Towards Automating Duality

Chris Brink
Dept. of Mathematics
University of Cape Town
7700 Rondebosch
South Africa
email: cbrink@maths.uct.ac.za

Dov M. Gabbay*
Imperial College of Science, Technology and Medicin
Dept. of Computing
Huxley Building, 180 Queen's Gate
London SW7 2AZ,
England
email: dg@doc.ic.ac.uk

Hans Jürgen Ohlbach†
Max–Planck–Institut für Informatik
Im Stadtwald
D-6600 Saarbrücken 11
F. R. Germany
email: ohlbach@mpi-sb.mpg.de

April 29, 1994

Abstract

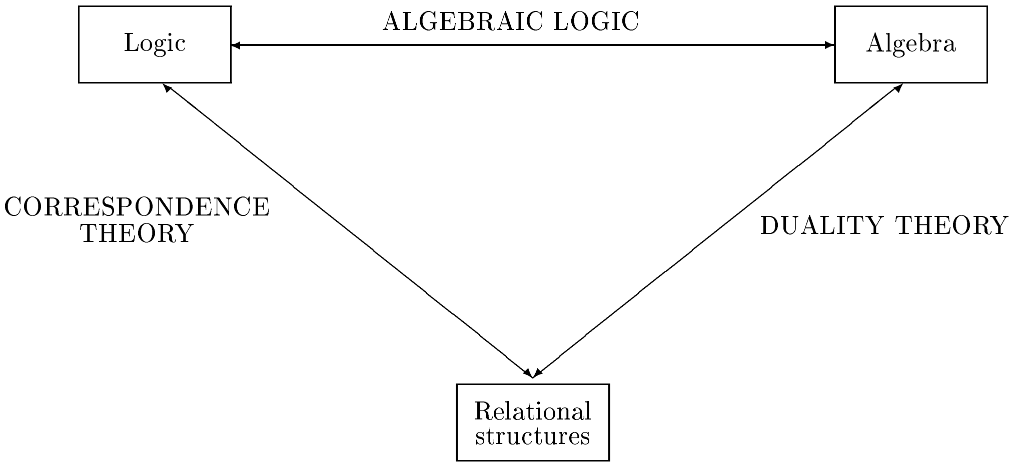
Dualities between different theories occur frequently in mathematics and logic — between syntax and semantics of a logic, between structures and power structures, between relations and relational algebras, to name just a few. In this paper we show for the case of structures and power structures how corresponding properties of the two related structures can be computed fully automatically by means of quantifier elimination algorithms and predicate logic theorem provers. We illustrate the method with some examples which were computed with the Otter theorem prover. All technical details and many further examples can be found in [BGO93].

*The second author is a SERC Senior Research Fellow.
†This work was supported by the ESPRIT project 6471 MEDLAR and by the BMFT funded project LOGO (ITS 9102). Parts of the paper were written while the third author was visiting the University of Cape Town.

Contents

1 Introduction

To any logic satisfying certain minimal requirements there corresponds both an algebra and a relational semantics, and the general picture of such relationships can be presented as in the figure below.



As a paradigm case, consider the modal logic S4. Its algebraic counterpart is the variety of closure algebras, related to the logic through the Lindenbaum/Tarski construction and an algebraic completeness result. The Kripke semantics of S4, on the other hand, is given by the theory of quasi-orders (reflexive transitive relations). Finally, quasi-orders and closure algebras are related to each other through a power construction first used in a (now famous but long neglected) paper of Jónsson and Tarski [JT51][JT52].

Algebraic Logic, broadly speaking, stands in the tradition of the algebraization of classical propositional logic as the variety of Boolean algebras. This has been generalised to many extensions of classical propositional logic (notably modal logics), yielding various *Boolean Algebras with Operators* ([Jón92]), and also to variations on classical propositional logics (e.g. intuitionistic logic, relevance logic, many-valued logics), typically yielding distributive lattices with operators. These propositional cases are fairly well understood (Blok and Pigozzi [BP89]), but the problem of algebraization of first- and higher-order logics has proved much more difficult. The *cylindric algebras* of Henkin, Monk and Tarski [HMT71] [HMT85], the *monadic algebras* of Halmos [Hal62] and the work of Craig [Cra74] are all attempts to present algebraic versions of first-order logic. For a comprehensive overview see Nemeti [Nem92].

Duality Theory, from the perspective of non-classical logic (Bull and Segerberg [BS84]), studies the relationship between the semantics of a logic and its algebra, and how to obtain each from the other. Typically, the semantics is defined with reference to some relational structure (sometimes called a *frame*, or *model structure*); the corresponding algebra is obtained by a power construction, and from the algebra the model structure can be recovered by imposing a relational structure on the set of ultrafilters. More generally, duality theory studies the relationship between algebras and relational structures without necessarily referring to any logic. The variety of *relation algebras* is a case in point. These algebras arose in response to the problem posed by Tarski [Tar41] of finding equational axioms that would capture the calculus of binary relations, in the same way as the axioms for Boolean algebras capture the calculus of sets. Already in [JT51] [JT52] it was established that relation algebras stand in a duality relationship to structures called *generalized Brandt groupoids*, whereas the relationship of algebras of relations to first-order logic was only fully presented by Tarski and Givant in [TG87]. A different but related perspective on duality is that of the topologist. Stone [Sto37] related Boolean algebras to topological spaces, and the study of certain lattices and their topological duals became a topic of study in its own right (Johnstone [Joh82]). Through the work of Priestley [Pri70] and Hansoul [Han83] this extends also to lattices *with operators*, which are dual to topological spaces endowed with relational structure. By this route the semantics of some logics are in full topological duality to their algebras.

Correspondence Theory (or *definability theory*) considers the classical definability of nonclassical formulae (specifically propositional *modal* formulae) when viewed as relational principles (Van Benthem [vB84]). The notion of expressing the 'meanings' of modalities in terms of a *possible-world semantics* goes back to Kripke [Kri59], who coded principles of logic as properties of an accessibility relation between possible worlds. More generally, the question arises which modal formulae define *first-order* relational conditions - and how do they do it? Conversely, which first-order relational conditions are modally definable?

The relevance of these well-established studies to Computer Science is gradually becoming clear. One obvious application is to so-called program logics (e.g. dynamic logic, Kozen [Koz81]), many of which are variations on modal or multi-modal logic. Such logics have been explicitly linked to Boolean algebras with operators (Pratt [Pra90]), and suggestions have been made on the use of relation algebras for program specification (Jónsson [Jón], Hoare and He Jifeng [HJ87]). *Denotational semantics* of programming languages seems another promising area, particularly in view of the presentation of *domain theory* in logical form (Abramsky [Abr87]), and the constructions required for *powerdomains*. Finally, formalisms such as *bilattices* (Ginsberg [Gin88]) and various deontic and epistemic logics proposed for AI research again fit the triangular pattern of logic-algebra-semantics. One recent example is that the independently-conceived system KL-ONE for knowledge representation (Brachman and Schmolze [BS85]) can be viewed as having a semantics of relations interacting with sets, and thus to have both a modal logic presentation (Schild [Sch91]) and an algebraic one (Brink and Schmidt [BS92]).

The fact that there are well-established areas of research relating logic, algebra and semantics shows that translation of individual formulae from a logical to an algebraic (equational) and to a structural (first-order) version is not a trivial matter. Many such translations are known; all of these have been found by traditional pencil-and-paper methods. Recently, however, Gabbay and Ohlbach [GO92] have proposed to inject a measure of *automated reasoning* into the area, by the use of an algorithm (called SCAN) for quantifier elimination in second-order logic. They give a number of examples from *correspondence theory*, showing how Hilbert axioms may by means of SCAN be translated into first-order properties of the accessibility relation.

The aim of this paper is to continue research in this area by considering the automation of *duality theory*, in both directions. More particularly, it is to marry the use of SCAN to the concept of *power structures*, presented in Brink [Bri93] as a useful cross-disciplinary unifying concept. Being 'more mathematical', we hope in this way to make the proposal of automation accessible also to mathematicians unfamiliar with the intricacies of non-classical logics. It turns out that the methods for finding correspondences between Hilbert axioms and properties of the relational structures (correspondence theory) and for finding correspopnding properties for structures and power structures (duality theory) are almost identical. Therefore this paper is as relevant to correspondence theory as it is to duality theory.

The mathematical theory of power structures is introduced in the next section. The general framework for automating duality is presented in section 3. The kernel of the algorithms, the quantifier elimination algorithm, is introduced in section 4. Eventually we present a few of examples. Many further examples can be found in [BGO93].

2 Power Structures and Dualities

The theory of Boolean algebras with operators was introduced in Jónsson and Tarski [JT51],[JT52]. The operators on the elements of the Boolean algebra are assumed to be additive in each argument. One way in which a Boolean algebra with operators arises is as the *power algebra* of a relational structure. If a relational structure is defined over some set A, then its power algebra is defined over the power set of A. This power algebra is a Boolean algebra of sets, with the usual set-theoretic operations $\cup$, $\cap$ and $'$, but with additional operators on the Boolean algebra which are the *power operations* of the relations defined over A.

Definition 2.1 *For any set A, and any (n+1)-ary relation $R \subseteq A^{n+1}$, the power operation $R^{\uparrow} : \mathcal{P}(A)^n \to \mathcal{P}(A)$ is defined by:*

$$R^{\uparrow}(X_0, \ldots, X_{n-1}) = \{x_n \mid (\exists x_0 \in X_0) \ldots (\exists x_{n-1} \in X_{n-1})[Rx_0 \ldots x_n]\},$$

for every $X_0, \ldots, X_{n-1} \subseteq A$.

For any relational structure $\mathcal{A} = (A; R_1, \ldots, R_m)$, the power algebra $\mathcal{P}(\mathcal{A})$ is defined by:

$$\mathcal{P}(\mathcal{A}) = (\mathcal{P}(A); R_1^{\uparrow}, \ldots, R_m^{\uparrow}).$$

For example, if R is a binary relation, then $R^{\uparrow} : \mathcal{P}(A)^n \to \mathcal{P}(A)$ is defined by $R^{\uparrow}(X) = \{y \mid (\exists x \in X)[Rxy]\}$. Adding the operations $\cup$, $\cap$ and $'$ to the power algebra $\mathcal{P}(\mathcal{A})$ yields a Boolean algebra with operators $R_1^{\uparrow}, \ldots, R_m^{\uparrow}$. Jónsson and Tarski further proved that all Boolean algebras with operators arise in this fashion, thus obtaining a representation theorem.

Theorem 2.2 *Any Boolean algebra with operators is isomorphic to a subalgebra of the power algebra of some relational structure.*

An operator on sets is called *normal* if whenever one of the arguments of the operator is the empty set, then so is the outcome. A Boolean algebra with operators is normal if all its operators are normal. For these algebras, the inverse of the power construction of definition 2.1 provides a mechanism to obtain the underlying relational structure of the Boolean algebra with operators.

Definition 2.3 *For any set A, and any n-ary operator $F : \mathcal{P}(A)^n \to \mathcal{P}(A)$, the underlying (n+1)-ary base relation $F^{\downarrow} \subseteq A^{n+1}$ is defined by:*

$$F^{\downarrow}x_0 \ldots x_{n-1}x_n \text{ iff } x_n \in F(\{x_0\}, \ldots, \{x_{n-1}\}).$$

For any Boolean algebra $\mathcal{P}(\mathcal{A})$ with operators $F_1, \ldots, F_m$, the underlying relational structure $\mathcal{A}$ is defined by:

$$\mathcal{A} = (A; F_1^{\downarrow}, \ldots, F_m^{\downarrow}).$$

Theorem 2.4 *For any relation $R \subseteq A^{n+1}$, $(R^{\uparrow})^{\downarrow} = R$, and for any normal and additive operator $F{:}\mathcal{P}(A)^n \to \mathcal{P}(A)$, $(F^{\downarrow})^{\uparrow} = F$.*

In the context of duality theory, the Jónsson/Tarski construction provides a mechanism to translate between second-order properties defining the possible world semantics of a logic, and properties of operators defined on the Lindenbaum/Tarski algebra of the logic. The next two lemmas list a number of such translations, the first between properties of a unary operation and its corresponding binary relation, and the second between properties of a binary operation and its corresponding ternary relation. Traditional proofs can be found in [Bri89]; our machine-generated proofs appear in section 5.

Lemma 2.5 *Let $F : \mathcal{P}(U) \to \mathcal{P}(U)$ be normal and completely additive, and let $R \subseteq U^2$ be $F^{\downarrow}$, then properties of R correspond to properties of F as (ii) to (i) below. Conversely, let $R \subseteq U^2$ be any relation, and let $F = R^{\uparrow}$, then $F : \mathcal{P}(U) \to \mathcal{P}(U)$ is normal and completely additive, and properties of F correspond to properties of R as (i) to (ii) below.*

(a) (i) $\forall X \subseteq U: \ X \neq \emptyset \Rightarrow F(X) \neq \emptyset$,
(ii) Domain $(R) = U$;
(b) (i) $\forall X \subseteq U: \ X \subseteq F(X)$,
(ii) R is reflexive over U;
(c) (i) $\forall X \subseteq U: \ F(X) \subseteq X$,
(ii) R is the identity relation over a subset of U;
(d) (i) $F : \mathcal{P}(U) \to \mathcal{P}(U)$ is the identity function,
(ii) $R \subseteq U^2$ is the identity relation;
(e) (i) $E \subseteq U$ is a fixed point of $F : F(E) = E$,
(ii) $(\exists e \in E)[Rex]$ iff $x \in E$;
(f) (i) $\forall X \subseteq U: \ F(F(X)) \subseteq F(X)$,
(ii) R is transitive;
(g) (i) $\forall X \subseteq U: \ F(X) \subseteq F(F(X))$,
(ii) R is dense $[\forall x, y \in U : Rxy \Rightarrow (\exists z)[Rxz \& Rzy]]$;
(h) (i) $\forall X, Y \subseteq U: \ F(X) \bigcap Y = \emptyset$ iff $X \bigcap F(Y) = \emptyset$,
(ii) R is symmetric;
(i) (i) F maps singletons onto singletons,
(ii) R is a unary operation over U;
(j) (i) F is an involution over $\mathcal{P}(U)$,
(ii) R is an involution over U.

Notice that the function F is the algebraic counterpart of the $\Diamond$–operator in modal logic. This becomes clear when we compare the definition of F with the semantics of $\Diamond$.

$$x \in F(X) \Leftrightarrow \exists x_0 \ R(x_0, x) \wedge x_0 \in X$$
$$x \models \Diamond X \Leftrightarrow \exists x_0 \ R(x, x_0) \wedge x_0 \models X$$

The only difference is that the arguments of the R–relation are exchanged, which is due to different tradition in the different communities.

The Modal Hilbert axioms corresponding to the above algebraic properties are: (a) $\neg\Diamond\neg X \Rightarrow \Diamond X$, (b) $X \Rightarrow \Diamond X$, (c) $\Diamond X \Rightarrow X$, (d) $\Diamond X \Leftrightarrow X$, (e) ?, (f) $\Diamond\Diamond X \Rightarrow \Diamond X$, (g) $\Diamond X \Rightarrow \Diamond\Diamond X$, (h) $\Diamond\neg\Diamond X \Rightarrow X$, (i) $\Diamond X \Rightarrow \neg\Diamond\neg X$, (j) $\Diamond\Diamond X \Leftrightarrow X$.

Lemma 2.6 *Let $F : \mathcal{P}(U)^2 \to \mathcal{P}(U)$ be normal and completely additive, and let $R \subseteq U^3$ be $F^{\downarrow}$, then properties of R correspond to properties of F as (ii) to (i) below. Conversely, let $R \subseteq U^3$ be any relation, and $F = R^{\uparrow}$, then $F : \mathcal{P}(U)^2 \to \mathcal{P}(U)$ is normal and completely additive, and properties of F correspond to properties of R as (i) to (ii) below.*

(a) (i) F has no divisors of zero $[\forall X, Y \subseteq U: \; F(X,Y) = \emptyset \Rightarrow X = \emptyset$ or $Y = \emptyset]$,
(ii) $\forall x, y \; \in U, \exists z \in U$ such that $Rxyz$;
(b) (i) F is commutative $[\forall X, Y \subseteq U: \; F(X,Y) = F(Y,X)]$,
(ii) F is $(1,2)$-symmetric $[\forall x, y, z \; \subseteq U: \; Rxyz \Rightarrow Ryxz]$;
(c) (i) F is upper semi-idempotent $[\forall X \subseteq U: \; X \subseteq F(X,X)]$,
(ii) R is totally reflexive $[\forall x \subseteq U: \; Rxxx]$;
(d) (i) F is lower semi-idempotent $[\forall X \subseteq U: \; F(X,X) \subseteq X]$,
(ii) R is 3-prime $[\forall x, y, z \; \in U: \; Rxyz \Rightarrow z = x$ or $z = y]$;
(e) (i) F is idempotent $[\forall X \subseteq U: \; F(X,X) = X]$,
(ii) R is totally reflexive and 3-prime;
(f) (i) F associates from left to right $[\forall X, Y, Z \; \subseteq U: \; F(F(X,Y),Z) \subseteq F(X, F(Y,Z))]$,
(ii) R^2 associates from left to right
$[\forall x, y, z, u \; \in U: \; (\exists v)[Rxyv \wedge Rvzu] \Rightarrow (\exists w)[Rxwu \wedge Ryzw]]$
Abbreviation: $[R^2(xy)zu \Rightarrow R^2x(yz)u]$;
(g) (i) F associates from right to left $[\forall X, Y, Z \; \subseteq U: \; F(X, F(Y,Z)) \subseteq F(F(X,Y),Z)]$,
(ii) R^2 associates from right to left
$[\forall x, y, z, u, \; \in U: \; (\exists w)[Rxwu \wedge Ryzw] \Rightarrow (\exists v)[Rxyv \wedge Rvzu]]$
Abbreviation: $[R^2x(yz)u \Rightarrow R^2(xy)zu]$;
(h) (i) F is associative,
(ii) R^2 is associative $[R^2(xy)zu$ iff $R^2x(yz)u]$;
(i) (i) $E \subseteq U$ is a left identity of F $[\forall X \subseteq U: \; F(E,X) = X]$,
(ii) $E \subseteq U$ is a set of left identities of R $[\forall x, y \subseteq U: \; (\exists e \in E)[Rexy]$ iff $x = y]$.

Consider for example the modal logic $S4$. Its Kripke semantics is given by the theory of quasi-orders, sometimes called $S4$-model structures, while its algebraic counterpart is given by the variety of closure algebras. From any $S4$-model structure $\mathcal{A} = (A, R)$, with set of worlds A and quasi-order R, one obtains a corresponding closure algebra $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(A), R^{\uparrow})$ with closure operator $R^{\uparrow}$. Properties of R then translate to properties of $R^{\uparrow}$ as in lemma 2.5. For any closure algebra $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(A), F)$ over the power set of A, one obtains a corresponding $S4$-model structure $\mathcal{A} = (A, F^{\downarrow})$ with quasi-order $F^{\downarrow}$. Properties of F then translate to properties of $F^{\downarrow}$ as in lemma 2.5.

As a second example, consider the relevance logic $\mathcal{R}^{\neg}$, the logic obtained from the (standard) relevance logic $\mathcal{R}$ [AB75] by adding a Boolean negation operation $\neg$. Its Kripke semantics is given by the theory of $\mathcal{R}^{\neg}$-relational structures, while its algebraic counterpart is given by the class of $\mathcal{R}^{\neg}$-algebras. The following definitions are from [Bri89].

Definition 2.7 *An $\mathcal{R}^{\neg}$-model structure is a relational structure $\mathcal{U} = (U; R, ^*, E)$, where $R \subseteq U^3$, $^* : U \to U$ and $E \subseteq U$ are such that:*
(a) R is totally reflexive: $Raaa$,
(b) (1,2)-symmetric: $Rabc \Rightarrow Rbac$,
(c) and has identity elements in E: $(\exists e \in E)[Reab]$ iff $a = b$,
(d) R^2 is associative: $(\exists x)[Rabx \; \& Rxcd]$ iff $(\exists y)[Rayd \; \& Rbcy]$,
(e) * is an involution: $a^{**} = a$,
(f) and R and * obey the rule: $Rabc \Rightarrow Rac^*b^*$.

Definition 2.8 *An $\mathcal{R}^{\neg}$-algebra is an algebra $\mathcal{A} = (A; \vee, \neg, 1, \circ, \sim, e)$ such that:*
(a) $(A; \vee, \neg, 1)$ is a Boolean algebra,
(b) $\sim$ is an involution: $\sim\sim a = a$,
(c) with the De Morgan properties: $\sim (a \vee b) = \sim a \wedge \sim b$,
(d) $(A; \circ, e)$ is a commutative monoid,
(e) which is lattice-ordered: $a \circ (b \vee c) = (a \circ b) \vee (a \circ c)$,
(f) $\circ$ is upper semi-idempotent: $a \leq a \circ a$,
(g) and has the antilogism property: $a \circ b \leq c$ iff $(a \circ \sim c) \leq \sim b$.

In order to show that $\mathcal{R}^{\neg}$-algebras arise from $\mathcal{R}^{\neg}$-model structures by way of the power construction, the operations on $\mathcal{R}^{\neg}$ must be additive. Since the De Morgan negation $\sim$ is not additive, definition 2.8 is reformulated using the star operation $^\star : A \to A$, defined by

$$a^\star = \sim \neg a,$$

instead of $\sim$. The new definition agrees with definition 2.8 above in all respects except these: $\sim$ is replaced by *, and axioms (b), (c) and (g) become, respectively:

(b)$'$ * is an involution: $a^{\star\star} = a$,
(c)$'$ which is additive: $(a \vee b)^\star = a^\star \vee b^\star$,
(g)$'$ $\circ$ has the property: $a \circ (b^\star) \leq \neg c$ iff $a \circ (c^\star) \leq \neg b$.

The advantage of the alternative definition is that it turns an $\mathcal{R}^{\neg}$-algebra into a Boolean algebra with operators. Lemma 2.6 then provides the translation between properties of R and properties of $\circ$, and between properties of $^\sim$ and properties of *.

3 A Framework for Automating Duality

Developing dualities as for example

(b) (i) $\forall X \subseteq U : X \subseteq F(X)$,
(ii) R is reflexive over U

from lemma 2.5 consists of four problems:

Top-Down Direction
1. Given (i), find a suitable candidate for (ii).
2. Verify the equivalence of (i) and (ii).

Bottom-Up Direction
3. Given (ii), find a suitable candidate for (i).
4. Verify the equivalence of (i) and (ii).

Up to now there was no method for solving the problems 1 and 3, except by pure guessing or by very special methods in certain limited cases (Sahlquist formulae in modal logic, for example [vB84]). Of course, people with experience in this, quickly develop enough intuition for solving relatively simple problems of this kind. The more complex the formulae are, however, the less reliable is the intuition.

In contrast to this, our method is fully automatic and solves the guessing problem together with the verification problem in one go.

3.1 The Top–Down Direction

The top-down direction of the duality problem can be stated as follows: Given
(a) some functions F which are defined in terms of other relations and functions using the membership predicate $\in$:

(1) $Def(F,R)$: $\forall X_1,\ldots,X_n\ \forall x\ x \in F(X_1,\ldots,X_n) \Leftrightarrow \Phi$.

where Φ contains *no* occurrence of F, and
(b) a property $\Psi(F)$ of F that can be formulated in first–order predicate logic using again the special membership predicate $\in$, find a formula $\Gamma(R)$ such that

(2) $Def(F,R) \Rightarrow (\Psi(F) \Leftrightarrow \Gamma(R))$.

In the case of power structures, $Def(F,R)$ is given by definition 2.1.

(3) $\forall X_1,\ldots,X_n\ \forall x\ x \in F(X_1,\ldots,X_n)) \Leftrightarrow \exists x_1,\ldots,x_n\ x_1 \in X_1 \wedge \ldots \wedge x_n \in X_n \wedge R(x_1,\ldots,x_n,x)$

The version we need for the examples in lemma 2.5 is

(4) $\forall X\ \forall y\ y \in F(X) \Leftrightarrow \exists x\ x \in X \wedge R(x,y)$

Thus, the first requirement, a suitable predicate logic formulation for the definition of F is fulfilled in the case of power structures.

The second requirement is that the property of F has to be formulated as a predicate logic formula in terms of the $\in$ predicate. The structure of this formula $\Psi(F)$ must be such that application of $Def(F,R)$ as rewrite rule from left to right eliminates F completely and the resulting formula is of the structure

(5) $\Psi' = QX_1,\ldots,X_n\ \Psi''(\ldots \in X_1,\ldots,\ldots \in X_n)$ or equivalently
(6) $\Psi' = QX_1,\ldots,X_n\ \Psi''(X_1(\ldots),\ldots,X_n(\ldots))$

where Q is an existential or a universal quantifier. In the version (6), the set variables X_i have been replaced by their characteristic predicates. This brings to light the second–order nature of the problem which had been hidden in the membership predicate. Since Def is an equivalence, rewriting $\Psi(F)$ to $\Psi'(R)$ is an equivalence transformation in the theory of $Def(F,R)$, i.e. $Def(F,R) \Rightarrow \Psi(F) \Leftrightarrow \Psi'(R)$.

We illustrate this with the property (b)(i) of lemma 2.5: $\forall X \subseteq U :\ X \subseteq F(X)$. First of all, the property is reformulated in terms of the membership predicate

$$\forall X\ \forall y\ y \in X \Rightarrow y \in F(X).$$

The condition $X \subseteq U$ is obsolete because U denotes the whole domain of our interpretation. Now (4) is applied as rewrite rule and we get

$$\forall X\ \forall y\ y \in X \Rightarrow (\exists x\ x \in X \wedge R(x,y)).$$

or

$$\forall X\ \forall y\ X(y) \Rightarrow (\exists x\ X(x) \wedge R(x,y)).$$

respectively. The function F is eliminated now, but the resulting second–order formula is not yet satisfactory. What we are after is a first–order property in terms of the relation R. To this end, a formula $\Gamma(R)$ has to be found which is equivalent to $\Psi'(R)$, but does not contain the predicate variables X_i. This turned out to be the kernel of the problem. It can be solved by a quantifier elimination procedure that computes for a second–order formula an equivalent first–order formula — if there is one and the procedure succeeds. Of course there is not always an equivalent first-order formula, and even if there is one, there is no procedure which is guaranteed to find it. The particular quantifier elimination procedure we shall employ is discussed in some detail in the next section.

To summarize, the recipe for the top-down direction is:

1. Formulate the definition of the functions F in the style of (1).
2. Formulate the property $\Psi(F)$ in terms of the membership predicate.
3. Eliminate F from Ψ.
4. Replace the set variables X_i by their characteristic predicates.
5. Apply quantifier elimination.

3.2 The Bottom–Up Direction

In the bottom–up direction of the duality problem we want to compute from the property $\Gamma(R)$ of the relation R and the definition $Def(F,R)$ for the function F a corresponding property $\Psi(F)$. There are two different methods for computing Ψ. In the first method we exploit that

$$(\exists R\ \Gamma(R) \wedge Def(F,R)) \Leftrightarrow \Psi(F)$$

implies

$$Def(F,R) \Rightarrow (\Gamma(R) \Rightarrow \Psi(F)$$

for the particular Ψ and R. This reduces the problem again to a quantifier elimination problem. The quantifier $\exists R$ has to be eliminated from $\exists R\ \Gamma(R) \wedge Def(F,R)$. If this succeeds, we have a candiate formula which has to be verified with the top–down method. Unfortunately it succeeds only in relatively simple cases. An evidence for failure is that $\Gamma(R)$ is recursive, as for example transitivity.

The second method is much more complicated and it needs some heuristic guidance. It consists of a guessing and verification step. The guessing step, however, can be systematized such that the whole procedure is again fully automatic.

In the guessing step a theorem prover is used for synthesizing a candidate formula as a Skolem term. To this end, the connectives necessary to build $\Psi(F)$ as a term are axiomatized as function symbols and a formula

$$\exists f\ \forall x\ x \in f$$

is proved constructively. The binding $\Psi(F)$ of f used in the proof is the desired candidate formula. We enumerate the proofs and try to verify the generated formula with the top–down method. If there are enough connectives available, the correct result should eventually be found.

Usually there are different options for the formulation of Ψ. If it can be expected that Ψ can be formulated in terms of the set connectives union, intersection, complement and subset, things are simpler. The axioms for these connectives are:

$$\begin{array}{ll}
\forall X, Y\ \forall x\ x \in \mathit{union}(X,Y) & \Leftrightarrow (x \in X \vee x \in Y) \\
\forall X, Y\ \forall x\ x \in \mathit{intersection}(X,Y) & \Leftrightarrow (x \in X \wedge x \in Y) \\
\forall X\ \forall x\ x \in \mathit{complement}(X) & \Leftrightarrow (\neg x \in X) \\
\forall X, Y\ \forall x\ x \in \mathit{subset}(X,Y) & \Leftrightarrow (x \in X \Rightarrow x \in Y)
\end{array}$$

The *subset* connective is actually an abbreviation: $\mathit{subset}(X,Y) = \mathit{complement}(X) \cup Y$ and can be used to model the normal subset relation as a function.

The input to the theorem prover consists of these axioms, together with $Def(F,R)$ and $\Gamma(R)$. The theorem to be proven is $\exists f\ \forall x\ x \in f$. The result are proofs with bindings for f, for example $f = \mathit{subset}(F(X), F(F(X)))$, i.e. $F(X) \subseteq F(F(X))$.

Things get more complicated if the formula Ψ contains special predicates on sets and logical connectives. The correspondence

(i) (i) F maps singletons onto singletons,
(ii) R is a unary operation over U

of lemma lemma 2.5 is such a case. In order to synthesize 'F maps singletons onto singletons' we must axiomatize a predicate *singleton*, the connective *implies* and synthesize the term

$$\mathit{implies}(\mathit{singleton}(X), \mathit{singleton}(F(X))).$$

That means in particular that *singleton* must be defined as a function. This is only possible by means of a *Holds*–predicate:

$$\forall X\ \mathit{Holds}(\mathit{singleton}(X)) \Leftrightarrow \forall x, y\ x \in X \wedge y \in X \Rightarrow x = y$$

The axiom for the implication as function is

$$\forall X\ \mathit{Holds}(\mathit{implies}(X,Y)) \Leftrightarrow \mathit{Holds}(X) \Rightarrow \mathit{Holds}(Y).$$

From axioms of this kind we could try to prove $\exists f\ \mathit{Holds}(f)$. Unfortunately this approach turned out to be intractable for technical reasons. Since automated theorem provers usually negate the theorem and search for a refutation, the negation of $\exists f\ \mathit{Holds}(f)$ which is $\forall f \neg \mathit{Holds}(f)$ is added to the formula set. This causes the problem that all formulae with negated occurrence of *Holds* get subsumed and deleted. The problem becomes unsolvable. Turning subsumption off is no solution because the search space gets so terribly large that no interesting theorem can be proven.

A much more elegant solution comes from the possible worlds idea in modal logic. We make the $Holds$–predicate and the $\in$–relation world dependent. That means we use the definitions

$$\forall w\ \forall X\ Holds(w, singleton(X)) \Leftrightarrow \forall x, y\ \in (w, x, X) \wedge \in (w, y, X) \Rightarrow x = y$$
$$\forall w\ \forall X\ Holds(w, implies(X, Y)) \Leftrightarrow Holds(w, X) \Rightarrow Holds(w, Y).$$

and prove a theorem $\exists f\ \forall w\ Holds(w, f)$. From a logical point of view, this 'world'–argument is redundant, but it avoids the subsumption problem. The negated theorem $\forall f\ \exists w\ \neg Holds(w, f)$ gets Skolemized to $\neg Holds(k(w), f)$ and does not subsume anything.

Summarizing, we propose the following procedure for computing $\Psi(F)$ from $Def(F, R)$ and $\Gamma(R)$:

1. Try quantifier elimination for $\exists R\ Def(R, F) \wedge \Gamma(R)$.
   If this does not succeed:
2. Try to find a solution in terms of set connectives.
   (a) Axiomatize the set connectives.
   (b) From these axioms together with $Def(F, R)$ and $\Gamma(R)$ prove the theorem
   $\exists f\ \forall x\ x \in f$.
   (c) Each binding for f is a candidate for $\Psi(F)$ that needs to be verified with the top–down method.
3. Try to find a solution in terms of general connectives and predicates on sets.
   (a) Axiomatize the connectives and the predicates on sets (for example 'singleton', 'emptyset' etc.) with a world dependent $Holds$–predicate
   (b) From these axioms together with $Def(F, R)$ and $\Gamma(R)$ prove the theorem
   $\exists f\ \forall w\ Holds(w, f)$.
   (c) Each binding for f is a candidate for $\Psi(F)$ that needs to be verified with the top–down method.

In step (c) many theorems have to be enumerated with a theorem prover. Each binding for the variable f represents a tautology. Unfortunately there are many tautologies and almost all of them are useless for our purposes. There is however a general heuristic for restricting the number of useless theorems: Check for each proof whether it uses the clauses to be translated. If not, the binding of f represents a useless tautology. From now on, all clauses containing terms which are instances of this binding can be deleted. For example if the first binding is $f = i(x, x)$ which stands for $X \Rightarrow X$ then henceforth all derived clauses with instances of $i(x, x)$ occurring somewhere can be deleted. Each useless proof then contributes to a further restriction of the search space.

The procedures will be illustrated in detail in the examples section.

4 Quantifier Elimination

In [GO92] we have developed an algorithm which can compute for second–order formulae of the kind $\exists P_1, \ldots, P_k\ \Phi$ where Φ is a first–order formula, an equivalent first–order formula — if there is one. Since $\forall P_1, \ldots, P_k\ \Phi \Leftrightarrow \neg\exists P_1, \ldots, P_k\ \neg\Phi$ this algorithm can also eliminate universal quantifiers by first negating the formula, eliminating the existential quantifiers and then negating the result. Related methods can also be found in [Ack35a, Ack35b, Ack54, Sza92, BGW92, Sim93]. The definition of the algorithm is:

Definition 4.1 (The SCAN Algorithm)
Input to SCAN is a formula $\alpha = \exists P_1, \ldots, P_n\ \psi$ with predicate variables $P_1, \ldots, P_n$ and an arbitrary first–order formula ψ.
Output of the SCAN — if it terminates — is a formula φ_α which is *logically equivalent* to α, but not containing the predicate variables $P_1, \ldots, P_n$.
SCAN performs the following three steps:

1. ψ is transformed into clause form.

2. All C–resolvents and C–factors with the predicate variables $P_1, \ldots, P_n$ have to be generated. C–resolution ('C' for constraint) is defined as follows:

$$\frac{\begin{array}{ll} P(s_1,\ldots,s_n) \vee C & P(\ldots) \text{ and } \neg P(\ldots) \\ \neg P(t_1,\ldots,t_n) \vee D & \text{are the } \textit{resolution literals} \end{array}}{C \vee D \vee s_1 \neq t_1 \vee \ldots \vee s_n \neq t_n}$$

and the C-factorization rule is defined analogously:

$$\frac{P(s_1,\ldots,s_n) \vee P(t_1,\ldots,t_n) \vee C}{P(s_1,\ldots,s_n) \vee C \vee s_1 \neq t_1 \vee \ldots \vee s_n \neq t_n}.$$

Notice that only C-resolutions between different clauses are allowed (no self resolution). A C-resolution or C-factorization can be optimized by destructively resolving literals $x \neq t$ where the variable x does not occur in t with the reflexivity equation. C–resolution and C–factorization takes into account that second order quantifiers may well impose conditions on the interpretations which must be formulated in terms of equations and inequations.

As soon as *all* resolvents and factors between a particular literal and the rest of the clause set have been generated (the literal is 'resolved away'), the clause containing this literal *must* be deleted (purity deletion). This deletion step is responsible for getting rid of the predicate to be eliminated. If all clauses are deleted this way, this means that α is a tautology.

All equivalence preserving simplifications may be applied freely. These are for example:

- Tautologous resolvents can be deleted.
- Subsumed clauses can be deleted.
- Subsumption factoring can be performed. Subsumption factoring means that a factor subsumes its parent clause. This may be realized by just deleting some literals. For example $Q(x) \vee Q(a)$, where x is a variable, can be simplified to $Q(a)$.
- Subsumption resolution can also be performed. Subsumption resolution means that a resolvent subsumes its parent clause, and this again may be realized by deleting some literals [OS91]. For example the resolvent between $P \vee Q$ and $\neg P \vee Q \vee R$ is just $Q \vee R$ such that $\neg P$ can be deleted from the clause.

If an empty clause is generated, this means that α is contradictory.

3. If the previous step terminates and there are still clauses left then reverse the Skolemization. If this is not possible, the only chance is to take parallel (second–order) Henkin quantifiers [Hen61] or to leave the Skolem functions existentially quantified.

$\triangleleft$

The next example illustrates the different steps of the SCAN algorithm in more detail. The input is: $\exists P\ \forall x, y\ \exists z\ (\neg P(a) \vee Q(x)) \wedge (P(y) \vee Q(a)) \wedge P(z)$.
In the first step the clause form is to be computed:

$$C_1 : \neg P(a) \vee Q(x)) \qquad C_2 : \ \ P(y) \vee Q(a) \qquad C_3 : \ \ P(f(x,y))$$

f is a Skolem function. In the second step of SCAN we begin by choosing $\neg P(a)$ to be resolved away. The resolvent between C_1 and C_2 is $C_4 = Q(x) \vee Q(a)$ which is equivalent to $Q(a)$ (this is one of the equivalence preserving simplifications). The C-resolvent between C_1 and C_3 is $C_5 = (a \neq f(x,y) \vee Q(x))$. There are no more resolvents with $\neg P(a)$. Therefore C_1 is deleted. We are left with the clauses

$$\begin{array}{llll} C_2 & P(y) \vee Q(a) & C_3 & P(f(x,y)) \\ C_4 & Q(a) & C_5 & a \neq f(x,y) \vee Q(x) \end{array}.$$

Selecting the next two P-literals to be resolved away yields no new resolvents. Thus, C_2 and C_3 are simply to be deleted as well. All P-literals have now been eliminated. Restoring the quantifiers we then get

$$\forall x\ \exists z\ Q(a) \wedge (a \neq z \vee Q(x))$$

as the final result.

The SCAN algorithm is correct in the sense that its result is really equivalent to the input formula. It cannot be complete, i.e. there may be second–order formulae which have a first–order equivalent, but SCAN (as any other algorithm) cannot find it. Completeness is not possible, otherwise the theory of arithmetic would be enumerable.

The points where SCAN does not compute a first–order equivalent are (i) the resolution does not terminate and (ii) reversing Skolemization is not possible. In the second case there is a (again second–order) solution in terms of parallel Henkin quantifiers or existentially quantified Skolem functions.

5 Examples

In [BGO93] we go through the examples of lemma 2.5 and lemma 2.6 one by one and show the top–down and bottom-up solutions together with the technical details necessary to repeat the experiment with the OTTER theorem prover ([McC89] or in [WOLB91])[1]. The list of examples was compiled (in [Bri89]) before our method had been developed. Thus, this is *not* a selection of cases where our method just happens to work. In this paper, however only three of the examples are presented.

Let us briefly repeat the recipe for the top–down direction
1. Formulate the definition of the functions F in the style of (1).
2. Formulate the property $\Psi(F)$ in terms of the membership predicate.
3. Eliminate F from Ψ.
4. Replace the set variables by their characteristic predicates.
5. Apply quantifier elimination.

The SCAN algorithm proceeds as follows:

6. If the formula is universally quantified: Negate it.
7. Generate a clause form.
8. 'Resolve away' all predicates to be eliminated and delete the pure clauses afterwards.
9. Reconstruct the quantifiers.
10. If the original formula had been negated, negate the result again.

The only example in our list which requires more than two or three resolution steps is example (f) for ternary relations. We use this example to demonstrate the use of the OTTER theorem prover for performing quantifier elimination. In all the other examples quantifier elimination is done by hand.

The following abbreviations are used for the connectives and predicates:

e	= membership relation $\in$
H	= *Holds*–predicate
ne	= not emptyset–predicate $\neg\emptyset$
s	= subset connective $\subseteq$
i	= implication $\Rightarrow$.

The naming convention for variable symbols and constant symbols is: Variable symbols are taken from the end of the alphabet u, v, w, x, y, z. Constant symbols which are generated from existential quantifiers are written underlined. That means for example $\underline{x}$ is a Skolem constant stemming from a '$\exists x$' quantification.

The CPU times for most of the examples range from a few seconds to a few minutes on a Macintosh IIfx. Only for example (a) for ternary relations, more than 1 hour was needed on a Solburn machine.

[1] We have used OTTER as it is, but with certain tricky application of demodulation and the weighting mechanism. In a particularliy tailored implementation, which is on the way, this will not be necessary any longer.

5.1 Binary Relations

We begin with the examples of lemma 2.5. They correlate a binary relation R with a unary function F. The definition of F for this case (def. 2.1) is used as a rewrite rule:

(r1) $\quad (y \in F(X)) \to (\exists x\ x \in X \wedge R(x,y))$

Example a

(a) (i) $\forall X \subseteq U :\ X \neq \emptyset \Rightarrow F(X) \neq \emptyset,$
(ii) Domain $(R) = U$

Direction (i) $\to$ (ii) (Top–Down)

Logical Formulation: $(\exists x\ x \in X) \Rightarrow (\exists y\ y \in F(X))$
Rewritten: $(\exists x\ X(x)) \Rightarrow (\exists y\ (\exists x\ X(x) \wedge R(x,y)))$
Negated and Clausified: $X(\underline{x})$
$\neg X(x), \neg R(x,y)$
X 'Resolved Away': $\neg R(\underline{x}, y)$
Quantifiers Reconstructed: $\exists x\ \forall y\ \neg R(x,y)$
Negated again: $\forall x\ \exists y\ R(x,y)$ (which means $domain(R) = U$)

Direction (ii) $\to$ (i) (Bottom-Up)

First trial: Quantifier Elimination
$\exists R\ \Gamma(R) \wedge Def(F,R) =$
$\exists R\ (\forall x\ \exists y\ R(x,y)) \wedge (\forall y\ e(y,F(X)) \Leftrightarrow \exists x\ e(x,X) \wedge R(x,y))$

Clauses:
$R(x, f(x))$
$\neg e(y,F(X)), e(g(y),X)$
$\neg e(y,F(X)), R(g(y),y)$
$\neg e(x,X), \neg R(x,y), e(y,F(X))$

R resolved away
$\neg e(y,F(X)), e(g(y),X)$
$\neg e(x,X), e(f(x),F(X))$

Quantifiers reconstructed:
$\forall y\ e(y,F(X)) \Rightarrow \exists z\ e(z,X)$
$\forall x\ e(x,X) \Rightarrow \exists y\ e(y,F(X))$

The set notation of the combined formula is:
$(\exists x\ x \in X \Leftrightarrow (\exists y\ y \in F(X))$ which implies $X \neq \emptyset \Rightarrow F(X) \neq \emptyset$

Although the quantifier elimination trial was successful, we also show the theorem prover version. It illustrates the use of the world dependent *Holds*–predicate. The | sign in the OTTER protocoll denotes the logical or. The `$ans`–literals are a technical means for recording bindings of variables. From a logical point of view they are simply nonexisting.

OTTER **Protocol:**

```
formula_list(usable).
(all w (all z (all X (e(w,z,F(X)) <-> (exists x (e(w,x,X) & R(x,z))))))).
(all w (all X (all Y (H(w,i(X,Y)) <-> (H(w,X) -> H(w,Y)))))).
(all w (all X (H(w,ne(X)) <-> (exists x e(w,x,X))))).
end_of_list.
```

```
formula_list(sos).
(all x (exists y R(x,y))).
-(exists f (all w (H(w,f) & -$ans(f)))).
end_of_list.

Clauses:

list(usable).
7  -e(w,z,F(x1)) | e(w,$f1(w,z,x1),x1).      8  -e(w,z,F(x1)) | R($f1(w,z,x1),z).
9  e(w,z,F(x1)) | -e(w,x,x1) | -R(x,z).      10 -H(w,i(x2,x3)) | -H(w,x2) | H(w,x3).
11 H(w,i(x2,x3)) | H(w,x2).                  12 H(w,i(x2,x3)) | -H(w,x3).
13 -H(w,ne(x4)) | e(w,$f2(w,x4),x4).         14 H(w,ne(x4)) | -e(w,x,x4).
end_of_list.

list(sos).
15  R(x,$f3(x)).
16  -H($f4(x5),x5) | $ans(x5).
end_of_list.

---------------- PROOF ----------------
9  e(x,y,F(z)) | -e(x,u,z) | -R(u,y).
11  H(x,i(y,z)) | H(x,y).
12  H(x,i(y,z)) | -H(x,z).
13  -H(x,ne(y)) | e(x,$f2(x,y),y).
14  H(x,ne(y)) | -e(x,z,y).
15  R(x,$f3(x)).
16  -H($f4(x),x) | $ans(x).
17 [binary,15,9] e(x,$f3(y),F(z)) | -e(x,y,z).
19 [binary,16,12] $ans(i(x,y)) | -H($f4(i(x,y)),y).
21 [binary,16,11] $ans(i(x,y)) | H($f4(i(x,y)),x).
26 [binary,19,14] $ans(i(x,ne(y))) | -e($f4(i(x,ne(y))),z,y).
30 [binary,21,13] $ans(i(ne(x),y)) | e($f4(i(ne(x),y)),$f2($f4(i(ne(x),y)),x),x).
48 [binary,17,26] -e($f4(i(x,ne(F(y)))),z,y) | $ans(i(x,ne(F(y)))).
49 [binary,48,30] $ans(i(ne(v64),ne(F(v64)))).
------------ end of proof -------------
```

In set notation, the answer is $\forall X\ X \neq \emptyset \Rightarrow F(X) \neq \emptyset$

Example f

(f) (i) $\forall X \subseteq U:\ F(F(X)) \subseteq F(X)$,
(ii) R is transitive

Direction (i) → (ii) (Top–Down)

Logical Formulation:	$\forall z\ z \in F(F(X)) \Rightarrow z \in F(X)$
Rewritten:	$\forall z\ (\exists y\ (\exists x\ X(x) \wedge R(x,y)) \wedge R(y,z)) \Rightarrow (\exists u\ X(u) \wedge R(u,z))$
Negated and Clausified:	$X(\underline{x})$
	$R(\underline{x},\underline{y})$
	$R(\underline{y},\underline{z})$
	$\neg X(u), \neg R(u,\underline{z})$
X 'Resolved Away':	$R(\underline{x},\underline{y})$
	$R(\underline{y},\underline{z})$
	$\neg R(\underline{x},\underline{z})$
Quantifiers Reconstructed:	$\exists x,y,z\ \neg R(x,y) \wedge R(y,z) \wedge \neg R(x,z)$

Negated again: $\forall x, y, z\ R(x,y) \wedge R(y,z) \Rightarrow R(x,z)$

Direction (ii) → (i) (Bottom-Up)

Otter **Protocol**

```
formula_list(usable).
(all z (all X (e(z,F(X)) <-> (exists x (e(x,X) & R(x,z)))))).
(all z (all X (all Y (e(z,s(X,Y)) <-> (e(z,X) -> e(z,Y)))))).
end_of_list.

formula_list(sos).
(all x (all y (all z ((R(x,y) & R(y,z)) -> R(x,z))))).
-(exists f (all z (e(z,f) & -$ans(f)))).
end_of_list.

Clauses:
list(usable).
3 -e(z,F(x1)) | e($f1(z,x1),x1).      4 -e(z,F(x1)) | R($f1(z,x1),z).
5 e(z,F(x1)) | -e(x,x1) | -R(x,z).    6 -e(z,s(x2,x3)) | -e(z,x2) | e(z,x3).
7 e(z,s(x2,x3)) | e(z,x2).            8 e(z,s(x2,x3)) | -e(z,x3).
end_of_list.

list(sos).
9 -R(x,y) | -R(y,z) | R(x,z).         10 -e($f2(x4),x4) | $ans(x4).
end_of_list.

---------------- PROOF ----------------
3  -e(x,F(y)) | e($f1(x,y),y).        4  -e(x,F(y)) | R($f1(x,y),x).
5  e(x,F(y)) | -e(z,y) | -R(z,x).     7  e(x,s(y,z)) | e(x,y).
8  e(x,s(y,z)) | -e(x,z).             9  -R(x,y) | -R(y,z) | R(x,z).
10  -e($f2(x),x) | $ans(x).
12 [hyper,10,7] $ans(s(x,y)) | e($f2(s(x,y)),x).
15 [hyper,12,4] $ans(s(F(x),y)) | R($f1($f2(s(F(x),y)),x),$f2(s(F(x),y))).
16 [hyper,12,3] $ans(s(F(x),y)) | e($f1($f2(s(F(x),y)),x),x).
19 [hyper,16,4] $ans(s(F(F(x)),y))
             R($f1($f1($f2(s(F(F(x)),y)),F(x)),x),$f1($f2(s(F(F(x)),y)),F(x))).
20 [hyper,16,3] $ans(s(F(F(x)),y)) | e($f1($f1($f2(s(F(F(x)),y)),F(x)),x),x).
115 [hyper,19,9,15] $ans(s(F(F(x)),y)) |
             R($f1($f1($f2(s(F(F(x)),y)),F(x)),x),$f2(s(F(F(x)),y))).
171 [hyper,115,5,20] $ans(s(F(F(x)),y)) | e($f2(s(F(F(x)),y)),F(x)).
174 [hyper,171,8] $ans(s(F(F(x)),y)) | e($f2(s(F(F(x)),y)),s(z,F(x))).
175 [binary,174,10] $ans(s(F(F(x)),F(x))).
------------ end of proof -------------
```

In set notation, the answer is $F(F(X)) \subseteq F(X)$.

5.2 Ternary Relations

We continue with an example of lemma 2.6. It correlates a ternary relation R with a binary function F. The definition of F for this case (def. 2.1) is used as a rewrite rule:

(r1) $(z \in F(X,Y)) \rightarrow (\exists x, y\ x \in X \wedge y \in Y \wedge R(x,y,z))$

Example f

(f) (i) F associates from left to right $[\forall X, Y, Z \subseteq U :\ F(F(X,Y),Z) \subseteq F(X, F(Y,Z))]$,

(ii) R^2 associates from left to right
$[\forall x, y, z, u \ \in U : \ (\exists v)[Rxyv \wedge Rvzu] \Rightarrow (\exists w)[Rxwu \wedge Ryzw]]$
Abbreviation: $[R^2(xy)zu \Rightarrow R^2x(yz)u]$

Direction (i) $\rightarrow$ (ii) (Top–Down)

Logical Formulation:

$$\forall z \ z \in F(F(X,Y),Z) \Rightarrow z \in F(F(X,Y),Z)$$

Rewritten:

$$\forall z \ (\exists x, y \ (\exists x_1, y_1 \ x_1 \in X \wedge y_1 \in Y \wedge R(x_1,y_1,x)) \wedge y \in Z \wedge R(x,y,z)) \Rightarrow$$
$$(\exists x, y \ x \in X \wedge (\exists x_1, y_1 \ x_1 \in X \wedge y_1 \in Y \wedge R(x_1,y_1,y)) \wedge R(x,y,z))$$

Since this example is quite complicated, we use it for demonstrating the application of the theorem prover for generating all the resolvents.

OTTER **Protocol:**

```
formula_list(sos).
-(all z ((exists x (exists y ((exists x1 (exists y1
(X(x1) & Y(y1) & R(x1,y1,x)))) & Z(y) & R(x,y,z)))) ->
(exists x (exists y (X(x) &
(exists x1 (exists y1 (Y(x1) & Z(y1) & R(x1,y1,y)))) & R(x,y,z)))))).
end_of_list.

-------> sos clausifies to:
1  X($c2).
2  Y($c1).
3  R($c2,$c1,$c4).
4  Z($c3).
5  R($c4,$c3,$c5).
6  -X(x) | -Y(y) | -Z(z) | -R(y,z,u) | -R(x,u,$c5).
7 [factor,6] -X(x) | -Y(x) | -Z($c5) | -R(x,$c5,$c5).

** KEPT: 25 [binary,23,4] -R($c1,$c3,x) | -R($c2,x,$c5).
```

Clauses 3,5 and 25 are the only generated clauses consisting of R–literals only .
Quantifiers Reconstructed:

$$\exists c1, c2, c3, c4, c5 \ R(c2, c1, c4) \wedge R(c4, c3, c5) \wedge (\forall x \ \neg R(c1, c3, x) \vee \neg R(c2, x, c5))$$

Negated again:

$$\forall c1, c2, c3, c4, c5 \ (R(c2, c1, c4) \wedge R(c4, c3, c5)) \Rightarrow (\exists x \ R(c1, c3, x) \wedge R(c2, x, c5)),$$

or in a more readable notation:

$$\forall x, y, z, u \ (\exists v \ R(x, y, v) \wedge R(v, z, u)) \Rightarrow (\exists w \ R(x, w, u) \wedge R(y, z, w))$$

Direction (ii) $\rightarrow$ (i) (Bottom-Up)

OTTER **Protocol:**

```
formula_list(sos).
(all w all z all X all Y (e(w,z,F(X,Y)) <->
          (exists x exists y (e(w,x,X) & e(w,y,Y) & R(x,y,z))))).
(all z all X all Y (e(z,s(X,Y)) <-> (e(z,X) -> e(z,Y)))).
(all x all y all z all u ((exists v (R(x,y,v) & R(v,z,u)))
          -> (exists w (R(y,z,w) & R(x,w,u))))).
-(exists f (all z (e(z,f) & -$ans(f)))).
end_of_list.

Clauses:
```

```
list(sos).
2 -e(z,F(x1,x2)) | e($f2(z,x1,x2),x1).
3 -e(z,F(x1,x2)) | e($f1(z,x1,x2),x2).
4 -e(z,F(x1,x2)) | R($f2(z,x1,x2),$f1(z,x1,x2),z).
5 e(z,F(x1,x2)) | -e(x,x1) | -e(y,x2) | -R(x,y,z).
6 -e(z,s(x3,x4)) | -e(z,x3) | e(z,x4).
7 e(z,s(x3,x4)) | e(z,x3).
8 e(z,s(x3,x4)) | -e(z,x4).
9 -R(x5,x6,x) | -R(x,x7,x8) | R(x6,x7,$f3(x5,x6,x7,x8)).
10 -R(x5,x6,x) | -R(x,x7,x8) | R(x5,$f3(x5,x6,x7,x8),x8).
11 -e($f4(x9),x9) | $ans(x9).
end_of_list.

----> UNIT CONFLICT at 265.00 sec ---->
-------------- PROOF ----------------
2  -e(x,F(y,z)) | e($f2(x,y,z),y).
3  -e(x,F(y,z)) | e($f1(x,y,z),z).
4  -e(x,F(y,z)) | R($f2(x,y,z),$f1(x,y,z),x).
5  e(x,F(y,z)) | -e(u,y) | -e(v,z) | -R(u,v,x).
7  e(x,s(y,z)) | e(x,y).
8  e(x,s(y,z)) | -e(x,z).
9  -R(x,y,z) | -R(z,u,v) | R(y,u,$f3(x,y,u,v)).
10  -R(x,y,z) | -R(z,u,v) | R(x,$f3(x,y,u,v),v).
11  -e($f4(x),x) | $ans(x).
13 [ur,7,11] e($f4(s(x,y)),x) | $ans(s(x,y)).
16 [ur,8,11] -e($f4(s(x,y)),y) | $ans(s(x,y)).
21 [ur,2,13] e($f2($f4(s(F(x,y),z)),x,y),x) | $ans(s(F(x,y),z)).
22 [ur,3,13] e($f1($f4(s(F(x,y),z)),x,y),y) | $ans(s(F(x,y),z)).
24 [ur,21,3] $ans(s(F(F(x,y),z),u)) | e($f1($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),y).
25 [ur,21,2] $ans(s(F(F(x,y),z),u)) |
             e($f2($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),x).
30 [ur,4,21] R($f2($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
                         $f1($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
                         $f2($f4(s(F(F(x,y),z),u)),F(x,y),z)) |
             $ans(s(F(F(x,y),z),u)).
31 [ur,4,13] R($f2($f4(s(F(x,y),z)),x,y),$f1($f4(s(F(x,y),z)),x,y),$f4(s(F(x,y),z))) |
             $ans(s(F(x,y),z)).
263 [ur,30,10,31] $ans(s(F(F(x,y),z),u)) | R($f2($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
             $f3($f2($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
             $f1($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
             $f1($f4(s(F(F(x,y),z),u)),F(x,y),z),$f4(s(F(F(x,y),z),u))),
             $f4(s(F(F(x,y),z),u))).
264 [ur,30,9,31] $ans(s(F(F(x,y),z),u)) |
             R($f1($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),$f1($f4(s(F(F(x,y),z),u)),
             F(x,y),z),$f3($f2($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
             $f1($f2($f4(s(F(F(x,y),z),u)),F(x,y),z),x,y),
             $f1($f4(s(F(F(x,y),z),u)),F(x,y),z),$f4(s(F(F(x,y),z),u)))).
279 [ur,263,5,16,25] $ans(s(F(F(x,y),z),F(x,u))) |
             -e($f3($f2($f2($f4(s(F(F(x,y),z),F(x,u))),F(x,y),z),x,y),
             $f1($f2($f4(s(F(F(x,y),z),F(x,u))),F(x,y),z),x,y),
             $f1($f4(s(F(F(x,y),z),F(x,u))),F(x,y),z),$f4(s(F(F(x,y),z),F(x,u)))),u).
306 [ur,279,5,24,22] $ans(s(F(F(x,y),z),F(x,F(u,v)))) |
             -R($f1($f2($f4(s(F(F(w,u),v6),v7)),F(w,u),v6),w,u),
             $f1($f4(s(F(v8,v),v9)),v8,v),$f3($f2($f2($f4(s(F(F(x,y),z),
             F(x,F(u,v)))),F(x,y),z),x,y),$f1($f2($f4(s(F(F(x,y),z),F(x,F(u,v)))),
             F(x,y),z),x,y),$f1($f4(s(F(F(x,y),z),F(x,F(u,v)))),F(x,y),z),
```

```
            $f4(s(F(F(x,y),z),F(x,F(u,v))))) |
            $ans(s(F(F(w,u),v6),v7)) | $ans(s(F(v8,v),v9)).
307 [binary,306,264] $ans(s(F(F(v64,v65),v66),F(v64,F(v65,v66)))).
------------ end of proof -------------
```

In set notation, the answer is $F(F(X,Y),Z) \subseteq F(X,F(Y,Z))$.

6 Summary

We have shown how corresponding properties of relations in a structure and the corresponding functions in the power structure can be computed automatically in both directions. In the direction from power structures to structures we used a quantifier elimination method whereas in the other direction an automated guess and verify method was proposed. The guessing part, however, could also be automated using a theorem prover which can enumerate proofs.

The duality problem for power structures is prototypical for many other applications, in particular for computing the correspondences between an axiomatic description of a logic by means of an Hilbert calculus and its model theoretic semantics. The examples we have investigated in this paper can be transferred directly to modal logic and relevance logic.

References

[AB75] A.R. Anderson and N.D. Belnap. *Entailment: The Logic of Relevance and Necessity.* Princeton University Press, 1975.

[Abr87] S. Abramsky. Domain theory in logical form. In *Proceedings, Symposium on Logic in Computer Science*, pages 47–53, Ithaca, NY, 1987.

[Ack35a] Wilhelm Ackermann. Untersuchung über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110:390–413, 1935.

[Ack35b] Wilhelm Ackermann. Zum Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 111:61–63, 1935.

[Ack54] Wilhelm Ackermann. *Solvable Cases of the Decision Problem.* North–Holland Pu. Co., 1954.

[BGO93] Chris Brink, Dov Gabbay, and Hans Jürgen Ohlbach. Towards automating duality. Technical Report MPI-I-93-220, Max Planck Institut für Informatik, Saarbrücken, Im Stadtwald, 6600 Saarbrücken, 1993.

[BGW92] Leo Bachmair, Harald Ganzinger and Uwe Waldmann. Theorem proving for hierarchic first-order theories, 1992. To appear in Proc. ALP'92, Lecture Notes in Comp. Science.

[BP89] W.J. Blok and D. Pigozzi. Algebraizable logics. *Memoirs of the American Mathematical Society*, 77(396):1–78, January 1989.

[Bri89] C. Brink. $\mathbf{R}^{\neg}$-algebras and $\mathbf{R}^{\neg}$-model structures as power constructs. *Studia Logica*, 48:85-109, 1989.

[Bri93] C. Brink. Power structures. *Algebra Universalis*, 30:177–216, 1993.

[BS84] R. Bull and K. Segerberg. Basic modal logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 1–88. Reidel, Dordrecht, Holland, 1984.

[BS85] R.J. Brachman and J.G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[BS92] C. Brink and R.A. Schmidt. Subsumption computed algebraically. *Computers and Mathematics with Applications*, 23:329–342, 1992. Special Issue on Semantic Networks in Artificial Intelligence.

[Cra74] W. Craig. *Logic in algebraizable form*, volume 72 of *Studies in Logic and Foundations of Mathematics.* North-Holland, 1974.

[Gin88] M.L. Ginsberg. Multivalued logics: a uniform apporoach to reasoning in artificial intelligence. *Computing Intelligence*, 4:265–316, 1988.

[GO92] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second–order predicate logic. *South African Computer Journal*, 7:35–43, July 1992. Appeared also in Proc. of KR92, Morgan Kaufmann, 1992, pages 425–436.

[Hal62] P.R. Halmos. *Algebraic Logic.* Chelsea, New York, 1962.

[Han83] G. Hansoul. A duality for Boolean algebras with operators. *Algebra Universalis*, 17:34–49, 1983.

[Hen61] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods*, pages 167–183. Pergamon Press, Oxford, 1961.

[HJ87] C.A.R. Hoare and He Jifeng. The weakest prespecification. *Information Processing Letters*, 24:127–132, 1987.

[HMT71] L. Henkin, J.D. Monk, and A. Tarski. *Cylindric Algebras I*, volume 64 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, Amsterdam, 1971.

[HMT85] L. Henkin, J.D. Monk, and A. Tarski. *Cylindric Algebras II*, volume 115 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, Amsterdam, 1985.

[Joh82] P.T. Johnstone. *Stone spaces.* Cambridge studies in advanced mathematics. Cambridge University Press, 1982.

[Jón] B. Jónsson. Program specifications as Boolean operators: A very preliminary draft. Manuscript,Department of Mathematics, Vanderbilt University, Nashville, TN, 1991.

[Jón92] B. Jónsson. A survey of Boolean algebras with operators. Congress of Young Mathematicians, Montreal, 1991.

[JT51] B. Jónsson and A. Tarski. Boolean algebras with operators, Part I. *American Journal of Mathematics*, 73:891–939, 1951.

[JT52] B. Jónsson and A. Tarski. Boolean algebras with operators, Part II. *American Journal of Mathematics*, 74:127–162, 1952.

[Koz81] D. Kozen. On the duality of dynamic algebras and Kripke models. In E. Engeler, editor, *Logic of Programs*, volume 125 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1981.

[Kri59] S.A. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24:1–14, 1959.

[McC89] William W. McCune. *OTTER User's Guide.* Mathematical and Computer Science Division, Argonne National Laboratory, april 1989.

[Nem92] I. Nemeti. Algebraizations of quantifier logics, an introductory overview. *Studia Logica*, to appear.

[OS91] Hans Jürgen Ohlbach and Jörg H. Siekmann. The Markgraf Karl refutation procedure. In Jean Luis Lassez and Gordon Plotkin, editors, *Computational Logic, Essays in Honor of Alan Robinson*, pages 41–112. MIT Press, 1991.

[Pra90] V.R. Pratt. Dynamic algebras as a well-behaved fragment of relation algebras. In C.H. Bergman, R.D. Maddux, and D.L. Pigozzi, editors, *Algebraic Logic and Universal Algebra in Computer Science*, volume 425 of *Lecture Notes in Computer Science*, pages 77–110. Springer-Verlag, 1990.

[Pri70] H.A. Priestley. Representation of distributive lattices by means of ordered Stone spaces. *Bulletin of the London Mathematical Society*, 2:186–190, 1970.

[Sch91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI 91* (Sydney, 24–30 August 1991), pages 466–471. Morgan Kaufmann, 1991.

[Sim93] Harold Simmons. The Monotonous Elimination of Predicate Variables. Forthcoming in Journal of Logic and Computation, 1993.

[Sto37] M.H. Stone. The theory of representations for Boolean algebras. *American Mathematical Society*, 6:37–111, 1937.

[Sza92] Andrzej Szałas. On correspondence between modal and classical logic: Automated approach. Technical Report MPI–I–92–209, Max Planck Institut für Informatik, Saarbrücken, March 1992.

[Tar41] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6:73–89, 1941.

[TG87] Alfred Tarski and Steven Givant. *A Formalization of Set Theory without Variables.* American Mathematical Society Colloquium Publications 41, Providence, Rhode Island, 1987.

[vB84] Johan van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 167–248. Reidel, Dordrecht, Holland, 1984.

[WOLB91] Larry Wos, Ross Overbeek, Ewing Lusk and Jim Boyle. *Automated Reasoning. Introduction and Applications*, Second Edition, McGraw–Hill Inc., 1991.