MAX-PLANCK-INSTITUT FÜR INFORMATIK

Translation Methods for
Non-Classical Logics

An Overview

Hans Jürgen Ohlbach

MPI–I–93–225 July 1993

Im Stadtwald
W 66123 Saarbrücken
Germany

Author's Address

Hans Jürgen Ohlbach
Max–Planck–Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken,
F. R. Germany
email: ohlbach@mpi-sb.mpg.de

Publication Notes

This report also appears in the first volume of the Bulletin of IGPL.

Acknowledgements

This work was supported by the Esprit project MEDLAR, by the BMFT project LOGO (ITS 9102) and by the project D2 of the 'Sonderforschungsbereich 314' of the German Science Foundation (DFG).

I would like thank all my colleagues in the various projects who contributed to this work. In particular I am grateful to Luis Fariñas del Cerro, Dov Gabbay, Andreas Herzig, Andreas Nonnengart, and Renate Schmidt for many important discussions and valuable contributions.

Abstract

This paper gives an overview on translation methods we have developed for nonclassical logics, in particular for modal logics. Optimized 'functional' and semi-functional translation into predicate logic is described. Using normal modal logic as an intermediate logic, other logics can be translated into predicate logic as well. As an example, the translation of modal logic of graded modalities is sketched. In the second part of the paper it is shown how to translate Hilbert axioms into properties of the semantic structure and vice versa, i.e. we can automate important parts of correspondence theory. The exact formalisms and the soundness and completeness proofs can be found in the original papers.

Contents

1 Introduction

Most inference system for nonclassical logics are defined in the style of Gentzen, sequent or tableaux calculi. Alternatively one can translate the theorems to be proved into predicate logic and then use standard predicate logic inference systems. In particular theorem provers and logic programming can be applied directly to the translated formulae. Furthermore it turned out that the extension of the translation methods to quantified versions of nonclassical logic is straightforward. Compared to classical methods where it is very difficult to use unification instead of exhaustive instantiation, translation allows the application of unification and resolution, which improves the performance of the systems considerably.

The general setting for this approach is the following: The logic we want to develop a translator for has to be presented by means of a possible worlds semantics. This means there is a possible worlds structure with certain properties and there are semantics definitions for the connectives in terms of this possible worlds structure. The properties of the possible worlds structure can either be given directly or they can be specified implicitly with Hilbert axioms. The typical example is Kripke semantics for modal logic. The possible worlds structure consists of a set of worlds and binary accessibility relations for each modal operator (in a multi–modal version).

There are two main problems to be solved. The first problem is to develop a translator which produces optimal 'code' in the sense that the translated formulae can be processed efficiently. Since the translation depends on the semantics of the logic, this amounts to figuring out alternative and more compact presentations of the semantics. The second problem is to find the axiomatization of the semantic structure in case it is only implicitly specified via Hilbert axioms. For example the Hilbert axiom $\Box P \Rightarrow P$ corresponds to the reflexivity of the accessibility relation. Finding these correspondences is very important for developing translators for different logics. In the second part of the paper we shall see how this can be done automatically.

2 Translation of Formulae

The possible worlds semantics of a particular connective can be turned into a translation rule. For example the semantics of a modal operators

$$w \models \Box P \text{ iff for all } v\ \mathcal{R}(w,v) \text{ implies } v \models P$$
$$w \models \Diamond P \text{ iff there is a } v \text{ with } \mathcal{R}(w,v) \text{ and } v \models P$$

can be turned into the translation rule

$$\pi_r(\Box P, w) = \forall v\ \mathcal{R}(w,v) \Rightarrow \pi_r(P,v)$$
$$\pi_r(\Diamond P, w) = \exists v\ \mathcal{R}(w,v) \wedge \pi_r(P,v)$$

A formula $\Box\Diamond P$ is then translated into $\forall u\ \mathcal{R}(0,u) \Rightarrow \exists v\ \mathcal{R}(u,v) \wedge P'(v)$ where 0 is the initial world[1] and the one-place predicate P' with a 'world term' as argument corresponds to the propositional variable P. Another example for a translation rule derived from the semantics is the translation of the implication connective in relevance logic. From

$$w \models A \to B \text{ iff for all } u, v\ \mathcal{R}(w,u,v) \text{ implies if } u \models A \text{ then } v \models B$$

we obtain

$$\pi_r(A \to B, w) = \forall u, v\ \mathcal{R}(w,u,v) \Rightarrow (\pi_r(A,u) \Rightarrow \pi_r(B,v)).$$

Since the structure of the π_r–translated formulae is not arbitrary, special constraint deduction methods could be developed for the case of translated modal logic formulae [Gen91, Sch93]. The $\mathcal{R}$–literals are treated as constraints and a constraint handling mechanism is derived from the theory that describes the frame property for the given modal system.

[1] Modal formulae are theorems if they hold in all worlds. If we negate the formula $\forall w$ ... in order to derive a refutation, the universal quantifier becomes an existential quantifier which Skolemizes to the world constant 0.

Taking the standard "relational" semantics for the connectives, however, yields a translation function which, due to the new $\mathcal{R}$–literals, destroys the the structure of the formulae. Moreover, the transformation into conjunctive normal form may duplicate these extra $\mathcal{R}$–literals exponentially often. For a normal theorem prover or logic programming system without a constraint handling mechanism, this introduces a lot of redundancy.

The main question is therefore: is it possible to transform the semantics of a logic such that the corresponding translation function yields predicate logic formulae which allow more efficient predicate logic inferencing? We shall present some alternative transformations.

2.1 Functional Semantics for Modal Logic

The standard relational Kripke semantics can be reformulated by decomposing the accessibility relation into a set AF of "accessibility functions." An accessibility function is a function mapping worlds to accessible worlds. For the serial case where there is always an accessible world this idea yields the following semantics for $\Box$.

$$w \models \Box P \text{ iff for all } \gamma \in AF\text{: } \gamma(w) \models P$$

which is turned into a translation rule translating into a many-sorted logic with a sort `AF` for each modality

$$\pi_f(\Box P, w) = \forall \gamma{:}\mathtt{AF}\ \pi_f(P, \gamma(w)).$$

This 'functional' translation has been investigated by various authors, in particular [Wal87a, Ohl88, JR88, Her89, AE92, Ohl90, Gas92, Ohl93]. The translated formulae are more compact than in the relational case. Moreover, the structure of the formulae is preserved which is useful for example for defining modal Horn clauses. Furthermore, reasoning about accessible worlds is done automatically by a unification algorithm, and not by explicit inference steps.

Unfortunately, things get more complex if the accessibility relation is not serial. Since the accessibility functions are partial in this case, the semantics definition and the translation has to be modified to deal with the undefined cases.

$$w \models \Box P \text{ iff for all } \gamma \in AF\text{: if } \gamma(w) \text{ is defined then } \gamma(w) \models P$$

We describe the corresponding functional translation for a multi modal logic with m modalities $[p_1], \ldots, [p_m]$. As target logic we take a standard order-sorted predicate logic. In particular, we use an instance of the logic of [Wal87b] and [SS89]. Its alphabet consists of three disjoint sets of symbols: the sort symbols, the function symbols and the predicate symbols. The set of sort symbols consists of `W`, denoting the set of worlds and $\mathtt{AF}_1, \ldots, \mathtt{AF}_m$, corresponding to the set of accessibility functions, one set AF_n for each modality p_n. The set of function symbols consists of two symbols: the constant symbol 0 of sort `W` (the initial world) and the symbol $\downarrow$ with declarations $\mathtt{AF}_n \times \mathtt{W} \mapsto \mathtt{W}$ for each n (the application function $\downarrow(\gamma, w) = \gamma(w)$.) We usually write $\gamma(w)$ instead of $\downarrow(\gamma, w)$. The set of predicate symbols consists of the equality symbol $=$, an arbitrary number of one place predicate symbols $p_1, p_2, \ldots$ with argument sort `W` and a special one place predicate symbol *def* also with argument sort `W`. $def(\gamma(w))$ means that $\gamma(w)$ is defined.

The translation function is now:

$$\begin{array}{lll}
\pi_f(p, x) & = p'(x) & \text{for } p \text{ a propositional symbol} \\
\pi_f(\neg\varphi, x) & = \neg\pi_f(\varphi, x) & \\
\pi_f(\varphi \wedge \psi, x) & = \pi_f(\varphi, x) \wedge \pi_f(\psi, x) & \\
\pi_f(\varphi \vee \psi, x) & = \pi_f(\varphi, x) \vee \pi_f(\psi, x) & \\
\pi_f(\langle p_n \rangle \varphi, x) & = \exists \gamma{:}\mathtt{AF}_n\ def(\gamma(x)) \wedge \pi_f(\varphi, \gamma(x)) & \\
\pi_f([p_n]\varphi, x) & = \forall \gamma{:}\mathtt{AF}_n\ def(\gamma(x)) \Rightarrow \pi_f(\varphi, \gamma(x)). &
\end{array}$$

It can be shown that this translation *preserves satisfiability*. A modal formula has a model if and only if the translated formula has a first–order predicate logic model. This is sufficient for doing *refutational theorem proving*.

2.2 Optimizations of the Functional Translation

Although the sets AF_n are really sets of functions, the predicate logic interpretations of the sorts $\mathtt{AF}_n$ need not be functions at all. In the soundness proof for the translation, a modal logic model is transformed into a predicate logic model where the sorts $\mathtt{AF}_n$ are indeed interpreted as sets of accessibility functions. In the completeness proof, however, we have to go the other way round and construct from a predicate logic model, where the sorts $\mathtt{AF}_n$ are interpreted as some set of objects, a modal logic model with the sets AF_n of accessibility functions. The relation between the interpretation $\|w\|$ of a term w and the interpretation $\|\downarrow(f,w)\|$ of the term $\downarrow(f,w)$ is sufficient to define a function γ_f with $\gamma_f(\|w\|) = \|\downarrow(f,w)\|$. Thus, a set AF_n of functions can always be reconstructed from the sets $\|\mathtt{W}\|, \|\mathtt{AF}_n\|$, the interpretation of the $\downarrow$–function and the *def*–predicate.

That means no special properties of $\|\mathtt{W}\|, \|\mathtt{AF}_n\|$ and the interpretation of $\downarrow$ are needed, but if $\|\mathtt{AF}_n\|$ is really a set of functions and $\downarrow$ denotes the application function, this has no influence on the completeness proof. This observation gives us a handle for restricting the set of predicate logic models by interpreting the sorts $\mathtt{AF}_n$ more 'function like'. In other words, without loosing soundness and completeness of the translation, certain additional axioms can be assumed which describe characteristic properties of sets of functions and which can then be exploited to simplify the proofs of the translated formulae.

First of all, functions can be composed with other functions yielding new functions. For example if γ and δ are functions then $\gamma \circ \delta$ is their composition. Unfortunately, if γ and δ are interpreted as functions mapping worlds to $\mathcal{R}$–accessible worlds then $\gamma \circ \delta$ maps worlds no longer necessarily to $\mathcal{R}$–accessible worlds, but to worlds accessible in two steps. Only if the accessibility relation is transitive, there is no difference. Syntactically this means, we cannot introduce a composition function symbol $\circ$ with sort declarations $\circ\colon \mathtt{AF}_n \times \mathtt{AF}_n \to \mathtt{AF}_n$. Instead of this, a sort $\mathtt{AF}^*$ has to be introduced for representing arbitrary compositions of elements of $\|\mathtt{AF}_n\|$. That means in particular, $\|\mathtt{AF}_n\| \subseteq \|\mathtt{AF}^*\|$, or as sort declaration $\mathtt{AF}_n \sqsubseteq \mathtt{AF}^*$ for every n can be assumed. The sort declaration for the composition function symbol is now $\circ\colon \mathtt{AF}^* \times \mathtt{AF}^* \to \mathtt{AF}^*$ and the axioms

$$\forall\gamma{:}\mathtt{AF}_n\ \forall\delta{:}\mathtt{AF}_k\ \forall w{:}\mathtt{W}\ \downarrow(\gamma,\downarrow(\delta,w)) = \downarrow(\delta\circ\gamma,w)$$

connect $\downarrow$ with $\circ$. Of course, $\circ$ can be assumed associative. Furthermore the sort declaration of the $\downarrow$–function can be generalized to $\downarrow{:}\mathtt{AF}^* \times \mathtt{W} \to \mathtt{W}$.

These considerations licences the usage of the so called 'world path' syntax. Instead of complex nested terms like $\downarrow(\gamma_k,\downarrow(\gamma_{k-1},\downarrow(\ldots,\ldots\downarrow(\gamma_1,0)\ldots)$ we simply write $\downarrow(\gamma_1\circ\ldots\circ\gamma_k,0)$ or even simpler $[\gamma_1\ldots\gamma_k]$. This is possible because $\downarrow(\ldots,0)$ is the only pattern occurring in the translation. For example a formula $[p]\langle q\rangle P$ would then be translated into $\forall\gamma{:}\mathtt{AF}_p\ \exists\delta{:}\mathtt{AF}_q\ P([\gamma\ \delta])$ (seriality assumed). It turned out that theory unification algorithms derived for modal systems like S4 work best on this simple string notation.

A terminating, sound and complete theory unification algorithm for the modal systems with reflexive, symmetric and transitive accessibility relations that operates on the world path syntax has been given in [Ohl88].

The algorithm is presented in a Martelli–Montanari style as a number of transformation rules for sets of equations.

$f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)$	$\to\ s_1 = t_1\ \&\ldots\& s_n = t_n$	Decomposition
$[s\ \mathbf{s}] = [t\ \mathbf{t}]$	$\to\ s = t\ \&\ \mathbf{s} = \mathbf{t}$	Separation
$[\mathbf{s}\ w\ \mathbf{s}'] = t$	$\to\ w = []\ \&\ [\mathbf{s}\ \mathbf{s}'] = t$	Identity
$[\mathbf{s}\ s\ w\ \mathbf{s}'] = t$	$\to\ w = s^{-1}\ \&\ [\mathbf{s}\ \mathbf{s}'] = t$	Inverse
$[w\ \mathbf{s}] = [\mathbf{t}\mathbf{t}']$	$\to w = \mathbf{t}\ \&\ \mathbf{s} = \mathbf{t}'$	Path-Separation
$[w\ s\ \mathbf{s}] = [\mathbf{t}\ t\ v\ \mathbf{t}']$	$\to\ v = [v_1\ v_2]\ \&\ w = [\mathbf{t}\ t\ v_1]\ \&[s\ \mathbf{s}] = [v_2\ \mathbf{t}']$	Splitting

Letters in bold face stand for sequences of terms whereas normal letters denote single terms. The Decomposition and Separation rules are always necessary. The Identity rule is necessary when the accessibility relation is reflexive. The Inverse rule covers the symmetry case and the Path Separation and Splitting rule treat the transitivity case. The Splitting rule terminates because a special syntactic invariant can be guaranteed for functionally translated formulae: Each occurrence

of a variable in a world path has the same prefix ('prefix stability' or 'unique path property'). Unification problems like $[xa] = [ax]$ which usually cause problems do therefore not occur.

The next optimization we present simplifies reasoning with the *def*–predicate. If there is an accessible world v from a world w, i.e. there is an accessibility function γ with $\gamma(w) = v$ there is no reason for the other accessibility functions to be undefined on w. They can map w at least to the world v which is accessible form w anyway. Thus a 'maximal defined-ness condition' can be assumed:

$$\forall w{:}\mathtt{W}\ (\exists\gamma{:}\mathtt{AF}_n\ \mathit{def}(\gamma(w))) \Rightarrow \forall\gamma{:}\mathtt{AF}_n\ \mathit{def}(\gamma(w))$$

As a consequence, $\mathit{def}(f(w))$ is redundant information for any f because from $\mathit{def}(f(w))$ we can immediately derive $\mathit{def}(\gamma(w))$ for a variable γ of the same sort as f. That means, only w and the sort of f matters. Therefore we replace *def* with a new predicate $\mathit{cont}(w, n)$ (meaning the accessibility relation $\mathcal{R}_n$ continues from world w). The correlation is $\mathit{cont}(w, n)$ iff $\exists\gamma{:}\mathtt{AF}_n\ \mathit{def}(\gamma(w))$. The translation function can now be optimized by moving the *cont*–predicate out of the scope of the quantifier in

$$\pi_f([p_n]\varphi, x) = \mathit{cont}(x, n) \Rightarrow \forall\gamma{:}\mathtt{AF}_n\ \pi_f(\varphi, \gamma(x))$$
$$\pi_f(\langle p_n\rangle\varphi, x) = \mathit{cont}(x, n) \wedge \exists\gamma{:}\mathtt{AF}_n\ \pi_f(\varphi, \gamma(x))$$

This simplifies reasoning in the cases where literals $\mathit{def}(f(s))$ and $\neg\mathit{def}(g(t))$ are not resolvable directly because although s and t might be unifiable, f and g clash in the unification algorithm. Without the above extra axiom, it may require complicated reasoning to show that these two literals are in fact complementary. On the other hand, the corresponding *cont*–literals $\mathit{cont}(s, n)$ and $\neg\mathit{cont}(t, n)$ produced by the optimized translation are directly resolvable.

In the third optimization we exploit that the accessibility functions can be assumed to be strict, i.e. the result of an application to something undefined is again undefined:

$$\forall w{:}\mathtt{W}\ \forall\gamma{:}\mathtt{AF}^*\ \neg\mathit{def}(w) \Rightarrow \neg\mathit{def}(\gamma(w))$$

or in terms of the *cont*–predicate and in world path notation:

$$\forall x{:}\mathtt{AF}^*\ \forall\gamma{:}\mathtt{AF}^*\ \neg\mathit{cont}([x], n) \Rightarrow \neg\mathit{cont}([x\gamma], n)$$

This axiom can be turned into a special unification rule for the *cont*–literals in world–path notation which simply ignores trailing parts of the strings if the start strings are unifiable.

For example the translation of $[n][n]P$ yields

$$\mathit{cont}([], n) \Rightarrow \forall\gamma{:}\mathtt{AF}_n \mathit{cont}([\gamma], n) \Rightarrow \forall\delta{:}\mathtt{AF}_n\ P([\gamma\delta]).$$

The clause form is $\neg\mathit{cont}([], n) \vee \neg\mathit{cont}([\gamma], n) \vee P([\gamma\delta])$. By applying this special axiom or the corresponding theory unification, we can factorize the clause and eliminate the first literal completely.

As already mentioned, the *def*–predicate becomes superfluous if *all* accessibility relations are serial. This is of course the maximal optimization we can achieve for treating this predicate.

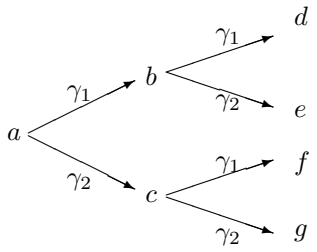
In [Ohl93, OS93] we present two quantifier exchange rules which can be applied to translated formulae.

Rule 1: $\forall\gamma_1 \ldots \forall\gamma_n\ \exists\gamma{:}\mathtt{AF}_n\ \Psi([w\gamma_1 a_1\gamma_2 a_2 \ldots a_{i-1}\gamma_i a_i\gamma]) \rightarrow$
$\exists\gamma\ \forall\gamma_1 \ldots \forall\gamma_i\ \Psi([w\gamma_1 a_1\gamma_2 a_2 \ldots a_{i-1}\gamma_i a_i\gamma])$

w and the γ_i may be of sort $\mathtt{AF}_j$ for arbitrary j or of sort $\mathtt{AF}^*$. '$\Psi([w\gamma_1 a_1\gamma_2 a_2 \ldots a_{i-1}\gamma_i a_i\gamma])$' means that all occurrences of γ are prefixed by the same string $w\gamma_1 a_1\gamma_2 a_2 \ldots a_{i-1}\gamma_i a_i$ where the a_j are, possibly empty, strings of Skolem constants.

Rule 2: $\forall\gamma{:}\mathtt{AF}_n\ \exists\delta\ \Psi([w\delta\gamma]) \rightarrow \exists\delta\ \forall\gamma\ \Psi([w\delta\gamma])$

These rules remove certain dependencies in Skolem functions generated from $\Diamond$–operators. For example $\Box\Diamond P$ can be translated into $\forall\gamma\ \exists\delta P([\gamma\delta])$ (seriality assumed) and then into $\exists\delta\ \forall\gamma\ P([\gamma\delta])$ which avoids that δ depends on γ. The rationality behind this quantifier exchange can be illustrated with the following example.



This little frame consists of the worlds a, b, c, d, e, f. The function set $\{\gamma_1, \gamma_2\}$ specifies the accessibility relation. There is however another function set containing two more functions γ_3 mapping b to d and c to g, and γ_4 mapping b to e and c to f, which specifies exactly the same frame. The difference is that the extended function set is rich enough to ensure that whenever $\forall\gamma\ \exists\delta\ P(\delta(\gamma(a)))$ is valid in that frame then $\exists\delta\ \forall\gamma\ P(\delta(\gamma(a)))$ is also valid.

In tree frames there is always a function set which is rich enough to allow exchange of quantifier in this way. Frames which are not trees can be unreeled to tree frames by copying worlds. This has, however, subtle consequences which we cannot discuss here (see [OS93]).

Using the quantifier exchange rules, various propositional modal systems can be translated into a fragment of predicate logic *without* function symbols, i.e. into a *decidable* fragment of predicate logic. The translated formulae themselves do no longer contain Skolem functions depending on variables. Only the axioms describing the characteristic properties of the frames in the particular modal system may contain function symbols. For those propositional modal systems where an axiomatization of the frame properties without function symbols and without equations causing lengthening of the world paths is possible, decidability is obvious.

The quantifier exchange rules turned out to be quite powerful. For example they bring systems like the McKinsey system $\Box\Diamond P \Rightarrow \Diamond\Box P$ whose corresponding property of the accessibility relation is not first–order axiomatizable into the domain of the functional translation.

2.3 Semi–Functional Translation

In the functional version of the semantics, properties of the accessibility relation are in general expressed as equations. For example the reflexivity of $\mathcal{R}$ corresponds to the equation $\forall w\ \gamma(w) = w$ where γ is an identity function. In order to realize reasoning with the functionally translated formulae efficiently, these equations have to be turned into theory unification algorithms. If this is not possible, complex equational reasoning may become necessary. A way to avoid equational reasoning while retaining the advantages of the functional translation has been developed by Andreas Nonnengart ([Non93]).

The idea of this mixed translation in the *serial* case is to translate the $\Box$–operator relationally and the $\Diamond$–operator functionally.

$$\begin{aligned}\pi_m(\langle p_n\rangle\varphi, x) &= \exists\gamma{:}\mathtt{AF}_n\ \pi_m(\varphi, \gamma(x))\\ \pi_m([p_n]\varphi, x) &= \forall y{:}\mathtt{W}\ \mathcal{R}_n(x, y) \Rightarrow \pi_m(\varphi, y)\end{aligned}$$

Since the $\Box$ and $\Diamond$–operators are now no longer dual in the usual way, we get extra conditions which enforce the duality $\Box P \Leftrightarrow \neg\Diamond\neg P$. The two extra conditions are:

$$\begin{aligned}&\forall w\ \forall\gamma\ \mathcal{R}(w, \gamma(w))\\ &\forall u, v\ \mathcal{R}(u, v) \Rightarrow \exists\gamma\ v = \gamma(w)\end{aligned}$$

For each Hilbert axiom of the particular modal system we obtain now a corresponding semantic property formulated in the mixed language with accessibility relation and accessibility functions. For example $\forall w, \gamma_1, \gamma_2\ \mathcal{R}(w, \gamma_2(\gamma_1(w)))$ corresponds to $\Box P \Rightarrow \Box\Box P$ and $\forall w, \gamma_1, \gamma_2\ \mathcal{R}(\gamma_1(w), \gamma_2(w))$ corresponds to $\Diamond P \Rightarrow \Box\Diamond P$.

Conditional equational completion [Gan91] with the two basic axioms above and the specific axioms for the modal systems eliminates the equation $v = \gamma(w)$ in most cases and yields quite compact set of equation free theory clauses. For example for the system KD45 ($\Box P \Rightarrow P$, $\Box P \Rightarrow \Box\Box P$, $\Diamond P \Rightarrow \Box\Diamond P$), the whole set collapses to a single unit clause $\forall u, v{:}W\,\forall\gamma{:}\mathtt{AF}\ \mathcal{R}(u, \gamma(v))$. See [Non93] for more examples.

Thus, the semi–functional translation yields a still quite compact representation of the translated formulae while also the properties of the accessibility relation can be expressed in a very short and compact way, but *without equations*. In many cases the resulting theory clauses are

Horn clauses. That means from the point of view of the theory clauses, this kind of translation is very suitable for a translation into standard Prolog.

3 Translation into Normal Modal Systems

Since the methods for translating modal logic into predicate logic are quite well developed, the next step is to use the normal modal logic as an intermediate language for translating other logics into predicate logic. For example it is well known that intuitionistic logic can be translated into modal S4. That means via S4 we can translate intuitionistic logic functionally into predicate logic. We have started to investigate the possibility for translating other logics into modal logic. One example is given in the next section. Other examples can be found in [BH93] and [GH93].

3.1 Graded Modalities

Modal logics of graded modalities have operators M_n and L_n. $M_n\varphi$ is interpreted as φ holds in more than n worlds and $L_n\varphi$ is interpreted as $\neg\varphi$ holds in at most n worlds [vdH92]. According to Fattorosi–Barnaba and de Caro [FBdC85], the following axioms and inference rule make up a complete Hilbert calculus:

the axioms of propositional logic
$M_{n+1}\Phi \Rightarrow M_n\Phi$
$L_0(\Phi \Rightarrow \Psi) \Rightarrow (M_n\Phi \Rightarrow M_n\Psi)$
$L_0\neg(\Phi \wedge \Psi) \Rightarrow ((M!_n\Phi \wedge M!_m\Psi) \Rightarrow M!_{n+m}(\Phi \vee \Psi))$
$\vdash \Phi$ and $\vdash \Phi \Rightarrow \Psi$ implies $\vdash \Psi$
$\vdash \Phi$ implies $\vdash L_0\Phi$

where $M!_n$ is the 'exactly n'–operator. The problem with this formulation is that the known tableaux calculi for this kind of logic must generate $n+1$ Skolem constants whenever they hit a formula $M_n\phi$ [HB91]. For example a formula $city \Leftrightarrow town \wedge M_{100000}citizen$ defining a city as a town with more than 100000 citizens would trigger the generation of 100001 Skolem constants for the citizens. Alternatively one could translate such formulae directly into predicate logic. A statement 'at most n', however, would then have to be translated into $O(n^2)$ equations which triggers so many case distinctions. In none of these approaches there is a place to apply simple arithmetic.

In [HOS93] we show how this logic can be translated into a normal multi–modal system and then into predicate logic such that arithmetic can be applied. The idea is to add an extra class of worlds representing sets of normal worlds. $M_n\varphi$ is then translated into $\langle n\rangle\Box\varphi$ which intuitively means: there is an $\mathcal{R}_n$–accessible world (which stands for a set of $\geq n+1$ normal worlds) and in all the $\mathcal{R}$-accessible worlds (which are just these $\geq n+1$ worlds) φ holds. Besides the standard axioms for normal modal systems, the corresponding Hilbert axioms of this system are:

$\vdash [0]\Diamond\Phi \Rightarrow [n]\Box\Phi$
if $\vdash \Phi$ then $\vdash [n]\Diamond\Phi$
$\vdash [n]\Phi \Rightarrow [n+1]\Phi$
$\vdash \langle n+m\rangle\Box(\Phi \vee \psi) \Rightarrow (\langle n\rangle\Box\Phi \vee \langle m\rangle\Box\psi)$
$\vdash (\langle n\rangle\Box(\Phi \wedge \psi) \wedge \langle m\rangle\Box(\Phi \wedge \neg\psi)) \Rightarrow \langle n+m+1\rangle\Box\Phi$

This is not the direct translation of the original system, and only the completeness proof reveals the role of the axioms. In fact it is somewhat more general than the original system. In particular formulae like $[n]P$ make sense in this system by interpreting P as a predicate on *sets of objects*. For example $[10](\Box soccer\text{-}player \Rightarrow soccer\text{-}team)$ expresses 'for every set with more than 10 objects: if all elements are soccer players then this set makes up a soccer team'.

Formulae in this logic can now be functionally translated in the usual way. Additionally, in order to characterize this particular system, the Hilbert axioms translate into seven theory clauses. They were computed using the quantifier elimination algorithm (see below).

P_1: $\mathtt{AF}_m \sqsubseteq \mathtt{AF}_n$ if $n \leq m$.
P_2: $\neg cont(\kappa, n) \vee [\kappa f_0^{1n}(x_n, z)y] \approx [\kappa x_n z]$.
P_3: $\neg cont(\kappa, k) \vee [\kappa x_k h^{2nmk}(\kappa, y, z)] \approx [\kappa f_n^{2nmk}(\kappa, x_k, z)y]$ $\qquad k \geq n+m$.
P_4: $\neg cont(\kappa, k) \vee [\kappa x_k h^{2nmk}(\kappa, y, z)] \approx [\kappa g_m^{2nmk}(\kappa, x_k, y)z]$.

P_5: $\neg cont(\kappa, n) \vee \neg cont(\kappa, m) \vee [\kappa x_n f^{3nm}(\kappa, y_m, z)] \approx [\kappa y_m g^{3nm}(x_n, z)] \vee$
$\neg cont(\kappa, n+m+1)$.
P_6: $\neg cont(\kappa, n) \vee \neg cont(\kappa, m) \vee [\kappa x_n f^{3nm}(\kappa, y_m, z)] \approx [\kappa y_m g^{3nm}(\kappa, x_n, z)] \vee$
$[\kappa x_n f^{3nm}(\kappa, y_m, z)] \approx [\kappa h^{3nm}_{n+m+1}(\kappa, x_n, y_m) z] \vee$
$[\kappa y_m g^{3nm}(\kappa, x_n, z)] \approx [\kappa h^{3nm}_{n+m+1}(\kappa, x_n, y_m) z])$.
P_7: $\neg cont(\kappa, n) \vee \neg cont(\kappa, m)$ if $n < m$

κ is of sort $\mathtt{AF}^*$ and stands for an arbitrary start sequence in a world path. The superscripts $1, 2, 3$ of f, g, h just distinguish different f's, g's and h's. Actually these formulae are schemata for clauses. The symbols k, n, m have to be instantiated with all non-negative integers. The subscript $_n$ of the variables and Skolem functions denotes sorts $\mathtt{AF}_n$ of the terms. The $\approx$–relation is almost like equality, but lacking the substitutivity property for functions. (This has to do with the application of the quantifier exchange rules mentioned above. See [OS93] for the details.) Since the set of formulae stands for an infinite number of clauses, they can only be put to work in a theory or constraint resolution framework. This is the place where arithmetic can be incorporated.

As there is a close correspondence between the M_n–operator and the $atmost(n, r, \varphi)$ construct in the knowledge representation language KL-ONE, this translation of the logic of graded modalities is of particular interest for knowledge representation systems.

The following example shows how the reasoning with the translated clauses work.

Graded Modalities	KL-ONE Formulation
$M_0 M_3$ *true*	*at-least*$(1, R,$ *at-least*$(4, R, \top))$
$M_0 L_3$ *false*	*at-least*$(1, R,$ *at-most*$(3, R, \top))$
L_1 *false*	*at-most*$(1, R, \top)$

Translation

Graded Modalities	Modal Logic	Predicate Logic
$M_0 M_3$ *true*	$\langle 0\rangle\Box\langle 3\rangle\Box$ *true*	$cont([], 0) \wedge cont([a_0 x], 3)$
$M_0 L_3$ *false*	$\langle 0\rangle\Box[3]\Diamond$ *false*	$cont([], 0) \wedge \neg cont([b_0 y], 3)$
L_1 *false*	$[1]\Diamond$ *false*	$\neg cont([], 1)$

$n = 0, m = 0, \kappa = []$ instance of the clause P_5:

$$\neg cont([], 0), [x_0 f^{300}([], y_0, z)] = [y_0 g^{300}([], x_0, z)], cont([], 1).$$

Theory resolution with unifier $\{x_0 \mapsto a_0, y_0 \mapsto b_0, x \mapsto f^{300}([], b_0, z), y \mapsto g^{300}([], a_0, z)\}$ yields the empty clause. Notice that the reasoning is on the abstract level without generating instances of the sets explicitly.

4 Automating Correspondence Theory

Correspondence theory relates properties of the accessibility relations with Hilbert axioms. For example $\Box P \Rightarrow \Box\Box P$ corresponds to the transitivity of the accessibility relation. Since our translation methods need an explicit axiomatization of the properties of the possible worlds structure, we have to compute these properties somehow in case they are specified only implicitly as Hilbert axioms. If this can be done automatically, we no longer rely on the well investigated logics found in the literature, but we can develop applications where the user can specify his own logic in an abstract Hilbert style and this is then automatically translated into executable code.

In order to see what this means, let us rewrite the Hilbert axiom $\Box P \Rightarrow \Box\Box P$ which is implicitly assumed to hold for all P and in all worlds into predicate logic, using the standard relational semantics of $\Box$. The translation yields

$$\forall P' \; \forall u \; (\forall v \; \mathcal{R}(u, v) \Rightarrow P'(v)) \Rightarrow \forall v \; \mathcal{R}(u, v) \Rightarrow \forall w \; \mathcal{R}(v, w) \Rightarrow P'(w)$$

This is a second–order predicate logic formula. The key observation for computing correspondences automatically was that such second–order predicate logic formulae are sometimes equivalent to

first–order formulae without the P'. If there is in fact a first–order equivalent, then this is precisely a representation for the desired correspondence property. Finding first–order equivalents means eliminating second–order quantifiers.

4.1 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 $\forall x\ x = x$. 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). 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 and output the result. If Skolemization cannot be undone, the only chance is to take parallel (second–order) Henkin quantifiers [Hen61] or to leave the Skolem functions existentially quantified. In this case the resulting formula is again second–order.

$\lhd$

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:

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

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.

4.2 A Framework for Automating Correspondence Theory

In this section we show how to make use of the SCAN algorithm for translating Hilbert axioms into semantic properties. The method is applicable not only for modal logic, but for all logics with semantics definitions for its connectives which can be axiomatized in a first–order framework. In particular for the case of modal logic we can apply it to compute the frame properties also for the functional translation. In order to give a complete picture, we consider also the inverse direction, from the frame properties to the Hilbert axioms.

Developing correspondences as for example between
(i) $\forall P\ \Box P \Rightarrow \Box\Box P$ and (ii) the underlying accessibility relation is transitive.
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).

The bottom–up direction is not strictly relevant to the translation topic. During the development of a new logic, it is, however, very instructive if not only a semantics, but also a Hilbert system is known. Whereas notions like ‘belief’ or ‘knows’ are usually primarily specified by Hilbert

axioms, just the opposite is the case for well known mathematical structures such as for example linear orderings as semantics in a temporal logic. Here the property of the semantical structure is given and the corresponding Hilbert axiom is to be computed.

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 matter 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.

4.3 The Top–Down Direction

The top-down direction of the correspondence problem can be stated as follows: What needs to be given is first of all

1. some operators F whose semantics is defined in terms of other relations and functions R_i using the 'holds'–predicate H:

   (1) $Def(F, R_i) = \forall X_1, \ldots, X_n\ \forall x\ H(F(X_1, \ldots, X_n), x) \Leftrightarrow \Phi$.
   where Φ contains *no* occurrence of F.

   The 'holds' predicate is used to formulate formulae as predicate logic terms and to present the problem as a pure first–order predicate logic problem that can be submitted to a standard theorem prover. For example, $Def(F, R_i)$ could be defined as

   (2) $\forall X_1, \ldots, X_n\ \forall x\ H(F(X_1, \ldots, X_n), x) \Leftrightarrow$
   $\forall x_1, \ldots, x_n\ \mathcal{R}(x, x_1, \ldots, x_n) \Rightarrow H(X_1, x_1) \Rightarrow \ \ldots\ \Rightarrow H(X_n, x_n)$

   For $n = 1$ this is the definition of the modal $\Box$–operator. For $n = 2$ this is the definition of the relevance logic implication.

2. The second part of the problem specification is the Hilbert axiom $\Psi(F)$ which is to be translated. This should again be formulated in first–order predicate logic using again the special 'holds'–predicate. For example the first–order formulation of $\Box P \Rightarrow \Box\Box P$ is

   (3) $\forall P\ \forall w\ H(implies(\Box(P), \Box(\Box(P))), w)$.

   The structure of the formulae $\Psi(F)$ must be such that application of $Def(F, R_i)$ for all F as rewrite rule from left to right (with suitable renamings of bound variables) eliminates the F completely and the resulting formula is of the structure

   (4) $\Psi' = QX_1, \ldots, X_n\ \Psi''(H(X_1, \ldots), \ldots, H(X_n, \ldots))$ or equivalently

   (5) $\Psi' = QX'_1, \ldots, X'_n\ \Psi''(X'_1(\ldots), \ldots, X'_n(\ldots))$

   where Q is an existential or a universal quantifier. In the version (5), the variables X_i have been replaced with one-place predicate symbols X'_i. This brings to light the second–order nature of the problem which had been hidden in the holds predicate. For example rewriting (3) with (2, $n = 1$) and a corresponding rule for *implies* yields

   (6) $\forall P'\ \forall w\ (\forall x\ \mathcal{R}(w, x) \Rightarrow P'(x)) \Rightarrow \forall u\ \mathcal{R}(w, u) \Rightarrow \forall v\ \mathcal{R}(u, v) \Rightarrow P'(v)$

Our goal is to find a formula $\Gamma(R_i)$ such that

(7) $Def(F, R_i) \Rightarrow (\Psi(F) \Leftrightarrow \Gamma(R_i))$.

Since Def is an equivalence, rewriting $\Psi(F)$ to $\Psi'(R_i)$ is an equivalence transformation in the theory of Def, i.e. $Def(F, R_i) \Rightarrow \Psi(F) \Leftrightarrow \Psi'(R_i)$. Thus, computing a correspondence property amounts to computing the formula $\Gamma(R_i)$ with $QX'_1, \ldots, X'_n\ \Psi'(R_i) \Leftrightarrow \Gamma(R_i)$. 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.

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

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

We illustrate the procedure with the following examples:

Example 4.2 (Modal K4-Axiom)
Relational Translation:

Hilbert axiom:	$\Box P \Rightarrow \Box\Box P$	
H-Formulation:	$\forall P\ \forall w\ H(implies(\Box P, \Box\Box P), w)$	$(= \Psi(\Box, implies))$
Semantics:	$\forall P\ \forall w\ H(\Box P, w) \Leftrightarrow (\forall v\ \mathcal{R}(w, v) \Rightarrow H(P, v))$	$(= \mathrm{Def}(\Box,\mathcal{R}))$
	Semantics of *implies* as expected	
translated (i.e. Semantics applied as rewrite rule).		
	$\forall P'\ \forall w\ (\forall x\ \mathcal{R}(w, x) \Rightarrow P'(x)) \Rightarrow \forall u\ \mathcal{R}(w, u) \Rightarrow \forall v\ \mathcal{R}(u, v) \Rightarrow P'(v)$	
negated:	$\exists P'\ \exists w\ (\forall x\ \mathcal{R}(w, x) \Rightarrow P'(x)) \wedge \exists u\ \mathcal{R}(w, u) \wedge \exists v\ \mathcal{R}(u, v) \wedge \neg P'(v)$	
clause form:[2]	$\neg\mathcal{R}(\mathrm{w}, x) \vee P'(x)$	
	$\mathcal{R}(\mathrm{w}, \mathrm{u})$	
	$\mathcal{R}(\mathrm{w}, \mathrm{v})$	
	$\neg P'(\mathrm{v})$	
P' resolved away:	$\neg\mathcal{R}(\mathrm{w}, \mathrm{v})$	
	$\mathcal{R}(\mathrm{w}, \mathrm{u})$	
	$\mathcal{R}(\mathrm{u}, \mathrm{v})$	
unskolemized:	$\exists w, u, v\ \neg\mathcal{R}(w, v) \wedge \mathcal{R}(w, u) \wedge \mathcal{R}(u, v)$	
negated:	$\forall w, u, v\ \mathcal{R}(w, u) \wedge \mathcal{R}(u, v) \Rightarrow \mathcal{R}(w, v)$	(transitivity)

Functional Translation (seriality assumed):
As already mentioned, the method is parametrized with the semantics of the connectives. Therefore it works as well for the functional semantics of the modal operators and we obtain the characteristic frame property in terms of the functinal translation.

Hilbert axiom:	$\Box P \Rightarrow \Box\Box P$
H-Formulation:	$\forall P\ \forall w\ H(\Box P \Rightarrow \Box\Box P, w)$
Semantics:	$\forall P\ \forall w\ H(\Box P, w) \Leftrightarrow \forall\gamma\ H(P, \gamma(w))$
translated (i.e. Semantics applied as rewrite rule):	
	$\forall P'\ \forall w\ (\forall\gamma\ P'(\gamma(w))) \Rightarrow \forall\delta\ \forall\iota\ P'(\iota(\delta(w)))$
negated:	$\exists P'\ \exists w\ (\forall\gamma\ P'(\gamma(w))) \wedge \exists\delta\ \exists\iota\ \neg P'(\iota(\delta(w)))$
clause form:	$P'(\gamma(\mathrm{w}))$
	$\neg P'(\iota(\delta(\mathrm{w})))$
P resolved away:	$\gamma(\mathrm{w}) \neq \iota(\delta(\mathrm{w}))$
unskolemized:	$\exists w\ \exists\delta, \iota\ \forall\gamma\ \gamma(w) \neq \iota(\delta(w))$
negated:	$\forall w\ \forall\delta, \iota\ \exists\gamma\ \gamma(w) = \iota(\delta(w))$

$\lhd$

4.4 The Bottom–Up Direction

In the bottom–up direction of the correspondence problem we want to compute from the property $\Gamma(R_i)$ of the symbols R_i and the definition $Def(F, R_i)$ for the operator F a corresponding property

[2] To distinguish between variables and Skolem constants, we write the Skolem constants Roman style.

$\Psi(F)$. This direction 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 [BGO93].

There are two different methods for guessing Ψ. In the first method we exploit that

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

implies

$$\forall R_i\ Def(F, R_i) \Rightarrow (\Gamma(R_i) \Rightarrow \Psi(F)).$$

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

In the second method for 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

$$\exists f\ \forall w\ H(f, w)$$

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 standard propositional connectives and, or, neg, impl, things are simpler. The axioms for these connectives are:

$$\begin{array}{lll}
\forall X, Y & \forall w\ H(and(X, Y), w) & \Leftrightarrow (H(X, w) \wedge H(Y, w)) \\
\forall X, Y & \forall w\ H(or(X, Y), w) & \Leftrightarrow (H(X, w) \vee H(Y, w)) \\
\forall X & \forall w\ H(neg(X), w) & \Leftrightarrow \neg H(X, w) \\
\forall X, Y & \forall w\ H(implies(X, Y), w) & \Leftrightarrow (H(X, w) \Rightarrow H(Y, w))
\end{array}$$

The input to the theorem prover consists of these axioms, together with $Def(F, R_i)$ and $\Gamma(R_i)$. The theorem to be proven is $\exists f\ \forall w\ H(f, w)$. The result are proofs with bindings for f, for example $f = implies(F(X), F(F(X)))$ (which of course stands for our standard example $\Box P \Rightarrow \Box\Box P$.)

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

1. Try quantifier elimination for $\exists R_i\ Def(R_i, F) \wedge \Gamma(R_i)$.
   If this does not succeed:
2. Try to find a solution in terms of propositional connectives.
   (a) Axiomatize the connectives.
   (b) From these axioms together with $Def(F, R_i)$ and $\Gamma(R_i)$ prove the theorem $\exists f\ \forall w\ H(f, w)$.
   (c) Each binding for f is a candidate for $\Psi(F)$ that needs to be verified with the top–down method.

Example 4.3 (For the Bottom-Up Direction)
Again we use the correspondence $\Box P \Rightarrow \Box\Box P$ with the transitivity of the accessibility relation to illustrate the bottom–up direction. The following is a protocol of the OTTER theorem prover [McC90]. It is the first of the generated proofs which actually uses the transitivity clause (this turned out to be a very powerful filter for eliminating junk proofs). The $\Box$–operator is encoded as the function `F` and the implication connective as the function `i`. Otter uses '|' for the disjunction symbol.

```
formula_list(usable).
(all w (all X (H(F(X),w) <-> (all v (R(w,v) -> H(X,v)))))).
(all w (all X (all Y (H(i(X,Y),w) <-> (H(X,w) -> H(Y,w)))))).
end_of_list.

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

Clauses:
1 -H(F(x1),w) | -R(w,v) | H(x1,v).
2  H(F(x1),w) | R(w,$f1(w,x1)).
3  H(F(x1),w) | -H(x1,$f1(w,x1)).
4 -H(i(x2,x3),w) | -H(x2,w) | H(x3,w).
5  H(i(x2,x3),w) | H(x2,w).
6  H(i(x2,x3),w) | -H(x3,w).
7 -R(x,y) | -R(y,z) | R(x,z).
8 -H(x4,$f2(x4)) | $ans(x4).

---------------- PROOF ---------------- (4.47 sec)
12 [hyper,8,5] $ans(i(x,y)) | H(x,$f2(i(x,y))).
17 [hyper,7,2,2] R(x,$f1($f1(x,y),z)) | H(F(y),x) | H(F(z),$f1(x,y)).
41 [hyper,17,3] R(x,$f1($f1(x,F(y)),y)) | H(F(F(y)),x).
56 [hyper,41,6] R(x,$f1($f1(x,F(y)),y)) | H(i(z,F(F(y))),x).
86 [hyper,56,8] R($f2(i(x,F(F(y)))),$f1($f1($f2(i(x,F(F(y)))),F(y)),y)) |
              $ans(i(x,F(F(y)))).
341 [hyper,86,1,12] $ans(i(F(x),F(F(y)))) | H(x,$f1($f1($f2(i(F(x),F(F(y)))),F(y)),y)).
349 [hyper,341,3] $ans(i(F(x),F(F(x)))) | H(F(x),$f1($f2(i(F(x),F(F(x)))),F(x))).
368 [hyper,349,3] $ans(i(F(x),F(F(x)))) | H(F(F(x)),$f2(i(F(x),F(F(x))))).
383 [hyper,368,6] $ans(i(F(x),F(F(x)))) | H(i(y,F(F(x))),$f2(i(F(x),F(F(x))))).
384 [binary,383,8] $ans(i(F(x),F(F(x)))).
```

In logical notation, the answer is $\Box P \Rightarrow \Box\Box P$. With the top-down method we have already verified that this is in fact the correct answer. $\lhd$

5 Summary

As long as the semantics of a logic can be formulated in first-order logic, it can be turned into a translation function into predicate logic. The actual presentation of the semantics, however, is not unique. For example a binary accessibility relation can be presented syntactically with a two-place function or with a sort decribing a set of 'accessibility' functions. As another example, a ternary relation as it is used in relevance logic, can be turned into three binary relations which in turn can be decomposed again into sets of accessibility functions. These transformation are the mechanisms that allow the translation to be tuned such that 'efficient predicate logic code' is produced.

Since the modal $\Box$–and $\Diamond$–operators are essentially universal and existential quantifiers in disguise, it seems that modal logic is the central logic in the sense that other logics can be expressed in modal logic. Therefore I propose a two–step process, first translate into a normal modal system and then use the functional or semi-functional translation into predicate logic. As an example we have shown this for modal logic with graded modalities.

To support the development of these translations, we have shown how to automate the computation of correspondences between Hilbert axioms and semantic properties. The method works in both directions and can be fully automated. A prototype implementation of the top–down part by Antonis Kotzamanidis is avaiblable. It uses the theorem prover Otter to realize the SCAN algorithm.

Although there is some progress in dealing with second–order semantic structures (cf. also [Her90]), this is the main limitation of this approach. In order to get a useful translation, eventually

everything must be massaged into first–order predicate logic. There are enough applications, however, in particular in the knowledge representation area, where this is guaranteed.

6 Open Problems and Future Work

The work presented in this overview can only be seen as single steps in the attempt to automate the development of application oriented nonclassical logics and to develop efficient generic reasoning systems.

There are still many problems on various levels of the approach to be solved. Let me mention just a few of them. First of all, there is not yet a detailed comparison between the efficiency of different translation methods for modal logics and between translation at all and other calculi which operate on the original syntax. Since this means finding suitable test examples, comparing different theorem provers, different search strategies, even implementing special unification algorithms for the functional translation, it would require a major effort.

One of the problems with the functional translation is that either equational reasoning or special theory unification algorithms are needed. Since equational reasoning is very inefficient, functional translation can compete only if the appropriate theory unification algorithems are implemented. At the time being, this still requires a human expert, which means that we cannot yet automate the development of such a translation system from an abstract specification of the logic. Narrowing might be the method to overcome this problem, but this has to be investigated.

A similar problem arises in the semi-functional translation. As we have seen, we get one conditioned equation from the translation of the duality formula $\Box P \Leftrightarrow \neg\Diamond\neg P$. Fortunately most of the formulae describing the correspondences between semantic structures and Hilbert axioms do not contain equations. Although we have not yet done a systematic investigation, we suceeded in finding alternative equation free representations for various modal systems using conditional equational completion (we used Harald Ganzinger's CEC program). This seems to be a promising route for generating optimized calculi automatically.

A spin off from this work might be methods for refutational complete transformations of standard predicate logic formulae to make life easier for the theorem prover. For example, each formula which could be seen as a relational translation of some modal formula can be transformed into the corresponding semi-functional or functional translation. But what to do with formulae which are almost, but not exactly relational translations? My impression is that there is a similar potential for improving the treatment of binary relations as the transition from unsorted to sorted logic improves the treatment of unary relations.

The approach we have developed for the logic of graded modalities might also turn out to be useful for the treatment of finite domains in predicate logic. The axiomatization of a finite domain D with n elements is

$$(\exists x_1, \ldots, x_n \; D(x_1) \wedge \ldots \wedge D(x_n) \wedge \forall y \; D(y) \Rightarrow (y = x_1 \vee \ldots \vee y = x_n)$$

Everybody who has ever tried to apply a predicate logic theorem prover to this kind of examples knows that the disjunction triggers a vast amount of cases distingtions. This is feasible only for small n.

Quantifier elimination is the key technique for computing semantic properties from Hilbert axioms. Unfortunately this is a problem without a complete solution. For example the McKinsey axiom $\forall P \; \Box\Diamond P \Rightarrow \Diamond\Box P$ alone corresponds to a second–order property of the accessibility relation (reversing skolemization in the SCAN algorithm needs second–order Henkin quantifiers). Combined with the transitivity axiom $\Box P \Rightarrow \Box\Box P$, however, these two define atomicity $\forall x \; \exists y \; (\mathcal{R}(x, y) \wedge \forall z \; \mathcal{R}(y, z) \Rightarrow z = y))$ [vB84, page203] which is obviously a first–order definable property.

Applied to the McKinsey axiom, SCAN actually computes this property if the critical clause which prevents reversing the skolemization in the normal way is replaced with its factor. Although we have some ideas, why transitivity might in this particular case enable this operation, we are far from having a general theory for processing combinations of axioms with these strange properties.

Actually the proof that the McKinsey axiom together with the transitivity axiom correspond to atomicity requires the axiom of choice. Therefore no simple solution of this problem is to be expected.

Another modal Hilbert axiom which corresponds to a second order property of the accessibility relation is Löb's axiom $\Box(\Box P \Rightarrow P) \Rightarrow \Box P$ axiomatizing the system G. It enforces that all chains in the possible worlds structure are finite. Applied to this axiom, SCAN loops. But it keeps on producing clauses with only $\mathcal{R}$–literals. The loop has a certain regular structure which is easy to recognize and which can be turned into a finite representation of an infinite formula. Automating loop detection, at least for limited cases, seems possible and may help in investigating more complex logics.

Applied to the axiom $\Diamond\Box P \vee \Box(\Box(\Box Q \Rightarrow Q) \Rightarrow Q)$, SCAN loops also. The difference to Löb's axiom is that in this case it cannot get rid of the predicate Q. Each resolvent still has literals with Q. And in fact, this axiom is known to be incomplete [HC84] in the sense that there is no frame class at all characterized by this axiom. It should be investigated whether this chracteristic behaviour of SCAN always indicates incompleteness of the Hilbert axiom.

The particular treatment of Hilbert systems we have shown in this paper relies on a basic completeness theorem for the semantics. For example in modal logic it is well known that the axiom K and the necessitation rule guarantee completness of the standard relational Kripke semantics. But what about Hilbert systems without such a basic completeness theorem? Again from modal logic it is known that minimal model semantics is complete provided closedness under equivalences ($\vdash P \Leftrightarrow Q$ implies $\vdash \Box P \Leftrightarrow \Box Q$) is guaranteed. In this semantics, $\Box P$ is valid in a world w if the truth set of P is a 'neighbourhood' of w. Relational semantics can be reconstructed if the neighbourhood structure is closed under intersection and supersets.

The game to play now is to find general schemas for very weak semantic structures, weak in the sense that as few Hilbert axioms as possible are tautologies in this semantics. Using this semantics, the Hilbert axioms are translated with quantifier elimination. The result can then be used to (automatically) prove certain key lemmas which licence the transition to a stronger semantics.

In [Gab93], Dov Gabbay has developed a general schema for a very weak algebraic semantics for a logic specified not with a Hilbert system, but with a consequence relation $\varphi \vdash \psi$ where φ and ψ are single formulae. This is a very promising starting point for developing a similar schema for arbitrary Hilbert systems.

Since logics can be specified in various ways, syntactically with various kinds of consequence relations as well as semantically, either with algebraic or other kinds of model theoretic semantics, the ultimate goal of this whole enterprise is to provide *automated* methods for transforming the specifications from any such framework into any other. At the time being this goal seems not to be completely unrealistic.

Acknowledgement

I would like thank all my colleagues in the various projects who contributed to this work. In particular I am grateful to Luis Fariñas del Cerro, Dov Gabbay, Andreas Herzig, Andreas Nonnengart, and Renate Schmidt for many important discussions and valuable contributions. The comments of various referees were also very helpful.

References

[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.

[AE92] Yves Auffray and Patrice Enjalbert. Modal theorem proving: An equational viewpoint. *Journal of Logic and Computation*, 2(3):247–297, 1992.

[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, 66123 Saarbrücken, 1993.

[BGW92] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Theorem proving for hierarchic first-order theories. In G. Levi and H. Kirchner, editors, *Algebraic and Logic Programming, Third International Conference*, pages 420–434. Springer-Verlag, LNCS 632, September 1992.

[BH93] Philippe Balbiani and Andreas Herzig. A translation from the modal logic of provability into K4. *Journal of Applied Non–Classical Logics*, 1993. to appear.

[FBdC85] M. Fattorosi-Barnaba and F. de Caro. Graded modalities. *Studia Logica*, 44:197–221, 1985.

[Gab93] Dov M. Gabbay. Classical vs non-classical logics. In J. Siekmann D. M. Gabbay, editor, *Handbook of Logic in Artificial Intelligence. Volume 1*. Oxford University Press, 1993. forthcoming.

[Gan91] H. Ganzinger. A completion procedure for conditional equations. *Journal of Symbolic Computation*, 11:51–81, 1991.

[Gas92] Olivier Gasquet. Deduction for multimodal logics. In *Proc. of Applied Logic Conference (Logic at Work). Amsterdam*, December 1992.

[Gen91] Ian P. Gent. *Analytic Proof Systems for Classical and Modal Logics of Restricted Quantification*. PhD thesis, University of Warwick, Coventry, England, 1991.

[GH93] Olivier Gasquet and Andreas Herzig. Theorem proving for non-normal modal logics. Abstracts of the IJCAI 93 workshop on Automated Theorem Proving, 1993.

[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, pp 425–436.

[HB91] Bernhard Hollunder and Franz Baader. Qualifying number restrictions in concept languages. In *Proceedings of the 2nd International Conference on Knowledge Representation and Reasoning*, pages 335–346, Cambridge, Mass., 1991.

[HC84] G.E. Hughes and M.J. Cresswell. *A Companion to Modal Logic*. Methuen & Co., London, 1984.

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

[Her89] Andreas Herzig. *Raisonnement automatique en logique modale et algorithmes d'unification*. PhD thesis, Université Paul-Sabatier, Toulouse, 1989.

[Her90] Andreas Herzig. A translation from propositional modal logic G into K4. Université Paul Sabatier, Toulouse, July 1990.

[HOS93] Ullrich Hustadt, Hans Jürgen Ohlbach, and Renate Schmidt. Qualified number restrictions and modal logic. Forthcoming MPI report, 1993.

[JR88] Peter Jackson and Han Reichgelt. A general proof method for modal predicate logic without the Barcan Formula or its converse. DAI Research Report 370, Department of Artificial Intelligence, University of Edinburgh, 1988.

[McC90] William McCune. OTTER 2.0. In Mark Stickel, editor, *Proc. of* 10^{th} *Internation Conference on Automated Deduction, LNAI 449*, pages 663–664. Springer Verlag, 1990.

[Non93] Andreas Nonnengart. First-order modal logic theorem proving and functional simulation. In *Proc. of IJCAI 93*, 1993.

[Ohl88] Hans Jürgen Ohlbach. A resolution calculus for modal logics. In Ewing Lusk and Ross Overbeek, editors, *Proc. of* 9^{th} *International Conference on Automated Deduction, CADE-88 Argonne, IL*, volume 310 of *Lecture Notes in Computer Science*, pages 500–516, Berlin, Heidelberg, New York, 1988. Springer-Verlag. extended version: SEKI Report SR-88-08, FB Informatik, Universität Kaiserslautern, 1988.

[Ohl90] Hans Jürgen Ohlbach. Semantics based translation methods for modal logics. SEKI Report SR-90-11, FB. Informatik, Univ. of Kaiserslautern, 1990.

[Ohl93] Hans Jürgen Ohlbach. Optimized translation of multi modal logic into predicate logic. In Andrei Voronkov, editor, *Proc. of Logic Programming and Automated Reasoning (LPAR)*. Springer LNAI, 1993.

[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.

[OS93] Hans Jürgen Ohlbach and Renate Schmidt. Optimized functional translation of multi modal logic into predicate logic. Forthcoming MPI Report, 1993.

[Sch93] Richard Brian Scherl. *A Constraint Logic Approach to Automated Modal Deduction.* PhD thesis, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, May 1993. Reprot No. UIUCDCS-R-93-1803.

[Sim93] Harold Simmons. The monotonous elimination of predicate variables. *Journal of Logic and Computation*, 1993. Forthcoming.

[SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin, Heidelberg, New York, 1989.

[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.

[vB84] Johan van Benthem. Correspondence theory. In Gabbay Dov M and Franz Guenthner, editors, *Handbook of Philosophical Logic, Vol. II, Extensions of Classical Logic, Synthese Library Vo. 165*, pages 167–248. D. Reidel Publishing Company, Dordrecht, 1984.

[vdH92] Wiebe van der Hoek. *Modalities for Reasoning about Knowledge and Quantities.* PhD thesis, Vrije Universiteit Utrecht, 1992.

[Wal87a] Lincoln A. Wallen. Matrix proof methods for modal logics. In *Proc. of* 10^{th} *IJCAI*, pages 917–923. IJCAI, Morgan Kaufmann Publishers, 1987.

[Wal87b] Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Pitman Ltd., London, 1987.