$Qx.(A(x)) \vee C$	and	$Qx.(A(x) \vee C)$
(11)	$C \vee Qx.A(x)$	and	$Qx.(C \vee A(x))$
(12)	$Qx.Qy.A$	and	$Qy.Qx.A$
(13)	$\forall x.(x \neq t \vee C(t \leftarrow x))$	and	$C(t)$ (where x is not in t)
(14)	$\exists x.(x = t \wedge C(t \leftarrow x))$	and	$C(t)$ (where x is not in t) $\square$

The following lemma is essential for the purpose of our algorithm (cf. e.g. [1, 6]).

Lemma 4.2 Let P be a predicate and $A(x_1, ..., x_n)$, $B(P)$ be classical first-order formulas (without second-order quantification). Let P occurs in B only positively and let A contains no occurrences of P at all. Then:

1. formula $\exists P.\forall x_1, ..., x_n.[P(x_1, ..., x_n) \vee A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg P)$ is equivalent to

$$B(P \leftarrow A(x_1, ..., x_n)),$$

   where, in the second formula, arguments $x_1, ..., x_n$ of A are each time substituted by actual arguments of P (with renaming the bound variables whenever necessary)

2. formula $\exists P.\forall x_1, ..., x_n.[\neg P(x_1, ..., x_n) \vee A(x_1, ..., x_n)] \wedge B(P)$ is equivalent to

$$B(P \leftarrow A(x_1, ..., x_n)),$$

   where, in the second formula, arguments $x_1, ..., x_n$ of A are each time substituted by actual arguments of P (with renaming the bound variables whenever necessary).

Proof: As proofs of both equivalences are quite similar, let us prove the first of them only.

Let us first prove that

$$\exists P.\forall x_1, ..., x_n.[P(x_1, ..., x_n) \vee A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg P) \quad (1)$$

implies

$$B(P \leftarrow A(x_1, ..., x_n)) \quad (2)$$

By assumption, P occurs in B only positively. Thus B is monotone in argument P, i.e. whenever $\forall x_1, ..., x_n.C(x_1, ..., x_n) \rightarrow D(x_1, ..., x_n)$, also $B(P \leftarrow C(x_1, ..., x_n))$ implies $B(P \leftarrow D(x_1, ..., x_n))$. Assume (1) holds. Let than P' be a predicate that satisfies (1). Thus

$$\forall x_1, ..., x_n.[P'(x_1, ..., x_n) \vee A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg P')$$

which can equivalently be reformulated as follows:

$$\forall x_1, ..., x_n.[\neg P'(x_1, ..., x_n) \rightarrow A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg P').$$

Thus, by monotonicity of B,

$$[B(P \leftarrow \neg P') \rightarrow B(P \leftarrow A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg P').$$

Now, by application of modus ponens, we obtain that $B(P \leftarrow A(x_1, ..., x_n))$, which shows the first of required implications.

What remains to prove is that formula (2) implies formula (1). Assume then that $B(P \leftarrow A(x_1, ..., x_n))$ is true. We shall exhibit P for which the formula

$$\forall x_1, ..., x_n.[P(x_1, ..., x_n) \vee A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg P)$$

is true as well. Define P to be $\neg A(x_1, ..., x_n)$. Then the above formula takes the form $\forall x_1, ..., x_n.[\neg A(x_1, ..., x_n) \vee A(x_1, ..., x_n)] \wedge B(P \leftarrow \neg\neg A(x_1, ..., x_n))$, i.e.

$$\forall x_1, ..., x_n.B(P \leftarrow A(x_1, ..., x_n)).$$

Since the substituted P was out of scope of the quantifier $\forall x_1, ..., x_n$ and $x_1, ..., x_n$ are substituted in A by actual arguments of P, the last formula reduces to $B(P \leftarrow A(x_1, ..., x_n))$. □

The above lemma implies the strategy of our algorithm. Namely, we shall try to turn the translated modal schema into suitable form and then to eliminate second-order quantification. If there is more than one second-order quantifier (what obviously happens when schema contains more than one propositional variable), one has to eliminate one quantification after another - cf. also example 4.5. In case of relational translation we deal with, all quantified predicate variables are unary. We presented more general form of the lemma in order to indicate possible extensions of the method.

Given an algorithm solving the problem we deal with, one would like to have an answer (at least) to the following questions:

- does the algorithm always stop
- is the algorithm *sound* in sense that whenever it does not fail, the returned formula is indeed the correspondence axiom for a given modal schema (w.r.t. given translation)
- is the algorithm *complete* in sense that whenever it fails, the correspondence axiom does not exist.

Let us emphasize here that our algorithm always stops and is sound. On the other hand, our algorithm is incomplete even when one restricts considerations to Kripke-like semantics and translation. There are, however, good reasons to believe that the problem we deal with is not partially decidable (at least in whole generality, when one attempts to give an algorithm that is independent of any particular translation of modal schemas), for first-order definability of Π^1_1 sentences[7] is not an arithmetical notion (cf. e.g. [8]).

We shall assume that three additional procedures are given:

- $CNF(A, B)$, which transforms classical first-order formula B into formula B in conjunctive normal form with all quantifiers in its prefix (that is, also into prenex normal form)
- $Skolemize(A, B)$, which eliminates existential quantifiers by applying the Skolem technique, whenever necessary
- $Unskolemize(A, B)$ which transforms formula A into formula B by eliminating Skolem functions introduced by $Skolemize$, if possible.

Note that the above procedures are standard ones and can be found in most textbooks on logic or logic programming. We, of course, require that the first procedure gives as a result an equivalent formula (in sense of logical equivalence). However, unlike in most sources, instead of satisfiability/unsatisfiability preservation, we require that both later procedures preserve equivalence of formulas. Namely, those procedures are based on the following (second-order) equivalence:

- $\forall x_1 ... x_n . \exists y . A(x_1, ..., x_n, y, ...)$ is equivalent to the following second-order formula[8]
$$\exists f . \forall x_1 ... x_n . A(x_1, ..., x_n, y \leftarrow f(x_1, ..., x_n), ...).$$

The above equivalence is used in $Skolemize$ to introduce existential second-order quantification (but in the beginning of considered formula) instead of existential first-order quantification. The $Unskolemize$ procedure uses the equivalence in order to eliminate second-order quantification and replace it by a first-order one.

Algorithm 4.3 (search for correspondence axiom)

Input:
1. function T translating modal formulas into classical first-order formulas
2. modal schema S (given syntactically as modal formula)

Output: correspondence axiom for S or answer that algorithm failed[9]

(1) let $P_1, ..., P_k$ be all propositional variables appearing in S and let A be second-order formula $\forall P_1, ..., P_k . T(S)$ — note that now A is the second-order formula that corresponds to schema S

(2) let B be $\neg A$ — now B takes the form $\exists P_{i_1}, ..., P_{i_r} . B'$

while there are quantified P's in B **do**

[7] i.e. sentences that begin with second-order quantifier of the form $\forall P$ followed by a first-order formula

[8] we assume the axiom of choice

[9] which does not mean that correspondence axiom does not exist

(3) let P be one of $P_{i_1}, ..., P_{i_r}$, say P_{i_j}, now set B to be formula $\exists P.B'$ — we try to eliminate quantification over P; all other P's are now treated as predicate constants

(4) call $CNF(B', C)$ and set B to be $\exists P.C$

(5) if C begins with existential quantifiers, move those before $\exists P$ (using proposition 4.1.12)

(6) if any conjunct of C containing P contains (among others) positive occurrence of P or any of those conjuncts contains (among others) negative occurrence of P then set F to be formula obtained from C by deleting all conjuncts containing P (the empty conjunction consider, as usual, as $\top$) and execute step (11) — for justification of this step see proposition 4.4

(7) if there is a conjunct of C that contains both positive and negative occurrence of P, report that algorithm failed and stop[10]; otherwise transform equivalently C into form $\underline{C} \wedge \overline{C}$, where positive and negative occurrences of P are separated (i.e. $\underline{C}$ contains no negative occurrences of P and $\overline{C}$ contains no positive occurrences of P)

(8) if any conjunct of $\underline{C}$ contains at most one occurrence of P then transform equivalently[11] $\underline{C}$ into form $\exists f_1...f_l.\forall z_1...z_n.(P(z_1...z_n) \vee D_1) \wedge D_2$, where f_i's are Skolem functions and D_1, D_2 contain no occurrences of P; otherwise, if any conjunct of $\overline{C}$ contains at most one occurrence of P then transform[12] $\overline{C}$ into form $\exists f_1...f_l.\forall z_1...z_n.(\neg P(z_1...z_n) \vee D_1) \wedge D_2$, where F_i's, D_1 and D_2 are as before; if both $\underline{C}$ and $\overline{C}$ contain a conjunct with two or more different occurrences of P then report that algorithm failed and stop[10]

(9) now we are able to apply suitable part of lemma 4.2, for B receives now the form

$$\exists f_1...f_l.\exists P.[\forall z_1...z_n.(P(z_1...z_n) \vee D_1)] \wedge [D_2 \wedge \overline{C}]$$

(or $\exists f_1...f_l.\exists P.[\forall z_1...z_n.(\neg P(z_1...z_n) \vee D_1)] \wedge [D_2 \wedge \underline{C}]$, respectively); let E be formula that results from application of lemma 4.2 to B

(10) if Skolemization was necessary while executing step (8) then try to eliminate Skolem functions by calling $Unskolemize(E, F)$; if unskolemization is not possible then report that algorithm failed and stop

(11) let now B be $\exists P_{i_1}...P_{i_r}.F$; remove all irrelevant second-order quantifiers over P_i's that do not appear in F (in particular remove quantifier $\exists P$, i.e. $\exists P_{i_j}$) — now $\neg B$ is equivalent to modal schema S, but contains no occurrence of P_{i_j} (and maybe of some other second-order quantifiers)

od

(12) return $\neg B$ as the result — now $\neg B$ is the correspondence axiom $\square$

Observe that the above algorithm always stops and is sound in the sense that whenever it does not fail, the resulting formula is indeed the correspondence axiom. That is obvious, as all above transformations preserve equivalence of formulas, and lemma 4.2 indeed eliminates quantification over P. What has no counterpart in the elimination technique

[10] note that this step can be strengthened - cf. next section

[11] more precise description of this step is given in algorithm 4.6

[12] note that it is sometimes more convenient to transform $\overline{C}$ first if its transformation is simpler - such an optimization is sometimes used in examples given below.

of lemma 4.2 and maybe still requires some explanation is the transformation applied in step (6). The below proposition justifies the transformation.

Proposition 4.4 Let A be a classical first-order formula of the form $Qx_1....Qx_n.(A_1 \wedge ... \wedge A_q)$, where A_1, ..., A_q are disjunctions of atomic formulas. Assume that any conjunct of A containing P contains (among others) positive occurrence of P or any of those conjunct contains (among others) negative occurrence of P. Let B be formula $\exists P.A$

Then B is equivalent to

$$Q_1x_1....Q_nx_n.(A_{i_1} \wedge ... \wedge A_{i_r}),$$

where $i_1, ..., i_r \in \{1, ..., q\}$ and A_{i_1}, ..., A_{i_r} are all conjuncts that do not contain occurrences of P (the empty conjunction is, by convention, $\top$).

Proof:

($\rightarrow$) By assumptions formula A implies formula which results from A by replacing P (respectively $\neg P$) by $\top$. But, after replacing, all conjuncts that contained P (respectively $\neg P$) are disjunctions that contain $\top$, i.e. are equivalent to $\top$. Thus A implies $Q_1x_1....Q_nx_n.(A_{i_1} \wedge ... \wedge A_{i_r})$, and so $\exists P.A$ implies $\exists P.Q_1x_1....Q_nx_n.(A_{i_1} \wedge ... \wedge A_{i_r})$. The second formula contains no occurrence of P, so the quantifier $\exists P$ can be removed from this formula. This proves implication ($\rightarrow$).

($\leftarrow$) Assume $Q_1x_1....Q_nx_n.(A_{i_1} \wedge ... \wedge A_{i_r})$ is true under some interpretation and valuation of free variables. We have to show that formula $\exists P.A$ is true as well. Thus we have to exhibit P for which A is true (under the same interpretation and valuation). Set P to be $\top$ (or, in case of negative occurrences of P, to be $\bot$). Now all conjuncts containing P reduce to $\top$, thus A reduces to $Q_1x_1....Q_nx_n.(A_{i_1} \wedge ... \wedge A_{i_r})$, which is true, by initial assumption. $\Box$

Let us now illustrate the method with a few examples. Note that below we use some optimizations, e.g. whenever positive and negative occurrences of P are separated, we do do not transform the whole formula into conjunctive normal form but one of its parts only, etc.

Example 4.5 Consider first modal schemas $\Box p \rightarrow p$, $\Box p \rightarrow \Box\Box p$ and $p \rightarrow \Box\Diamond p$. As we treat implication as abbreviation, those should be equivalently rewritten as $\neg\Box p \vee p$, $\neg\Box p \vee \Box\Box p$ and $\neg p \vee \Box\Diamond p$

Let us now apply the algorithm to those schemas. Let us start with the first one.

— translated schema (cf. section 3): $\forall P.\forall x.\neg(\forall y.\neg R(x,y) \vee P(y)) \vee P(x)$
— negated: $\exists x.\exists P.(\forall y.\neg R(x,y) \vee P(y)) \wedge \neg P(x)$ — note that the formula is already in separated form
— transformed: $\exists x.\exists P.\forall z.[\neg P(z) \vee z \neq x] \wedge [\forall y.\neg R(x,y) \vee P(y)]$ — now lemma 4.2(2) can be applied
— P eliminated: $\exists x.\forall y.\neg R(x,y) \vee y \neq x$
— negated: $\forall x.\exists y.R(x,y) \wedge y = x$ — now algorithm stops, however we can still simplify the above formula, using proposition 4.1(14) according to which $\exists y.R(x,y) \wedge y = x$ is equivalent to $R(x,x)$
— simplified: $\forall x.R(x,x)$ — i.e. reflexivity of R

Let us now consider the second schema.

— translated schema (cf. section 3):
$\forall P.\forall x.\neg(\forall y.\neg R(x,y) \vee P(y)) \vee \forall y.(\neg R(x,y) \vee \forall z.(\neg R(y,z) \vee P(z)))$
— negated: $\exists x.\exists P.(\forall y.\neg R(x,y) \vee P(y)) \wedge \exists y.(R(x,y) \wedge \exists z.(R(y,z) \wedge \neg P(z)))$
— note that the formula is already in separated form
— transformed: $\exists x.\exists P.\forall y.[P(y) \vee \neg R(x,y)] \wedge [\exists y.R(x,y) \wedge \exists z.(R(y,z) \wedge \neg P(z))]$
— now lemma 4.2(1) can be applied
— P eliminated: $\exists x.\exists y.R(x,y) \wedge \exists z.(R(y,z) \wedge \neg R(x,z))$
— negated: $\forall x.\forall y.\neg R(x,y) \vee \forall z.(\neg R(y,z) \vee R(x,z))$ — algorithm stops here, however the formula can still be simplified as follows
— simplified: $\forall x,y,z.R(x,y) \rightarrow (R(y,z) \rightarrow R(x,z))$ — i.e. transitivity of R

The elimination of P from the third schema proceeds as follows.

— translated schema (cf. section 3):
$\forall P.\forall x.\neg P(x) \vee (\forall y.\neg R(x,y) \vee \exists z.(R(y,z) \wedge P(z))$
— negated: $\exists x.\exists P.P(x) \wedge \exists y.(R(x,y) \wedge \forall z.(\neg R(y,z) \vee \neg P(z))$ — note that the formula is already in separated form
— transformed: $\exists x.\exists P.\forall z.[P(z) \vee x \neq z] \wedge [\exists y.R(x,y) \wedge \forall z.(\neg R(y,z) \vee \neg P(z))]$
— now lemma 4.2(1) can be applied
— P eliminated: $\exists x.\exists y.R(x,y) \wedge \forall z.(\neg R(y,z) \vee x \neq z)$
— negated: $\forall x.\forall y.\neg R(x,y) \vee \exists z.(R(y,z) \wedge x = z)$ — algorithm stops here, but we can simplify the formula using proposition 4.1(14) and definition of $\rightarrow$
— simplified: $\forall x,y.R(x,y) \rightarrow R(y,x)$ — i.e. symmetry of R

Let us now illustrate the algorithm in case of elimination of two second-order quantifications. For this purpose consider the schema $\Box(p \vee q) \rightarrow (\Box p \vee \Box q)$, i.e. $\neg\Box(p \vee q) \vee (\Box p \vee \Box q)$.

— translated schema: $\forall P.\forall Q.\forall x.\neg(\forall y.(R(x,y) \rightarrow (P(y) \vee Q(y))) \vee$
$(\forall z.(R(x,z) \rightarrow P(z)) \vee \forall v.(R(x,v) \rightarrow Q(v)))$
— negated: $\exists x.\exists P.\exists Q.(\forall y.(\neg R(x,y) \vee P(y) \vee Q(y)) \wedge (\exists z.(R(x,z) \wedge \neg P(z)) \wedge$
$\exists v.(R(x,v) \wedge \neg Q(v))$
— separated (w.r.t. Q): $\exists x,z,v.\exists P.\exists Q.\forall y.[Q(y) \vee (\neg R(x,y) \vee P(y))] \wedge [R(x,z) \wedge$
$\neg P(z) \wedge R(x,v) \wedge \neg Q(v)]$
— Q eliminated: $\exists x,z,v.\exists P.(R(x,z) \wedge \neg P(z) \wedge R(x,v) \wedge (\neg R(x,v) \vee P(v))$
— separated (w.r.t. P):
$\exists x,z,v.\exists P.[P(v) \vee \neg R(x,v)] \wedge R(x,z) \wedge \neg P(z) \wedge R(x,v) \wedge \neg P(v)$ — now use proposition 4.1(13)
— transformed:
$\exists x,z,v.\exists P.\forall u.[P(u) \vee (u \neq v \vee \neg R(x,u))] \wedge [R(x,z) \wedge \neg P(z) \wedge R(x,v)]$ — now lemma 4.2 can be applied
— P eliminated: $\exists x,z,v.[R(x,z) \wedge (z \neq v \vee \neg R(x,z)) \wedge R(x,v)]$
— negated: $\forall x,z,v.[\neg R(x,z) \vee (z = v \wedge R(x,z)) \vee \neg R(x,v)]$ — algorithm stops here, but we can still make some simplifications:
— simplified: $\forall x,z,v.((R(x,z) \wedge R(x,v)) \rightarrow (z = v \wedge R(x,z))$, i.e.
$\forall x,z,v.((R(x,z) \wedge R(x,v)) \rightarrow z = v$ $\Box$

The following more precise description of the eighth step of algorithm 4.3 completes the presentation of our algorithm.

Algorithm 4.6 (step (8))

Input: first-order formula C in conjunctive normal form, containing positive occurrences of P only[13]

Output: formula of the form $\exists f_1...f_l.\forall z_1...z_n.(P(z_1,...,z_n) \vee D_1) \wedge D_2$, where f_i's are Skolem functions and D_1, D_2 contain no occurrences of P. The resulting formula is equivalent to C

(1) if C contains existential quantifiers then call $Skolemize(C, E)$ — now E is of the form $\exists f_1...f_l.\forall x_1....x_m.(E_1 \wedge ... \wedge E_k)$, where f_i's are Skolem functions and E_i's are disjunctions of atomic formulas

(2) set D_2 to be formula $\forall x_1....x_m.(E_{i_1} \wedge ... \wedge E_{i_q})$, where E_{i_j}'s are all conjuncts of E that do not contain P; delete all those E_{i_j}'s from E

(3) move universal quantifiers in E before each conjunct (using proposition 4.1(7))— now E is of the form

$$\exists f_1...f_l.(\forall x_1....x_m.E_1 \wedge ... \wedge \forall x_1....x_m.E_r),$$

where E_i's are disjunctions of atomic formulas and each E_i contains P

for all conjuncts of E **do**

(4) using proposition 4.1(13) transform the conjunct into form

$$\forall x_1....x_m.\forall z_1....z_n.(P(z_1,....,z_n) \vee F),$$

where z_i's are fresh variables

od

(5) using proposition 4.1(11)-(12) and law $(p \vee q) \wedge (p \vee s) \leftrightarrow (p \vee (q \wedge s))$ transform equivalently E into form

$$\exists f_1...f_l.\forall z_1....\forall z_n.[P(z_1,...,z_n) \vee \forall x_1....x_m.(F_1 \wedge ... \wedge F_r)]$$

(6) return as the result formula

$$\exists f_1...f_l.\forall z_1....\forall z_n.[P(z_1,...,z_n) \vee \forall x_1....x_m.(F_1 \wedge ... \wedge F_r)] \wedge D_2. \qquad \square$$

Let us now give two examples where algorithm 4.3 fails (note, however, that both examples consider modal schemas that are not first-order definable).

Example 4.7 Consider the Löb axiom $\Box(\Box p \to p) \to \Box p$, i.e. $\neg\Box(\neg\Box p \vee p) \vee \Box p$.

— translated: $\forall P.\forall x.(\neg(\forall y.(R(x,y) \to (\neg\forall z.(R(y,z) \to P(z)) \vee P(y))) \vee \forall u.(R(x,u) \to P(u)))$

— negated:
$\exists x.\exists P.((\forall y.(\neg R(x,y) \vee (\exists z.(R(y,z) \wedge \neg P(z)) \vee P(y))) \wedge \exists u.(R(x,u) \wedge \neg P(u)))$

— try to separate:
$\exists z,u.\exists P.\exists f.\forall y.(\neg R(x,y) \vee R(y,f(y)) \vee P(y)) \wedge (\neg R(x,y) \vee \neg P(f(y)) \vee P(y))) \wedge (R(x,u) \wedge \neg P(u)))$ — new Skolem function f was introduced;

— report failure — the second conjunct cannot be separated

Consider now the McKinsey's axiom $\Box\Diamond p \to \Diamond\Box p$, i.e. $\neg\Box\Diamond p \vee \Diamond\Box p$.

[13] the case of negative occurrences of P is symmetric and thus need not be presented here

— translated: $\forall P.\forall x.(\neg\forall y.(R(x,y) \rightarrow \exists z.(R(y,z) \wedge P(z)) \vee \exists u.(R(x,u) \wedge \forall v.(R(u,v) \rightarrow P(v)))$
— negated: $\exists x.\exists P.(\forall y.(\neg R(x,y) \vee \exists z.(R(y,z) \wedge P(z)) \wedge \forall u.(\neg R(x,u) \vee \exists v.(R(u,v) \wedge \neg P(v)))$
— separated: $\exists x.\exists P.\exists f.\forall y.[\neg R(x,y) \vee P(f(y)] \wedge [\forall y.(\neg R(x,y) \vee R(y,f(y))) \wedge \forall u.(\neg R(x,u) \vee \exists v.(R(u,v) \wedge \neg P(v))]$
— transformed: $\exists x.\exists f.\exists P.\forall z.[P(z) \vee \forall y.(\neg R(x,y) \vee z \neq f(y))] \wedge [\forall y.(\neg R(x,y) \vee R(y,f(y))) \wedge \forall u.(\neg R(x,u) \vee \exists v.(R(u,v) \wedge \neg P(v))]$ — now lemma 4.2(1) can be applied
— P eliminated: $\exists x.\exists f.\forall y.(\neg R(x,y) \vee R(y,f(y))) \wedge \forall u.(\neg R(x,u) \vee \exists v.(R(u,v) \wedge \forall y.(v \neq f(y) \vee \neg R(x,y))))$
— report failure — the last formula cannot be unskolemized (which may even easier be seen after transformation into conjunctive normal form). □

Note that in case of Löb's axiom SCAN algorithm of [4] loops (while our stops).

Observe also that our algorithm is incomplete, for it fails in producing correspondence axiom for $(\Box p \rightarrow \Box\Box p) \wedge (\Box\Diamond p \rightarrow \Diamond\Box p)$[14]. The fact that McKinsey's axiom has a first-order equivalent in transitive frames can be found e.g. in [8]. On the other hand, our algorithm applied to $\forall p.(\Box p \rightarrow \Box\Box p) \wedge (\Box\Diamond p \rightarrow \Diamond\Box p)$, i.e. equivalently to $\forall p.(\Box p \rightarrow \Box\Box p)$ and, separately, to $(\Box\Diamond p \rightarrow \Diamond\Box p)$ produces formula that cannot be unskolemized. Observe that the same argument shows incompleteness of the SCAN algorithm.

5 Strengthening the algorithm

Observe that steps (7) and (8) of the algorithm can still be modified so that the algorithm accepts essentially more formulas. The whole idea, however, remains the same.

Let us first start with a simple modification of step (7). Namely, in some cases, when scopes of quantifiers allow for this, one can separate positive and negative occurrence of P in a conjunct. Assume E is in conjunctive normal form, with no existential first-order quantifiers placed after $\exists P$, i.e. E is of the form $\exists f_1....f_n.\exists P.\forall x_1...x_p.(E_1 \wedge ... \wedge E_m)$. Assume a conjunct, say E_i, is of the form $\underline{E} \vee \overline{E}$, where $\underline{E}$ contains no negative occurrences of P and $\overline{E}$ contains no positive occurrences of P. One can then transform E into form

$$\exists f_1....f_n.\exists P.\forall x_1...x_p.(E_1 \wedge ... \wedge E_{i-1} \wedge E_{i+1} \wedge ... \wedge E_r \wedge \underline{E}) \vee (E_1 \wedge ... \wedge E_{i-1} \wedge E_{i+1} \wedge ... \wedge E_r \wedge \overline{E}),$$

using the law $p \wedge (q \vee s) \leftrightarrow (p \wedge q) \vee (p \wedge s)$. Now, if the scopes of quantifiers do not make this impossible, one can separate the above formula into two formulas $\exists f_1....f_n.\exists P.\forall x_1...x_p.(E_1 \wedge ... \wedge E_{i-1} \wedge E_{i+1} \wedge ... \wedge E_r \wedge \underline{E})$ and $\exists f_1....f_n.\exists P.\forall x_1...x_p.(E_1 \wedge ... \wedge E_{i-1} \wedge E_{i+1} \wedge ... \wedge E_r \wedge \overline{E})$. After such a separation one can consider those formulas separately and, at the end, collect them into one disjunction.

The below example clarifies the above improvement of the algorithm.

Example 5.1 The formula $\exists x.\exists P.\forall y.(\neg P(x) \vee P(y) \vee G(y)) \wedge H(x)$ can be transformed as follows: $\exists x.\exists P.\forall y.(\neg P(x) \wedge H(x)) \vee (P(y) \vee G(y)) \wedge H(x))$. Now, by using proposition

[14] the counterexample was supplied to us by H.J. Ohlbach

4.1(11), we obtain $\exists x.\exists P.(\neg P(x) \wedge H(x)) \vee \forall y.((P(y) \vee G(y)) \wedge H(x))$, i.e. (by proposition 4.1(6)) $\exists x.\exists P.(\neg P(x) \wedge H(x)) \vee \exists x.\exists P.\forall y.((P(y) \vee G(y)) \wedge H(x))$. This gives us the required separation of positive and negative occurrences of P. Now the elimination of P can proceed separately for the above two disjuncts.

Note however, that the following formula:

$$\exists P.\forall x, y.(\neg P(x) \vee P(y) \vee G(x, y)) \wedge H(x)$$

cannot be separated by application of this technique. □

Similarly, different positive (respectively negative) occurrences of P in one conjunct can sometimes be separated by similar application of the technique described above. This allows us to strengthen the eighth step of algorithm 4.3. If the above technique fails, we can still apply, in case of step (8), yet another method based on *factorization principle* we introduce below. Note that the principle we present essentially strengthens the elimination technique of Ackermann. It is based on the following lemma.

Lemma 5.2 Let P be an n-argument predicate and let C be a formula that does not contain occurrences of P. Then formula

$$\forall x_1 ... x_m.(\pm P(t_{11}, ..., t_{1n}) \vee \pm P(t_{21}, ..., t_{2n}) \vee ... \vee \pm P(t_{k1}, ..., t_{kn}) \vee C)$$

is equivalent to formula

$$\exists Q.[\forall y_1 ... y_n.\neg Q(y_1, ..., y_n) \vee \pm P(y_1, ..., y_n)] \wedge$$
$$\forall x_1 ... x_m.[\pm P(t_{11}, ..., t_{1n}) \vee Q(t_{21}, ..., t_{2n}) \vee ... \vee Q(t_{k1}, ..., t_{kn}) \vee C],$$

where Q and y_i's are fresh variables and $\pm$ stands either for negation (in case when all occurrences of P are negative) or for no symbol (in case when all occurrences of P are positive).

Proof: The proof can be carried out by direct application of lemma 4.2. □

Note that in the above lemma occurrences of P are separated and lemma 4.2 can be applied to eliminate P. After such an elimination we still have to eliminate Q. We do this by application of our (strengthened) algorithm.

Observe that we proved lemma 5.2 by application of lemma 4.2. However, unlike in elimination technique we deal with, we introduced new second-order quantifications. Thus we strengthen the technique by proposing a new way of applying the basic equivalences of lemma 4.2. Observe that we have to modify the strategy of our algorithm in order to avoid loops. The strategy can now be the following:

> try to eliminate P and, in case of application of lemma 5.2, try to eliminate new quantifications. If complexity of new quantifications is greater than that of P, report failure and stop, otherwise continue the execution of algorithm.

The complexity of quantifications, say over S_1,..., S_t, is measured here by number of occurrences of S_i's in all conjuncts.

Below we show an example of application of factorization principle.

Example 5.3 Consider formula $\exists P.\forall x, y.(P(x) \vee P(y) \vee R(x,y)) \wedge (\neg P(x) \vee R(y,x))$:

— quantification over Q introduced:
$\exists Q.\exists P.\forall x, y.(P(x) \vee Q(y) \vee R(x,y)) \wedge (\neg P(x) \vee R(y,x)) \wedge \forall z.(\neg Q(z) \vee P(z))$
— transformed: $\exists Q.\exists P.[\forall z.(P(z) \vee (\forall u, v.(z \neq u \vee Q(v) \vee R(u,v)) \wedge \neg Q(z))] \wedge$
$[\forall x, y.(\neg P(x) \vee R(y,x))]$
— P eliminated: $\exists Q.\forall x, y.(\forall u, v.(x \neq u \vee Q(v) \vee R(u,v)) \wedge \neg Q(x)) \vee R(y,x))$
— transformed into conjunctive normal form:
$\exists Q.\forall x, y, u, v.(x \neq u \vee Q(v) \vee R(u,v) \vee R(y,x)) \wedge (\neg Q(x) \vee R(y,x))$
— transformed: $\exists Q.[\forall v.(Q(v) \vee \forall x, y, u.(x \neq u \vee R(u,v) \vee R(y,x))] \wedge$
$[\forall w, z.(\neg Q(w) \vee R(z,w))]$
— Q eliminated: $\forall w, z.(\forall x, y, u.(x \neq u \vee R(u,w) \vee R(y,x)) \vee R(z,w))$. □

6 Final remarks

We presented an algorithm for automated search of correspondence axioms. We mainly illustrated the use of algorithm on Kripke-like semantics of modal logics. Observe, however, that the technique we apply is based on elimination of second-order quantification from formulas. That makes it independent of any particular semantics. In fact, the semantics is given to the algorithm only via translation function which is a parameter for the algorithm. Thus restriction as to normality of considered logics, as inherited together with Kripke semantics, is again inessential in sense, that whenever dealing with non-normal logics, one has to apply another kind of semantics and translation anyway.

We did not consider the case of multimodal logics. This again was done without loss of generality, as what we were really working with was always some translated modal schema, i.e. Π^1_1 formula. The number of modalities is then inessential from the point of view of the algorithm. Similarly, the algorithm can deal with many-argument modalities etc. More generally - the algorithm is applicable to Π^1_1 formulas and thus to any semantics or translation which associates a Π^1_1 formula to a modal schema.

Observe that we considered only translations that preserve validity of formulas. In particular we did not deal with translations that are based on any interpreted formalism.

Recall that we proved soundness of the algorithm. Unfortunately, even the strengthened algorithm is incomplete. We believe that the problem, in most general form, is totally undecidable. It seems, however, that for particular semantics and translations complete algorithms can be given. This, however, is still an open problem.

Note that the factorization principle introduced in section 5 essentially strengthens the elimination technique of Ackermann. Thus, as a side-effect of this paper, we get a method for proving a subset of second order logic which is essentially stronger than that considered in [1, 6].

Bibliography

[1] W. Ackermann: *Solvable Cases of the Decision Problem*, North-Holland Pub. Co. (1954).

[2] R. A. Bull, K. Segerberg: *Basic Modal Logic*, in: Handbook of Philosophical Logic, Vol. 2 (D. Gabbay and F. Guenthner, eds.), D. Reidel Pub. Co. (1984), 1-88.

[3] M. Fitting: *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel Pub. Co. (1983).

[4] D. Gabbay and H. J. Ohlbach: *Automatic Synthesis of Correspondence Axioms*, unpublished manuscript (1991).

[5] G. E. Hughes, M.J. Cresswell: *An Introduction to Modal Logic*, Routledge (1989).

[6] J. T. Minor: *Proving a Subset of Second-Order Logic with First-Order Proof Procedures*, Ph.D. Thesis., The University of Texas at Austin (1979).

[7] H. J. Ohlbach: *Semantics-Based Translation Methods for Modal Logics*, J. Logic Computat., 1, 5 (1991), 691-746.

[8] J. Van Benthem: *Correspondence Theory*, in: Handbook of Philosophical Logic, Vol. 2 (D. Gabbay and F. Guenthner, eds.), D. Reidel Pub. Co. (1984), 167-247.