MAX-PLANCK-INSTITUT
FÜR
INFORMATIK

Synthesizing Semantics for Extensions of Propositional Logic

Hans Jürgen Ohlbach

MPI–I–94–225 June 94

INFORMATIK

Im Stadtwald
D 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

Acknowledgements

I would like to thank Dov Gabbay as well as my colleagues Andreas Nonnengart and Renate Schmidt for very fruitful discussions.

This work was supported by the ESPRIT project 6471 MEDLAR and by the BMFT funded project LOGO (ITS 9102).

Abstract

Given a Hilbert style specification of a propositional extension of standard propositional logic, it is shown how the basic model theoretic semantics can be obtained from the axioms by syntactic transformations. The transformations are designed in such a way that they eliminate certain derived theorems from the Hilbert axiomatization by turning them into tautologies.

The following transformations are considered. Elimination of the reflexivity and transitivity of a binary consequence relation yields the basic possible worlds framework. Elimination of the congruence properties of the connectives yields weak neighbourhood semantics. Elimination of certain monotonicity properties yields a stronger neighbourhood semantics. Elimination of certain closure properties yields relational possible worlds semantics for the connectives.

Propositional logic as basis of the specification allows to turn those parts which are not eliminated into second–order predicate logic (PL2) formulae. In many cases these formulae can be simplified to an equivalent first–order predicate logic (PL1) formula which describes the corresponding frame property. All transformations work for arbitrary n-place connectives. The steps can be fully automated by means of PL1 theorem provers and quantifier elimination algorithms. The meta theory guarantees that all transformation steps are sound and complete. As a by–product, translations into multi–modal logic are developed.

Key Words: Logic, Transformations, Semantics

Contents

1 Introduction

Logics can be defined in various ways. The most abstract way is by means of a Hilbert system. A Hilbert system is a kind of grammar. It specifies how to enumerate all formulae to be considered as theorems. For encoding vague notions, like "knows", "believes" and "wants", a Hilbert style axiomatization is usually the method of choice because in Hilbert systems their properties can be expressed in a very abstract and intuitive way. Proving theorems in Hilbert systems, however, is extremely inefficient. It involves enumerating the theorems until the formula under consideration is eventually obtained. (Lemma A.2, for example, gives a good impression what theorem proving from Hilbert axioms means.)

One reason for developing an alternative to Hilbert systems is the desire to get more efficient calculi. Another reason is to understand the logic better by bringing properties to the surface which are sometimes very deeply hidden. Who, besides Łukasiewics, would for example guess that the Hilbert axiom

$$(p \to q) \to r) \to ((r \to p) \to (s \to p)) \qquad (1)$$

$$\text{together with the rule:} \quad \text{from } p \text{ and } p \to q \text{ infer } q \qquad (2)$$

specify the implicational fragment of propositional logic [Łuk70, p. 295]? An alternative way to describe a logic is by mapping the syntactic constructs to a (hopefully) simple and well understood mathematical structure, a Boolean algebra or more generally, a structure, i.e. a set with relations and functions defined over this set. The mapping must include an interpreter or a satisfiability relation for evaluating the truth value of a given formula in the structure. Typical examples for this *semantical* description of a logic are Tarski's set theoretic semantics for predicate logic or Kripke's possible worlds semantics for modal logic [Kri59, Kri63].

The usual way the correlations between the axiomatic description and the semantics are presented is: the axioms and the semantics are defined and soundness and completeness are proved. Soundness and completeness guarantee that a formula is a theorem in the axiomatic description if and only if it is a valid formula in the semantics. Finding an appropriate semantic structure, however, is nontrivial and requires experience and intuition. The purpose of this paper is to overcome this problem and to show how the process of *finding a semantics can be automated* to a large extend. The long–term goal of this work is the development of a *logic engineering workbench* where *application oriented logics* can be specified on a very abstract level and the investigation of the so defined logic can be left to the computer.

As examples for semantics of a logic consider the different versions of the semantics of modal logic. Common to all of them is the possible worlds framework as basic semantic structure. Each possible world determines the interpretation of the propositional variables and the classical connectives in the usual way. The interpretation of formulae with non–classical operators is defined in terms of relations or functions connecting the worlds. The weakest semantics for modal logic is the (weak) neighbourhood semantics (also called minimal model semantics [Che80]). Each world has sets of worlds as 'neighbourhoods'. A formula $\Box p$ is true in a world w iff the truth set of p, i.e. the set of worlds where p is true, is among w's neighbourhoods. This semantics satisfies the ME rule, $p \Leftrightarrow q$ implies $\Box p \Leftrightarrow \Box q$, but no stronger axiom or rule. In strong neighbourhood semantics, $\Box p$ is true in a world w iff one of w's neighbourhoods is a *subset* of p's truth set. Strong neighbourhood semantics satisfies a monotonicity property: $p \Rightarrow q$ implies $\Box p \Rightarrow \Box q$. The next stage is the well known Kripke semantics with a binary accessibility relation. But this is not the end of the story. For example, modal logic S5 has a semantics in terms of an accessibility relation with the extra condition that the accessibility condition is an equivalence relation. This condition guarantees that the S5 axioms hold. An alternative semantics for S5 has the truth condition for the $\Box$–operator: $\Box p$ is true in a world iff p is true everywhere. In this semantics without accessibility relation, all S5 axioms are tautologies.

Each version of the semantics consists of two parts. The *basic semantics* contains just the definition of the primitive notions, neighbourhood relations or accessibility relations for example, and the satisfiability relation. The possible worlds together with the relations and functions operating on them are usually called *frames*.

The second part of the full specification of the semantics restricts the class of semantic structures by imposing constraints on the frames (so called *frame conditions*) and sometimes by restricting the assignment of truth values to the propositional variables. Modal logic T, for example is characterized by restricting the class of frames to those with reflexive accessibility relations. Intuitionistic logic as another example has a restriction on the assignment of propositional variables: if p is true in a world w then it remains true in all words accessible from w.

Each part of the semantics validates a certain part of the Hilbert axioms. The basic semantics of normal modal logic with binary accessibility relation for example validates the K-axiom $\Box(p \Rightarrow q) \Rightarrow (\Box p \Rightarrow \Box q)$ and the Necessitation rule: from p infer $\Box p$. The reflexivity condition on the accessibility relation validates the axiom $\Box p \Rightarrow p$. Obviously, there is a hierarchy of semantics. A semantics S_1 is *stronger* than a semantics S_2 if the basic part of S_1 validates more axioms than the basic part of S_2. A semantics is optimal for a Hilbert system if all axioms are validated in the basic part and no extra conditions are needed.

The questions we address in this paper are: where do these semantic notions, possible worlds, neighbourhoods, accessibility relations, satisfiability relations etc. come from? Is their invention really a creative act, or is there a systematic way to obtain them directly from the Hilbert axioms? The second question is, can one compute the frame conditions, i.e. the restrictions on a given semantic structure corresponding to Hilbert axioms with an automatic procedure? To answer these questions I proceed as indicated in the following picture.

$$\begin{tikzcd} & & & \text{Transformation rules } \Upsilon = \text{semantics definitions of the connectives} \\ HS \arrow[r] & PL1(HS) \arrow[r, "\text{Transformation}"] & {} \arrow[ur] \arrow[dr] & \\ & & & \Upsilon(PL1(HS)) = \text{frame conditions} \end{tikzcd}$$

We begin with an arbitrary Hilbert system HS. Using first–order predicate logic as meta logic, the first step is to encode HS in PL1. In the PL1 axiomatization $PL1(HS)$, the logical connectives are encoded as function symbols and formulae are encoded as terms. The propositional variables in Hilbert axioms are placeholders for arbitrary formulae. Therefore they become universally quantified variables in $PL1(HS)$. Actually we need a PL1 axiomatization of the Hilbert system in terms of a binary consequence relation $\Vdash^2$. $p \Vdash^2 q$ means q is derivable from p[1].

The key observation underlying my approach for systematically finding a semantics comes from automated theorem proving. In order to prove a conjecture from assumptions, one need not perform the proof directly, but we can transform assumptions and conjectures and prove the transformed conjectures from the transformed assumption, provided the transformation Υ guarantees

$$\mathit{assumption} \Rightarrow \mathit{conjecture} \ \text{ iff } \ \Upsilon(\mathit{assumption}) \Rightarrow \Upsilon(\mathit{conjecture}). \tag{3}$$

The working hypothesis of this paper is therefore

> *The semantics of an axiomatically defined logic is the result of a carefully designed transformation of predicate logic formulae. The transformation rules represent the interpretation function and the non–tautologous transformed Hilbert axioms are the frame conditions.*

An optimal transformation turns all axioms into tautologies. In this case, proving a theorem φ from HS reduces to proving $\Upsilon(\varphi)$ without any additional assumptions. The guideline for finding a good transformation is therefore the intention to *turn axioms into tautologies*, or to make them in some other way redundant.

[1]If the Hilbert system is originally specified with a unary 'provability' predicate $\Vdash^1$, the deduction theorem $p \Vdash^2 q$ iff $\Vdash^1 (p \rightarrow q)$ can be used to translate it into the formulation with a binary consequence relation. This, however, excludes axiomatizations in terms of the provability predicate where the top–level connectives of the formulae are not some sort of implication.

Ideally one would like to have a procedure $Trans\_Gen$ which, given a formula φ as input, computes a transformation Υ_φ with property (3) such that $\Upsilon_\varphi(\varphi) = true$. If $PL1(HS)$ consists of the formulae $\varphi_1, \ldots, \varphi_n$ one would compute $\Upsilon_{\varphi_1} = Trans\_Gen(\varphi_1)$, get $\varphi'_i = \Upsilon_{\varphi_1}(\varphi_i)$ for $i = 2, \ldots, n$, compute $\Upsilon_{\varphi'_2} = Trans\_Gen(\varphi'_2)$, apply it to $\varphi'_3, \ldots, \varphi'_n$ and repeat the process until all axioms are turned into tautologies. And, in fact, in [OGP94], we sketch such a procedure $Trans\_Gen$. Unfortunately this idea does not work in general. The reason is that Υ may produce an infinite conjunction of formulae, and then the process does not terminate. But even if this method worked, it would not be very satisfactory because we do not want any transformation, but a transformation that gives some insight into the structure of the logic. I therefore propose an approach which is restricted to a certain class of logics, but gives better results for them. The idea is to develop transformations for very concrete formulae or formula schemas. For each given Hilbert system, we then check which of the formulae out of our database of formulae with known transformations is a theorem. That means we decompose

$$PL1(HS) \Leftrightarrow \varphi_1 \wedge \ldots \wedge \varphi_k \wedge rest$$

and for the φ_i we develop transformations once and for all. The transformations for $\varphi_1 \wedge \ldots \wedge \varphi_k$ specify the basic semantics and the transformation of the *rest* gives the frame conditions. The properties φ_i investigated in this paper are the n–place connective versions of the following four properties of a one–place $\Box$–operator, formulated with a binary consequence relation $\stackrel{2}{\vdash}$.

φ_1	$=$	reflexivity and transitivity of $\stackrel{2}{\vdash}$	
φ_2	$=$	$p \stackrel{2}{\vdash} q \wedge q \stackrel{2}{\vdash} b \Rightarrow \Box p \stackrel{2}{\vdash} \Box q$	(ME rule)
φ_3	$=$	$p \stackrel{2}{\vdash} q \Rightarrow \Box p \stackrel{2}{\vdash} \Box q$	(monotonicity)
φ_4	$=$	$w \stackrel{2}{\vdash} \Box p \wedge w \stackrel{2}{\vdash} \Box q \Rightarrow w \stackrel{2}{\vdash} \Box(p \wedge q)$.	(closure property)

These properties are of a very basic nature. They hold in many logics. The transformations we have developed for φ_1 to φ_4 reconstruct the well known semantic structures. φ_1 gives rise to the basic possible worlds framework and the satisfiability relation. Elimination of φ_2 reconstructs weak neighbourhood semantics, elimination of φ_3 reconstructs strong neighbourhood semantics, and finally, φ_4 is responsible for the usual accessibility relation.

Although in this paper, only these four properties are investigated, the technique for finding the transformations is not restricted to these cases. Following our recipe, it should not be too difficult to extend the results to other properties.

Since the transformations for φ_1 to φ_4 are sound and complete (property (3)), each given Hilbert system for which some of the φ_i have been proved can be transformed, and we are guaranteed that a transformed theorem follows from the transformed Hilbert system if and only if it follows from the original system. This gives us a general soundness and completeness result for the whole class of logics with these properties φ_i. Proving the φ_i from a given Hilbert system is usually nontrivial. But fortunately it is a pure PL1 theorem proving problem which can be solved with an automated theorem prover.

As we shall see, the result of the transformation of a Hilbert system is an axiomatization of the corresponding frame properties. A considerable simplification of the axioms describing the frame properties is possible if a propositional logic is part of the axiomatization. The Lindenbaum algebra of a propositional logic is a Boolean algebra, which, by Stone's representation theorem, is isomorphic to a field of sets. This property can be exploited to turn formula variables, which in the PL1 formulation are ordinary variables ranging over some domain, into set variables ranging over the powerset of the set of 'worlds'. Each such set can be correlated with a formula which is valid exactly at this set. Thus, ordinary variables become formula variables. By this transformation the Hilbert axioms which do not become tautologies become second–order predicate logic formulae. In many cases, however, these PL2 formulae are equivalent to PL1 formulae describing properties of the neighbourhood or accessibility relations. Thus, we have a two-step transformation. The first step translates the original Hilbert axiom into a PL1 formula. The original formula variables are ordinary PL1 variables. The transition to PL2 turns these variables into predicate variables. In the second step the predicate variables are completely eliminated by a quantifier elimination procedure and an equivalent PL1 formula is computed – if there is one. This formula specifies a frame property. That means the *correspondence problem* [vB84], i.e. the problem of finding for a given Hilbert axiom a corresponding property at the semantic level, reduces to the problem of finding for a PL2 formula an equivalent PL1 formula. This problem can in many cases be solved with a quantifier elimination algorithm. The algorithm we have developed for this purpose [GO92a] is briefly described in Section 2.2.

PL2 formulae without PL1 equivalents specify so–called incomplete logics. Since the transition to PL2 formulae which turns ordinary variables into predicate variables is not mandatory, we still have a way to deal with these incomplete systems.

The transformation techniques are described in the next section. Section 4 is the main section of the paper. The particular transformations are introduced for arbitrary n-place connectives and soundness and completeness is proved. In order to illustrate the various aspects of the method and to demonstrate that the investigation of a concrete Hilbert system can be automated, we use a very complicated test example. For proving the key lemmas for this test example we used the Otter theorem prover. All proofs are listed in the Appendix.

Warning!

In this paper we use predicate logic as meta logic for the 'object logic' we want to investigate. Therefore there are two kinds of semantics involved. The standard semantics of predicate logic is needed when we talk about the general mechanisms and verify our methododology. The semantics of the object logic, on the other hand, is the result of the transformations of the Hilbert system. This should not be mixed up.

2 Transformation Operations

The transformations needed to synthesize the semantics follow essentially the same few schemas, K-transformations [OGP94] and quantifier elimination [GO92a]. A short overview of both is given in this section.

2.1 K-Transformations

To illustrate the basic idea of K-transformations[2], suppose we have some set Φ of axioms which, among other things, axiomatize a reflexive and transitive relation R, i.e.

$$\forall x\ R(x,x) \tag{4}$$

$$\forall x,y,z\ R(x,y) \wedge R(y,z) \Rightarrow R(x,z) \tag{5}$$

are either contained in Φ or derivable from Φ, and we want to get rid of the reflexivity and transitivity of R.

In order to show that a formula C is entailed by Φ, one usually tries to refute $\Phi \wedge \neg C$. Before the refutation is actually started, every transformation on $\Phi \wedge \neg C$ which preserves satisfiability and unsatisfiability is allowed. Skolemization of existential quantifiers is a typical example of a routinely applied transformation which preserves satisfiability and unsatisfiability, but not logical equivalences.

The translation we propose for eliminating reflexivity and transitivity of R exploits that these two properties together imply

$$\forall x,y\ R(x,y) \Leftrightarrow (\forall w\ R(w,x) \Rightarrow R(w,y)). \tag{6}$$

To see this, suppose $R(x,y)$ and $R(w,x)$ hold. By transitivity, $R(w,y)$ also holds, i.e. the "$\Rightarrow$"–part is shown. For the "$\Leftarrow$"–part, take $w = x$ and use the reflexivity of R to derive $R(x,y)$.

[2]'K-transformation' is short for 'Killer transformation'. These transformations 'kill' certain properties from axiomatizations.

Since (6) is entailed by Φ, we could add it to $\Phi \wedge \neg C$ without loosing satisfiability or unsatisfiability. However, instead of (6), we add

$$\forall x, y\ R(x, y) \Leftrightarrow (\forall w\ R'(w, x) \Rightarrow R'(w, y)) \tag{7}$$

to $\Phi \wedge \neg C$ where R' is a *new* predicate symbol. Clearly, if $\Phi \wedge \neg C$ is satisfiable then $\Phi \wedge \neg C \wedge (7)$ is also satisfiable: the interpretation of R' can be chosen to be the same as the interpretation of R. In this case (7) is equivalent to (6), which follows from Φ. Thus, (7) is also true in the extended interpretation. On the other hand, if $\Phi \wedge \neg C \wedge (7)$ is satisfiable then certainly $\Phi \wedge \neg C$ is satisfiable as well.

But now we have a definition of R in terms of R' where R' is an uninterpreted new predicate symbol. In the next step, (7) is used as a rewrite rule from left to right, replacing all occurrences of R in $\Phi \wedge \neg C$ by the formula with R'. We obtain the transformed formula $\Phi' \wedge \neg C' \wedge (7)$ with R' in place of R. This is a terminating equivalence preserving transformation.

What happens to the reflexivity and transitivity of R? $\forall x\ R(x, x)$ becomes $\forall x\ \forall w\ R'(w, x) \Rightarrow R'(w, x)$ which is a tautology (by the reflexivity of "$\Rightarrow$"). The transitivity (5) becomes $\forall x, y, z\ (\forall w\ R'(w, x) \Rightarrow R'(w, y)) \wedge (\forall w\ R'(w, y) \Rightarrow R'(w, z)) \Rightarrow (\forall w\ R'(w, x) \Rightarrow R'(w, z))$ which is also a tautology (by the transitivity of $\Rightarrow$). Thus, R' need neither be reflexive nor transitive.

Nothing would have been gained if the definition (7) of R, could not be removed afterwards. That means we have to show that $\Phi' \wedge \neg C' \wedge (7)$ is satisfiable if and only if $\Phi' \wedge \neg C'$ is satisfiable. Since $\Phi' \wedge \neg C'$ does not contain R any more, we can always find an interpretation for R, using (7) as definition. Therefore each model for $\Phi' \wedge \neg C'$ can be extended to a model for $\Phi' \wedge \neg C' \wedge (7)$. Thus, (7) can be eliminated. $\Phi' \wedge \neg C'$ is the final result of our transformation.

A small example illustrates how this works. Suppose, besides transitivity of R we have the facts

$$R(a, b) \wedge R(b, c) \wedge R(c, d)$$

From this, $R(a, d)$ can be derived. The transformed formulae are:

$$\begin{array}{l} \forall w\ R'(w, a) \Rightarrow R'(w, b) \\ \forall w\ R'(w, b) \Rightarrow R'(w, c) \\ \forall w\ R'(w, c) \Rightarrow R'(w, d). \end{array}$$

The negated and transformed theorem is $\neg \forall w\ R'(w, a) \Rightarrow R'(w, d)$ which is normalized to $R'(e, a) \wedge \neg R'(e, d)$ where e is a Skolem constant. The transformed formulae can now be refuted without transitivity of R'[3].

What has actually happened is that the role of the reflexivity and transitivity of R has been taken over by the reflexivity and transitivity of the implication connective. Many other examples of K-transformations are of a similar kind. The built in properties of predicate logic take over the role of special properties of non–logical symbols.

The General Transformation Procedure

The general procedure for transforming formulae Φ in a consistent way, i.e. without loosing satisfiability or unsatisfiability, consists of the following sequence of steps

$$\Phi \xrightarrow{\text{extension}} \Phi \wedge \text{transformer} \xrightarrow{\text{transformation}} \Phi' \wedge \text{transformer} \xrightarrow{\text{elimination}} \Phi'$$

where 'transformer' is a formula of the kind

$$Left \Leftrightarrow Right. \tag{8}$$

(7) is an example for (8). The 'extension' step involves finding the transformer. In general this is still a creative step. In [OGP94], however, we describe methods for automating this step to a

[3]In [OGP94] we optimize the transformation for reflexivity and transitivity. If reflexivity is kept and R is not renamed, then only the positive R–literals need to be transformed. For the purposes of this paper, however, this optimization is irrelevant.

certain extend. The actual transformation is done in the 'transformation' step. In the simplest case the transformation strategy is just definitional replacement where (8) is used as rewrite rule form left to right. It can, however, also be a much more complex combination of rewriting, inferencing and deleting formulae. In the elimination step we delete the transformer. Since removing formulae can turn unsatisfiable formula sets into satisfiable sets, this is also a nontrivial step which has to be justified. To ensure that the transformation is satisfiability preserving, which is sufficient to do theorem proving by refutation, the following lemmas have to be proved.

Definition 2.1 (Transformation Lemmas)
The *extension lemma* proves that satisfiability of Φ implies satisfiability of $\Phi\ \wedge$ transformer.
The *transformation lemma* proves that $\Phi\wedge$ transformer is satisfiable if and only if $\Phi'\wedge$ transformer is satisfiable, where Φ' is the transformed version of Φ.
The *elimination lemma* proves that satisfiability of Φ' implies satisfiability of $\Phi'\ \wedge$ transformer. $\lhd$

Lemma 2.2 (Faithfulness)
A K-transformation for which the extension lemma, the transformation lemma, and the elimination lemma have been proved, preserves satisfiability and unsatisfiability. We call this a *faithful* transformation. It is *sound and complete.* $\lhd$

There are standard cases of transformations which occur quite frequently. For these cases we can prove some of the lemmas once and for all. In an actual case, it has only to be checked whether the transformer is one of these standard types.

Definition 2.3 (Renamed Transformer)
A transformer $Left \Leftrightarrow Right$ is called a *renamed transformer* for a formula Φ iff there is a formula $Left \Leftrightarrow Right_0$ entailed by Φ and $Right$ is obtained from $Right_0$ by renaming constant, function, predicate symbols and sorts with new symbols not occurring in Φ. Different occurrences of the same symbol in $Right_0$ may be renamed differently. $Left \Leftrightarrow Right_0$ is called the *basis* of the transformer. $\lhd$

Lemma 2.4 (Extension Lemma for Renamed Transformers)
The extension lemma (Def. 2.1) holds for renamed transformers. $\lhd$

The proof of the extension lemma for this type of transformers amounts to proving that the basis $Left \Leftrightarrow Right_0$ follows from Φ. This is a standard theorem proving task which can be done with automated theorem provers.

The transformer (7) for the elimination of reflexivity and transitivity is an example for a renamed transformer. The proof of $Left \Leftrightarrow Right_0$, i.e. (6) is trivial in this case.

As we have seen in the example with reflexivity and transitivity, the renamed predicate R' need not be reflexive or transitive any more. An important observation is, however, that we can assume that the renamed symbols have some or all properties of the original symbols. The proof showing that there is an interpretation for (7) just interpreted R' like R, and therefore R' has R's properties in this interpretation. Thus, we are free to add suitable axioms for R' without changing satisfiability or unsatisfiability.

In the class of transformers specified in the next definition, the transformation process itself is reduced to a simple rewriting operation.

Definition 2.5 (Rewriting Transformers)
A transformer $\forall x_1,\ldots,x_n\ R(x_1,\ldots,x_n) \Leftrightarrow Right$ where R does not occur in $Right$, is called a *rewriting transformer*. The transformation strategy for rewriting transformers is just definitional replacement, i.e. all occurrences $R(s_1,\ldots,s_n)$ are replaced with the corresponding instances of $Right$. $\lhd$

Lemma 2.6 (Transformation Lemmas for Rewriting Transformers)
The transformation lemma and the elimination lemma (Def. 2.1) hold for rewriting transformers. $\lhd$

Renamed rewriting transformers are the simplest transformers of all. The only thing which has to be proved for this class of transformers is that the basis $Left \Leftrightarrow Right_0$ for the transformer follows from Φ. The transformer (7) turns out to be of this simple type.

The next class of transformers is of a more general nature. It allows us to exploit completeness results for special inference strategies, as for example ordered resolution. (24) is an example for a transformer of this kind.

Definition 2.7 (Saturation Transformers)
A transformer $Left \Leftrightarrow Right$ is called a *saturation transformer* for the formula Φ if there is a refutation complete deduction strategy and it can be shown that according to this strategy only finitely many inferences between the transformer itself and the formulae in Φ are possible.
Φ is transformed by drawing all inferences between the transformer and Φ, this strategy allows. Redundant formulae which are no longer necessary for finding a contradiction are removed. $\lhd$

Lemma 2.8 (Transformation Lemmas for Saturation Transformers)
The transformation lemma and the elimination lemma (Def. 2.1) hold for saturation transformers. $\lhd$

The formula Φ which is transformed usually consists of the two parts, the assumption and the negated onjecture. At the time of the development of the K-transformation, usually only the assumption is known. Therefore the transformation lemma has to be proved for all potential conjectures. Alternatively one can specify the class of admissible conjectures for which the K-transformation works.

2.2 Quantifier Elimination

In [GO92a] 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 be applied to universally quantified predicate variables. Related methods can also be found in [Ack35a, Ack35b, Ack54, Sza92, BGW92, Sim94].

The definition of the algorithm is:

Definition 2.9 (The SCAN Algorithm)
Input to SCAN is a formula $\alpha = \exists P_1, \ldots, P_n\ \Phi$ with predicate variables $P_1, \ldots, P_n$ and an arbitrary first–order formula Φ.
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. Φ 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:

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

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). If all clauses are deleted this way, this means that α is a tautology.

All equivalence preserving simplifications may be applied freely. 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. $\lhd$

The SCAN algorithm is correct in the sense that its result is logically 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 cannot find it. An algorithm which is complete in this sense cannot exists, otherwise the theory of arithmetic would be enumerable.

The points where SCAN can fail to compute a first–order equivalent for α 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 with existentially quantified Skolem functions.

3 A Test Example

For illustrating the various transformation steps, we use a test example. It is a propositional logic extended with two more operators. The first one is a binary connective $\leadsto$ as an example for a non–classical implication operator. $\leadsto$ is axiomatized with a variant (12) of the Łukasiewics axiom (1) and its own version of Modus Ponens (13) below, both formulated with the binary consequence relation.

Since, by the deduction theorem, the binary consequence relation corresponds to the propositional implication $\rightarrow$, (12) is not precisely the Łukasiewics axiom. The difference is that the top–level connective definitely is the propositional implication. For the connective $\leadsto$ we pretend for the moment not to know anything besides the axioms. The result of our analysis should of course be that this connective is in fact also propositional implication.

The second connective in our test example is the modal $\Box$–operator. The $\Box$–operator is axiomatized with the K-axiom and the Necessitation rule. Thus, we should be able to reproduce standard relational possible worlds semantics. To make things more interesting, however, the K-axiom is not formulated with the known propositional implication $\rightarrow$, but with the new connective $\leadsto$. The semantics of the $\Box$–operator therefore depends on the semantics of the $\leadsto$–connective.

Example 3.1 (Our Main Test Example)
We use the following seven formulae as our test example. The first three formulae axiomatize propositional logic. They are not important. Any other axiomatization of propositional logic would do as well. The next two axioms formalize our "new" connective $\leadsto$. We pretend not to know anything more about it. The last two formulae axiomatize the modal $\Box$–operator. (14) is a formulation of the Necessitation rule '$\vdash^1 P$ implies $\vdash^1 \Box P$' in terms of the binary consequence relation. (15) is a version of the K-axiom, formulated with the 'unknown' connective $\leadsto$.

$$\forall p,q,r,s \quad ((p \rightarrow q) \rightarrow r) \vdash^2 ((r \rightarrow p) \rightarrow (s \rightarrow p)) \tag{9}$$
$$\forall p,q,r,s \quad p \vdash^2 q \wedge (p \rightarrow q) \vdash^2 (r \rightarrow s) \Rightarrow r \vdash^2 s \tag{10}$$
$$\forall p \quad \bot \vdash^2 p \tag{11}$$
$$\forall p,q,r,s \quad ((p \leadsto q) \leadsto r) \vdash^2 ((r \leadsto p) \leadsto (s \leadsto p)) \tag{12}$$
$$\forall p,q,r,s \quad p \vdash^2 q \wedge (p \leadsto q) \vdash^2 (r \leadsto s) \Rightarrow r \vdash^2 s \tag{13}$$
$$\forall p \quad (\forall q\ q \vdash^2 p) \Rightarrow (\forall q\ q \vdash^2 \Box p) \tag{14}$$
$$\forall p,q \quad \Box(p \leadsto q) \vdash^2 (\Box p \leadsto \Box q). \tag{15}$$

$\lhd$

In the sequel let $\mathcal{A}_1$ be the initial Hilbert system for the logic $\mathcal{L}$ we want to investigate. The set consisting of (9) to (15) is an example for $\mathcal{A}_1$. The $\Box$–operator as part of the test example shows the correspondence between the techniques developed in this paper and the well known facts about modal logic. The generalization of the semantics construction from a one–place operator like $\Box$ to an non–classical n-place operator is illustrated with the binary $\leadsto$ connective. I recommend the reader to follow this example carefully and in particular to compare the development of the semantics for the $\Box$–operator with the standard methods you can find in the literature, for example in [Che80].

We use a many–sorted version of PL1 [Wal87] and we assume that the variables which are to be understood as formula variables are of sort F (for Formula). In most cases we use the letters p, q, r, s for formula variables without explicitly annotating them with the sort symbol F.

4 The Transformations

In this section, which is the main section of the paper, the transformation process is described. The transformations are designed to *eliminate properties* of $\mathcal{A}_1$, or more precisely, absorb these properties into the properties of our meta logic, predicate logic. First we eliminate the reflexivity and transitivity of $\mathrel{|^{2}}$. This transformation introduces the basic possible worlds framework and the satisfiability relation. In the second and third step, the congruence properties and the monotonicity properties of the connectives are eliminated. This gives rise to so–called neighbourhood semantics. Finally in the last step certain closure properties are eliminated and we reconstruct possible worlds semantics with $n+1$–ary accessibility relations for n–place connectives.

4.1 Reflexivity and Transitivity

To start with, we want to eliminate reflexivity and transitivity of the consequence relation $\mathrel{|^{2}}$. The method for eliminating reflexivity and transitivity has been explained in Section 2.1. The transformer is $\forall p, q\ p \mathrel{|^{2}} q \Leftrightarrow (\forall w\ w \models^{\in} p \Rightarrow w \models^{\in} q)$. But we can go a step further and introduce a new sort W for the variable w. The first one of our sequence of transformers is therefore for all p, q:

$$\mathcal{T}_1: \qquad p \mathrel{|^{2}} q \Leftrightarrow (\forall w{:}W\ \ w \models^{\in} p \Rightarrow w \models^{\in} q) \tag{16}$$

which is still a renamed rewriting transformer and therefore sound and complete.

We want to interpret the sort W as the set of 'worlds' in a possible worlds semantics and the predicate $\models^{\in}$ as the satisfiability relation. The intended reading for (16) is therefore really 'q is entailed by p iff q is true in all worlds satisfying p'. But for the moment (16) is still to be understood as an ordinary predicate logic formula.

Completeness of $\mathcal{T}_1$ gives us our first completeness lemma.

Lemma 4.1 (Completeness Lemma for Possible Worlds)
If $\mathcal{A}_1$ implies reflexivity and transitivity of $\mathrel{|^{2}}$ then for every formula φ: $\mathcal{A}_1$ entails φ iff $\mathcal{A}_2 \stackrel{\text{def}}{=} \mathcal{T}_1(\mathcal{A}_1)$ entails $\mathcal{T}_1(\varphi)$. $\lhd$

Notation: Frequently we will use the following notation: Instead of $w \models^{\in} p$ we write $w \in |p|$. Syntactically, this is just an abbreviation. It allows us to abbreviate formulae '$\forall w\ w \models^{\in} p \Rightarrow w \models^{\in} q$' by '$|p| \subseteq |q|$' and formulae '$\forall w\ w \models^{\in} p \Leftrightarrow w \models^{\in} q$' by '$|p| = |q|$'. Semantically, however, this $|...|$ notation has a particular meaning. $|...|$ can be understood as a function mapping domain elements to sets: $|p| \stackrel{\text{def}}{=} \{w \mid (w, \Im(p)) \in \Im(\models^{\in})\}$ where $\Im$ is any predicate logic interpretation. On the level of predicate logic semantics, defined by some $\Im$, expressions like $w \in |p|$ and $|p| \subseteq |q|$ and $|p| = |q|$ are to be understood literally. As we shall see in Subsection 4.4, under certain circumstances, $\models^{\in}$ can be interpreted as membership relation. In this case this abbreviating notation is equivalent to the original notation.

The methods discussed in this paper work for arbitrary n–place connectives. To simplify notation we therefore use $\vec{p}$ as an abbreviation for $p_1, \ldots, p_n$ where n is the arity of the connectives under consideration.

As proved in Lemma A.1 and A.2 in the Appendix, the consequence relation in our test example (3.1) is in fact reflexive and transitive ands therefore the transformer $\mathcal{T}_1$ is applicable. If we apply the transformer $\mathcal{T}_1$ to the last four axioms of our test example, we obtain

$$\begin{array}{rl}
\forall p, q, r, s & \forall w{:}W\ w \models^{\mathcal{L}} ((p \rightsquigarrow q) \rightsquigarrow r) \Rightarrow w \models^{\mathcal{L}} ((r \rightsquigarrow p) \rightsquigarrow (s \rightsquigarrow p)) \\
\forall p, q, r, s & (\forall w{:}W\ w \models^{\mathcal{L}} p \Rightarrow w \models^{\mathcal{L}} q) \wedge (\forall w{:}W\ w \models^{\mathcal{L}} p \rightsquigarrow q \Rightarrow w \models^{\mathcal{L}} r \rightsquigarrow s) \\
& \quad \Rightarrow (\forall w{:}W\ w \models^{\mathcal{L}} r \Rightarrow w \models^{\mathcal{L}} s) \\
\forall p & (\forall q\ \forall w{:}W\ w \models^{\mathcal{L}} q \Rightarrow w \models^{\mathcal{L}} p) \Rightarrow (\forall q\ \forall w{:}W\ w \models^{\mathcal{L}} q \Rightarrow w \models^{\mathcal{L}} \Box p) \\
\forall p, q & \forall w{:}W\ w \models^{\mathcal{L}} \Box(p \rightsquigarrow q)) \Rightarrow w \models^{\mathcal{L}} (\Box p \rightsquigarrow \Box q).
\end{array}$$

Obviously these formulae are no tautologies. Therefore we have to go on and find stronger transformations.

4.2 Congruence Properties

After having eliminated reflexivity and transitivity of $\vdash^{2}$, we now attack certain congruence properties. As we shall see, eliminating these properties on the syntactic side introduces neighbourhood semantics (also called minimal model semantics [Che80]) on the semantic side.

Assumption: In the sequel we assume that $\vdash^{2}$ is the only predicate symbol occurring in $\mathcal{A}_1$. In this case only literals $w \models t$ where w is a variable occur in $\mathcal{A}_2$. Furthermore, we assume that all formula variables in $\mathcal{A}_1$ are *universally quantified.*

The congruence properties we want to eliminate are

$$\forall \vec{p}, \vec{q}\ \textstyle\bigwedge_i (p_i \vdash^{2} q_i \wedge q_i \vdash^{2} p_i) \Rightarrow f(\vec{p}) \vdash^{2} f(\vec{q}). \tag{17}$$

For verifying (17) from $\mathcal{A}_1$, it can be split into individual lemmas for each argument position.

$$\forall p, q\ (p \vdash^{2} q \wedge q \vdash^{2} p) \Rightarrow \forall \vec{s}, \vec{s}'\, f(\vec{s}, p, \vec{s}') \vdash^{2} f(\vec{s}, q, \vec{s}'). \tag{18}$$

The instances of (18) for our test example are

$$\forall p, q\ (p \vdash^{2} q \wedge q \vdash^{2} p) \Rightarrow \forall s\ p \rightsquigarrow s \vdash^{2} q \rightsquigarrow s) \tag{19}$$

$$\forall p, q\ (p \vdash^{2} q \wedge q \vdash^{2} p) \Rightarrow \forall s\ s \rightsquigarrow p \vdash^{2} s \rightsquigarrow q \tag{20}$$

$$\forall p, q\ (p \vdash^{2} q \wedge q \vdash^{2} p) \Rightarrow \Box p \vdash^{2} \Box q. \tag{21}$$

The proofs for these formulae are also listed in the Appendix (Lemmas A.3, A.4, A.5). The congruence property (21) for the $\Box$–operator corresponds to the well known ME rule, $\vdash^{1}(P \Leftrightarrow Q)$ implies $\vdash^{1}(\Box P \Leftrightarrow \Box Q)$, which is the only rule required for classical modal systems.

The transformer $\mathcal{T}_1$ translates the congruence properties (17) for the connectives f into

$$\forall \vec{p}, \vec{q}\ \textstyle\bigwedge_i (\forall w{:}W\ w \models^{\mathcal{L}} p_i \Leftrightarrow w \models^{\mathcal{L}} q_i) \Rightarrow (\forall w{:}W\ w \models^{\mathcal{L}} f(\vec{p}) \Rightarrow w. \models^{\mathcal{L}} f(\vec{q})) \tag{22}$$

A straightforward proof from this yields for all $\vec{p}, w$:

$$w \models^{\mathcal{L}} f(\vec{p}) \quad \Leftrightarrow \quad \exists \vec{x}{:}W\ w \models^{\mathcal{L}} f(\vec{x}) \wedge \textstyle\bigwedge_i \forall v{:}W\ v \models^{\mathcal{L}} x_i \Leftrightarrow v \models^{\mathcal{L}} p_i \tag{23}$$

as *the basis* for the next K-transformation, a renamed transformer (Def. 2.3), which is, for all $\vec{p}, w$:

$$\mathcal{T}_2\colon \qquad w \models^{\mathcal{L}} f(\vec{p}) \Leftrightarrow \exists \vec{x}{:}S\ N_f(w, \vec{x}) \wedge \textstyle\bigwedge_i \forall v{:}W\ R_f^i(x_i, v) \Leftrightarrow v \models^{\mathcal{L}} p_i. \tag{24}$$

Here we take advantage of the fact that different occurrences of a predicate may well be renamed differently in a renamed transformer. The '$w \models^{\mathcal{L}} f(\vec{x})$' occurrence is renamed to $N_f(w.\vec{x})$ and the

'$v \models^{\mathcal{L}} x_i$' occurrence is renamed to $R^i_f(x_i, v)$. *This is the fist place where we introduced relations between worlds as new semantic structures.* It is easy to check that $\mathcal{T}_2$ in fact turns the congruence properties (22) into tautologies.

The instances of (24) for our test example are:

$$w \models^{\mathcal{L}} (p \rightsquigarrow q) \Leftrightarrow \exists x, y{:}S\ N_{\rightsquigarrow}(w, x, y) \wedge (\forall v{:}W\ R^1_{\rightsquigarrow}(x, v) \Leftrightarrow v \models^{\mathcal{L}} p) \wedge$$
$$(\forall v{:}W\ R^2_{\rightsquigarrow}(y, v) \Leftrightarrow v \models^{\mathcal{L}} q) \tag{25}$$
$$w \models \Box p \Leftrightarrow \exists x{:}S\ N_{\Box}(w, x) \wedge (\forall v{:}W\ R_{\Box}(x, v) \Leftrightarrow v \models^{\mathcal{L}} p). \tag{26}$$

Lemma 4.2 (Soundness and Completeness of the Transformer $\mathcal{T}_2$)
If $\mathcal{A}_2$ implies the congruence properties (22) for *all* functions f and as transformation strategy, rewriting with $\mathcal{T}_2$ until all occurrences of the connectives disappear, is chosen, then the extension, the transformation and the elimination lemmas (Def. 2.1) hold.

Proof: Extension Lemma: This lemma holds because the sort symbol S can be interpreted like W, $N_f(w, \vec{x})$ can be interpreted as $w \models^{\mathcal{L}} f(w, \vec{x})$, $R^i_f(x_i, v)$ can be interpreted as $v \models^{\mathcal{L}} x_i$ and then (24) is true in the extended model because (23) is.
Transformation Lemma: Unfortunately the transformer $\mathcal{T}_2$ is no longer a simple definition for a predicate. The left hand side is of the form $w \models^{\mathcal{L}} f(\vec{p})$, where w and the p_i are variables. Since, by the first transformer $\mathcal{T}_1$, all literals in $\mathcal{A}_2$ are of the form $v \models^{\mathcal{L}} f(\vec{s})$ where v is a variable and the s_i are terms, the transformers can also be applied as rewrite rules. Due to the structure of the transformers' right hand sides, each rewrite step eliminates one occurrence of a connective at a time. Furthermore, because there is a transformer for *every* connective f, eventually all occurrences of the connectives are eliminated. We end up with formulae $\mathcal{A}_3$ containing only literals with the N_f and R^i_f predicates with variables as their arguments, and literals $v \models^{\mathcal{L}} p_i$ where the p_i are the original variables. So far, this is an equivalence transformation which does not affect the models at all.
Elimination Lemma: Although there are further inferences possible between the transformers and $\mathcal{A}_3$, we can delete the transformers now without changing the satisfiability or unsatisfiability. The argument is proof theoretic. We show that there is a complete inference strategy which does not allow to draw any further inferences with the transformers in this particular case. The completeness of the strategy then guarantees that the transformers are useless.
The strategy we use is ordered resolution with elimination of redundant clauses [BG90]. Ordered resolution requires the definition of an ordering on literals. Only resolvents between the biggest literals in a clause need to be generated and we still have a complete procedure. The redundancy criterion, Bachmair and Ganzinger [BG90] have shown to be complete is: inferences are redundant if the inferred clause follows from smaller (in the given ordering) clauses.
If we choose the ordering such that literals with the connectives f are bigger than the other literals, we immediately see that no resolvents with literals at the right hand side of the transformers are generated. These literals are smaller than the literal at the left hand side.
But there are still resolutions possible between the literals $w \models^{\mathcal{L}} f(\vec{p})$ from the left hand sides of the transformers and literals $v \models^{\mathcal{L}} p$ in $\mathcal{A}_3$, where p is a variable. We have assumed that these variables are of the basic domain sort F (for 'Formula'). That means, there are quantifications $\forall p{:}F \ldots$. It is well known that sorts can be written as one–place predicates. Instead of $\forall p{:}F \ldots$, one writes $\forall p\ F(p) \Rightarrow \ldots$. Now we assume this syntactic form for $\mathcal{A}_3$ and extend the ordering by requiring that literals $F(f(...))$ are bigger than any other literals. Since the first rewrite step eliminates all occurrences of connectives in $\mathcal{A}_3$, there are of course no literals of the form $F(f(\ldots))$ in $\mathcal{A}_3$, but only literals of the form $F(p)$. Each resolution between $w \models^{\mathcal{L}} f(\vec{p})$ and $v \models^{\mathcal{L}} p$ now instantiates p to $f(\ldots)$. That means, the resolvent contains literals $F(f(\ldots))$ and is therefore bigger than the parent clauses. This resolvent is redundant and needs not be generated.
Thus, we find that all resolvents with the transformers are redundant, which finally licenses their deletion. $\lhd$

Soundness and completeness of $\mathcal{T}_2$ together with Lemma 4.1 gives us the second completeness lemma.

Lemma 4.3 (Completeness Lemma for Neighbourhood Relations)
If $\mathcal{A}_1$ implies reflexivity and transitivity of $\models^2$ and the congruence properties (17) for all connectives then for every formula φ: $\mathcal{A}_1$ entails φ iff $\mathcal{A}_3 \stackrel{\text{def}}{=} \mathcal{T}_2(\mathcal{T}_1(\mathcal{A}_1))$ entails $\mathcal{T}_2(\mathcal{T}_1(\varphi))$. $\lhd$

4.2.1 Sets as Neighbourhoods

The transformer (24) is not yet recognizable as neighbourhood semantics we are familiar with from textbooks. The next transformation therefore combines the relations N and R^i into one neighbourhood relation N' with sets as arguments. To this end we define a K-transformation $\mathcal{Q}$ by

$$\begin{array}{ll} & \exists\vec{x}{:}S\ N_f(w,\vec{x}) \land \bigwedge_i(\forall v{:}W\ R^i_f(x_i,v) \Leftrightarrow \varphi(v,p_i)) \\ \Leftrightarrow & \\ & \exists\vec{X}\ N'_f(w,\vec{X}) \land \bigwedge_i(\forall v{:}W\ v \in X_i \Leftrightarrow \varphi(v,p_i)) \end{array} \tag{27}$$

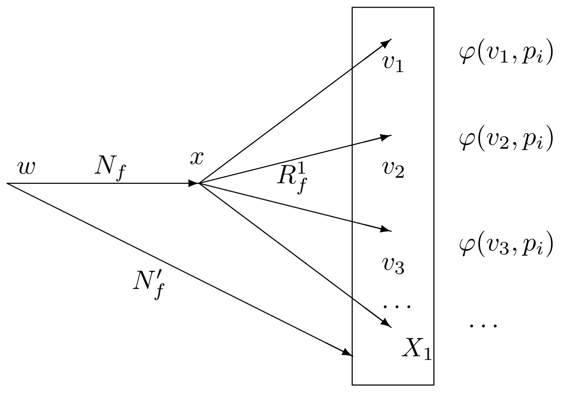
where $\varphi(v,p_i)$ is any formula containing v and p_i as free variables and $\in$ is the membership relation.

Lemma 4.4 (Soundness and Completeness of $\mathcal{Q}$)
If $\mathcal{A}_1$ implies reflexivity and transitivity of $\models^2$ together with the congruence properties (17) for all connectives, and the transformation strategy to be applied to $\mathcal{A}_3$ is rewriting inside–out using $\mathcal{Q}$ as rewrite rule from left to right, then the transformation lemmas hold for $\mathcal{Q}$.

Proof: Extension Lemma: The extension Lemma 2.4 for $\mathcal{Q}$ can be proved by defining the N'_f according to

$$N'_f(w,\vec{X}) \Leftrightarrow \exists\vec{x}\ N_f(w,\vec{x}) \land \bigwedge_i(\forall v\ x_i \in X_i \Leftrightarrow R^i_f(x_i,v))$$

The sets X_i are just the set of R^i_f–accessible points from some N_f–accessible point as the figure below illustrates for a single neighbourhood set.



This definition of N'_f entails the transformer (27).

Transformation Lemma: Since the transformer $\mathcal{T}_2$ generates formulae with exactly the structure of the left hand side of (27), the transformation strategy can simply be rewriting, starting with the innermost occurrences of the pattern. All instances of the left hand side are replaced by the corresponding instances of the right hand side. This is an equivalence preserving transformation. It eliminates the predicates N_f and R^i_f completely.

Elimination Lemma: In order to prove the elimination lemma we interpret N_f as N'_f and R^i_f as membership relation. This makes (27) a valid formula. $\lhd$

The composition of the two transformers $\mathcal{T}_2$ (24) and $\mathcal{Q}$ (27) is

$$\mathcal{T}_3{:} \qquad w \models^{\mathcal{L}} f(\vec{p}) \Leftrightarrow \exists\vec{X}\ N'_f(w,\vec{X}) \land \bigwedge_i X_i = |p_i| \tag{28}$$

where $X_i = |p_i|$ is an abbreviation for $\forall v\ v \in X_i \Leftrightarrow v \models^{\mathcal{L}} p_i$.

The instances of (28) for $\rightsquigarrow$ and $\Box$ in our test example are:

$$w \models^{\mathcal{C}} (p \rightsquigarrow q) \quad\Leftrightarrow\quad \exists X, Y\ N'_{\rightsquigarrow}(w, X, Y) \wedge X = |p| \wedge Y = |q| \tag{29}$$

$$w \models^{\mathcal{C}} \Box p \quad\Leftrightarrow\quad \exists X\ N'_{\Box}(w, X) \wedge X = |p|. \tag{30}$$

In (30) we recognize the standard definition of neighbourhood semantics for the modal $\Box$–operator. (29) is the generalization of neighbourhood semantics to an implication like connective.
Soundness and completeness of $\mathcal{T}_3$ together with Lemma 4.3 yields the next completeness result.

Lemma 4.5 (Completeness Lemma for Sets as Neighbourhoods)
If $\mathcal{A}_1$ implies reflexivity and transitivity of $\models^2$ and the congruence properties (17) for all connectives then for every formula φ: $\mathcal{A}_1$ entails φ iff $\mathcal{A}_4 \stackrel{\text{def}}{=} \mathcal{T}_3(\mathcal{T}_1(\mathcal{A}_1))$ entails $\mathcal{T}_3(\mathcal{T}_1(\varphi))$. $\triangleleft$

4.2.2 Binary Neighbourhood Relations

The transformer $\mathcal{Q}$ is only a structural modification, more or less syntactic sugar. An alternative to $\mathcal{Q}$ is the transformer $\mathcal{P}$. It reduces the n-ary neighbourhood relation to a binary relation and it breaks the equivalence into two implications. $\mathcal{P}$ is defined for all $\vec{p}$ and w as

$$\begin{array}{ll} & \exists \vec{x}{:}S\ N_f(w, \vec{x}) \wedge \bigwedge_i \forall v{:}W\ R^i_f(x_i, v) \Leftrightarrow \varphi_i(v, p_i) \\ \Leftrightarrow & \\ & \exists x{:}S' K_f(w, x) \wedge \bigwedge_i (\forall v{:}W\ R^{i+}_f(x_i, v) \Rightarrow \varphi_i(v, p_i)) \wedge (\forall v{:}W\ R^{i-}_f(x_i, v) \Rightarrow \neg\varphi_i(v, p_i)). \end{array} \tag{31}$$

Lemma 4.6 (Soundness and Completeness of $\mathcal{P}$)
If (i) $\mathcal{A}_1$ implies that $\models^2$ is reflexive and transitive and (ii) $\mathcal{A}_1$ implies the congruence properties (17) for all connectives and (iii) the transformation strategy to be applied to $\mathcal{A}_3$ is inside–out rewriting, using $\mathcal{P}$ as rewrite rule from left to right, then the transformation lemmas hold for $\mathcal{P}$.

Proof: Extension Lemma: To prove the extension lemma we have to define for a given interpretation $\Im$ an extension $\Im'$ with an interpretation for the new sort S' and the new relations K_f and R^{i+}_f such that (31) is true. For the new sort S' we define $\Im(S') \stackrel{\text{def}}{=} \Im(W)^n$, i.e. the set of n-tuples of worlds. For K we define $(w, (\vec{x})) \in \Im'(K)$ iff $(w, \vec{x}) \in \Im(N_f)$. Finally $((\vec{x}), v) \in \Im'(R^{i+}_f)$ iff $(x_i, v) \in \Im(R^i_f)$ and $((\vec{x}), v) \in \Im'(R^{i-}_f)$ iff $(x_i, v) \notin \Im(R^i_f)$. It is easy to show that these definitions satisfy (31).

Transformation Lemma: The transformation strategy is definitional replacement, which is an equivalence transformation. In $\mathcal{T}_2(\mathcal{T}_1(\mathcal{A}_1))$ the formulae all have the structure of the left hand side of (31). These formulae can be replaced by the corresponding instances of the right hand side. Equivalence transformations preserve models.

Elimination Lemma: To exhibit the elimination lemma we define for a given interpretation $\Im$ of the transformed system an extension $\Im'$ with an interpretation for the old sort S and the old relations N and R^i_f such that (31) again becomes true. We define $\Im'(S) \stackrel{\text{def}}{=} 2^{\Im(W)}$, $\Im'(R^i_f) \stackrel{\text{def}}{=} \ni$ and $(w, \vec{X}) \in \Im'(N_f)$ iff $X_i \supseteq R^{i+}_f(x)$ and $X_i \cap R^{i-}_f(x) = \emptyset$ for some $x \in \Im(S')$, where $R^{i+}_f(x) \stackrel{\text{def}}{=} \{y \mid (x, y) \in \Im(R^{i+}_f)\}$.
That means, the elements of $\Im'(S)$ are sets of worlds and the neighbourhoods of w are all the supersets of R^{i+}_f–accessible worlds which do not intersect with the set of R^{i-}_f–accessible worlds.
To show that this interpretation satisfies (31), first suppose the right hand side of (31) is true for some w and $\vec{p}$. That means there is some x with $R^{i+}_f(x) \subseteq |p_i|$ and $R^{i-}_f(x) \subseteq |p_i|^c$, where $|p_i| \stackrel{\text{def}}{=} \{v \mid \varphi(v, p_i)$ is true in $\Im'\}$, and $|p_i|^c$ is the complement of $|p_i|$. According to the construction of $\Im'(N_f)$, for all supersets X_i of $|p_i|$ which do not intersect with $|p_i|^c$, $N_f(w, \vec{X})$ holds. In particular this is true for $X_i = |p_i|$ and therefore the left hand side of (31) is also true.
Now assume the left hand side of (31) holds, i.e. there is some $\vec{X}$ with $N_f(w, \vec{X})$ and $X_i = |p_i|$. From the construction of $\Im'(N_f)$ we know there is some x with $K_f(w, x)$ and $R^{i+}_f(x) \subseteq X_i$ and $X_i \cap R^{i-}_f(x) = \emptyset$. Therefore $R^{i-}_f(x) \subseteq |p_i|^c$ and thus the right hand side of (31) is also true. $\triangleleft$

Composing the transformers $\mathcal{T}_2$ and $\mathcal{P}$ yields the new transformer $\mathcal{B}_1$ (for 'Binary'). For all w and $\vec{p}$:

$$w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{p}) \Leftrightarrow \exists x{:}S'\ K_f(w,x) \wedge \bigwedge_i (\forall v{:}W\ R_f^{i+}(x_i,v) \Rightarrow v \models p_i) \wedge (\forall v{:}W\ R_f^{i-}(x_i,v) \Rightarrow \neg v \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, p_i). \quad (32)$$

As before, soundness and completeness of $\mathcal{P}$ together with Lemma 4.3 yields a further completeness result.

Lemma 4.7 (Completeness Lemma for the $\mathcal{B}_1$ Transformation)
If $\mathcal{A}_1$ implies reflexivity and transitivity of $\Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}$ and the congruence properties (17) for all connectives then for every formula φ: $\mathcal{A}_1$ entails φ iff $\mathcal{B}_1(\mathcal{T}_1(\mathcal{A}_1))$ entails $\mathcal{B}_1(\mathcal{T}_1(\varphi))$. $\lhd$

The transformation $\mathcal{B}_1$ (32) is identical with a transformation of a multi–modal logic formula $\langle K_f\rangle(\bigwedge_i([R_f^{i+}]p_i \wedge [R_f^{i+}]\neg p_i))$ using the transformer $\mathcal{T}_6$, defined in (58) below, for a corresponding number of normal unary modal operators. That means, there is a translation

$$\pi_1(f(\vec{p})) = \langle K\rangle(\textstyle\bigwedge_i([R_f^{i+}]\pi_1(p_i) \wedge [R_f^{i+}]\neg\pi_1(p_i))) \quad (33)$$

from the given logic $\mathcal{L}$ into a normal multi–modal logic. $\langle ...\rangle$ is the parameterized $\Diamond$–operator and [...] is the parameterized $\Box$–operator. This translation is further investigated in Section 4.6.

4.3 Monotonicity Properties

The congruence properties (17) can be strengthened in many cases to *monotonicity properties*:

$$\text{upward monotonicity:} \quad \forall p,q\ p \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, q \Rightarrow \forall \vec{s},\vec{s}'\, f(\vec{s},p,\vec{s}') \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, f(\vec{s},q,\vec{s}') \quad (34)$$
$$\text{downward monotonicity:} \quad \forall p,q\ p \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, q \Rightarrow \forall \vec{s},\vec{s}'\, f(\vec{s},q,\vec{s}') \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, f(\vec{s},p,\vec{s}'). \quad (35)$$

For example, for our test set the implication connective $\rightsquigarrow$ is downward monotonic in the first argument and upward monotonic in the second argument and the $\Box$–operator is upward monotonic. This means,

$$p \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, q \Rightarrow \forall s\ (s \rightsquigarrow p) \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, (s \rightsquigarrow q) \quad (36)$$
$$p \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, q \Rightarrow \forall s\ (q \rightsquigarrow s) \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, (p \rightsquigarrow s) \quad (37)$$
$$p \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, q \Rightarrow \Box p \Vdash\!\!\raisebox{1ex}{$\scriptstyle 2$}\, \Box q \quad (38)$$

follow from $\mathcal{A}_1$. ((38) is the so–called MN rule.). (36) is proved in Lemma A.6, (37) is proved in Lemma A.7, and (38) is proved in Lemma A.8.

The monotonicity properties (34) and (35) are translated by $\mathcal{T}_1$ into

upward monotonicity:
$$(\forall w{:}W\ w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, p \Rightarrow w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, q) \Rightarrow \forall \vec{s},\vec{s}'\ \forall w{:}W\ w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{s},p,\vec{s}') \Rightarrow w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{s},q,\vec{s}') \quad (39)$$
downward monotonicity:
$$(\forall w{:}W\ w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, p \Rightarrow w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, q) \Rightarrow \forall \vec{s},\vec{s}'\ \forall w{:}W\ w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{s},q,\vec{s}') \Rightarrow w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{s},p,\vec{s}'). \quad (40)$$

If $f(\vec{p},\vec{q})$ is downward monotonic in the first arguments $\vec{p}$ and upward monotonic in the last arguments $\vec{q}$ (we shall assume this ordering of the argument positions from now on) one can prove from these two formulae for all $\vec{p},\vec{q},w$

$$w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{p},\vec{q}) \Leftrightarrow \forall \vec{x} \textstyle\bigwedge_i (\forall v{:}W\ v \models x_i \Rightarrow v \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, p_i) \Rightarrow \exists \vec{y} \bigwedge_j (\forall v{:}W\ v \models y_j \Rightarrow v \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, q_j) \wedge w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{x},\vec{y}).$$

The proof is given in Lemma A.9 of the Appendix. This equivalence can be used as the basis for a more stronger K-transformation than $\mathcal{T}_2$. The new transformer $\mathcal{T}_4$ is

$$\begin{aligned} w \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, f(\vec{p},\vec{q}) \quad \Leftrightarrow \quad & \forall \vec{x}{:}S\ \textstyle\bigwedge_i (\forall v{:}W\ R_f^i(x_i,v) \Rightarrow v \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, p_i) \Rightarrow \\ & \exists \vec{y}{:}S\ \textstyle\bigwedge_j (\forall v{:}W\ R_f^j(y_j,v) \Rightarrow v \models\!\!\!\raisebox{1ex}{$\scriptstyle\sqsubseteq$}\, q_j) \wedge N_f(w,\vec{x},\vec{y}). \end{aligned} \quad (41)$$

The soundness and completeness proof for this transformer is identical to the proof for $\mathcal{T}_2$ (Lemma 4.2).

The instances of (41) for our test connectives are

$$\begin{aligned} w \models (p \rightsquigarrow q) \quad &\Leftrightarrow \quad \forall x{:}S\ (\forall v{:}W\ \mathcal{R}^1_{\rightsquigarrow}(x,v) \Rightarrow v \models p) \Rightarrow \\ &\qquad\qquad \exists y{:}S\ (\forall v{:}W\ \mathcal{R}^2_{\rightsquigarrow}(y,v) \Rightarrow v \models q) \wedge N_{\rightsquigarrow}(w,x,y) \\ w \models \Box q \quad &\Leftrightarrow \quad \exists y{:}S\ (\forall v{:}W\ \mathcal{R}_{\Box}(y,v) \Rightarrow v \models q) \wedge N_{\Box}(w,y). \end{aligned}$$

Just as we derived variations of the neighbourhood relation for the congruence properties, we can derive the corresponding variations of the neighbourhood relation for the monotonicity properties. The variant with sets as arguments of the neighbourhood relation which corresponds to $\mathcal{T}_3$ is

$$\mathcal{T}_5\text{:} \qquad w \models f(\vec{p},\vec{q}) \Leftrightarrow \forall \vec{X}\ \vec{X} \subseteq |\vec{p}| \Rightarrow \exists \vec{Y}\vec{Y} \subseteq |\vec{q}| \wedge N'_f(w,\vec{X},\vec{Y}). \tag{42}$$

A version with binary neighbourhood relation which corresponds to $\mathcal{B}_1$ seems not possible in general. Only for connectives which are either upward monotonic or downward monotonic in all their arguments, a corresponding transformation is possible. For the upward monotonic version the K-transformation

$$\begin{aligned} &\exists \vec{y}{:}S\ \textstyle\bigwedge_i(\forall v{:}W\ R^i_f(y_i,v) \Rightarrow \varphi(v,q_i)) \wedge N_f(w,\vec{y}) \\ \Leftrightarrow\ & \\ &\exists y{:}S' K_f(w,y) \wedge \textstyle\bigwedge_i(\forall v{:}W\ R'^i_f(y,v) \Rightarrow \varphi(v,q_i)) \end{aligned}$$

is sounds and complete. This is because the n arguments $\vec{y}$ of N_f can be replaced with the one argument y of K_f by interpreting y as the n–tuple of the values assigend to $\vec{y}$ and by interpreting the R'^i_f–relations as projection functions which extract the i^{th} component of this n–tuple. Combined with $\mathcal{T}_4$ (41), we obtain a transformer

$$\mathcal{B}_2^+\text{:} \qquad w \models f(\vec{p})\ \Leftrightarrow \exists y{:}S' K_f(w,y) \wedge \textstyle\bigwedge_i(\forall v{:}W\ R'^i_f(y,v) \Rightarrow v \models q_i)$$

which in turn gives rise to a translation

$$\pi_2^+(f(\vec{p})) = \langle K_f\rangle(\textstyle\bigwedge_i[R'^i_f]\pi_2^+(p_i)) \tag{43}$$

into a multi–modal logic. For the downward monotonic version the K-transformation is

$$\begin{aligned} &\forall \vec{y}{:}S\ \textstyle\bigwedge_i(\forall v{:}W\ R^i_f(x_i,v) \Rightarrow \varphi_i(v,p_i)) \Rightarrow N_f(w,\vec{y}) \\ \Leftrightarrow\ & \\ &\forall y{:}S' K_f(w,y) \Rightarrow \textstyle\bigvee_i(\exists v{:}W\ R'^i_f(y,v) \wedge \neg\varphi_i(v,p_i)). \end{aligned}$$

The right hand side of the equivalence is the contrapositive of the left hand side. $K_f(w,y)$ is interpreted this time as $\neg N_f(w,\vec{y})$. Again y represents the n-tuple $\vec{y}$. Combined with $\mathcal{T}_4$, we obtain a transformer

$$\mathcal{B}_2^-\text{:} \qquad w \models f(\vec{p})\ \Leftrightarrow\ \forall y{:}S'\ K_f(w,y) \Rightarrow \textstyle\bigvee_i(\exists v{:}W\ R'^i_f(y,v) \wedge \neg v \models p_i) \tag{44}$$

which this time gives rise to a translation

$$\pi_2^-(f(\vec{p})) = [K_f](\textstyle\bigvee_i \langle R'^i_f\rangle \neg\pi_2^-(p_i))$$

into a multi–modal logic. Finally, we obtain the completeness lemma

Lemma 4.8 (Completeness Lemma for the Monotonic Case)
If (i) $\mathcal{A}_1$ implies that $\models^2$ is reflexive and transitive, and (ii) $\mathcal{A}_1$ implies the congruence properties (17) for *all* connectives and the monotonicity properties for some connectives then for every formula φ: $\mathcal{A}_1$ entails φ iff $\mathcal{A}$ entails φ', where $\mathcal{A}$ and φ' are transformed by the strongest transformers, i.e. the transformers for monotonicity if possible, otherwise the transformer for the congruence relations. They can be mixed arbitrarily. $\lhd$

4.4 Transformations Based on Stone's Representation Theorem

The transformations we consider in this section correspond to the transition from neighbourhood semantics to standard relational Kripke semantics in modal logic. Unfortunately this transition requires the relatively strong assumption that the logic defined by $\mathcal{A}_1$ is an extension of standard propositional logic. For propositional logic it is known that its Lindenbaum algebra is a Boolean algebra, and Boolean algebras are isomorphic to fields of sets. As we shall see, this enables us to take quantifications $\forall p\ \ldots w \models^{\mathcal{L}} p \ldots$ as quantifications over sets $\forall p\ \ldots w \in p \ldots$ or their characteristic predicates $\forall p\ \ldots p(w) \ldots$ respectively. With this assumption a semantics in the usual way is obtained and many properties of the system can be simplified considerably.

Lemma 4.9 (Congruence Relation)
If $\mathcal{A}_1$ entails the reflexivity and transitivity of $\models^{\mathcal{L}}$ and the congruence properties (17) hold for *all* connectives f then the relation $p \approx q \stackrel{\text{def}}{=} (\mathcal{A}_1$ entails $(p \vdash^{2} q \wedge q \vdash^{2} p))$, is a congruence relation on $\mathcal{L}$–terms.

Proof: $\approx$ is reflexive and transitive because $\vdash^{2}$ is. It is symmetric by definition. It is a congruence relation because we assumed that the congruence properties of $\vdash^{2}$ hold for *all* connectives. $\lhd$

For a term s let $[s]$ be its $\approx$–congruence class. $\mathcal{L}_{|\approx} \stackrel{\text{def}}{=} \{[s] \mid s \text{ is an } \mathcal{L}\text{–term}\}$ is called the *Lindenbaum algebra* of $\mathcal{L}$. If $\mathcal{L}$ contains propositional logic, *its Lindenbaum algebra is a Boolean algebra.*

Definition 4.10 (Transition to PL2)
The transformation $\mathcal{S}$ replaces all occurrences of literals of the form $w \models^{\mathcal{L}} p$, where p is a variable or constant symbol with a literal $p(w)$. Quantifications $\forall p\ \ldots w \models^{\mathcal{L}} p \ldots$ are turned into quantifications of the form $\forall p\ \ldots p(w) \ldots$ and likewise for existential quantifications. $\lhd$

Theorem 4.11 (Soundness and Completeness of $\mathcal{S}$)
If

1. $\mathcal{A}_1$ contains an axiomatization of standard propositional logic (this implies that $\models^{\mathcal{L}}$ is reflexive and transitive),
2. $\mathcal{A}_1$ entails the congruence properties (17) for *all* connectives f,
3. $\mathcal{T} \neq \mathcal{T}_1$ is any of the sound and complete transformations defined above and
4. φ is a formula with $\vdash^{2}$ as the only predicate and whose conjunctive normal form contains at least one positive literal in each conjunct (we say φ is *positive*)

then, $\mathcal{A}_1$ entails φ iff $\mathcal{S}(\mathcal{T}(\mathcal{T}_1(\mathcal{A}_1)))$ entails $\mathcal{S}(\mathcal{T}(\mathcal{T}_1(\varphi)))$.

Proof: <u>**Case 1:**</u> $\mathcal{A}_1$ entails $\forall p, q\ p \vdash^{2} q$.
Since at least one literal of φ's conjunctive normal form is positive, i.e. it has the form $s \vdash^{2} t$, φ is obviously true, given $\mathcal{A}_1$. By the previous completeness theorems, $\mathcal{T}(\mathcal{T}_1(\mathcal{A}_1))$ entails $\alpha \stackrel{\text{def}}{=} \forall p, q, w\ w \models^{\mathcal{L}} p \Rightarrow w \models^{\mathcal{L}} q$, i.e. $\mathcal{T}(\mathcal{T}_1(\mathcal{A}_1)) \Leftrightarrow \mathcal{T}(\mathcal{T}_1(\mathcal{A}_1)) \wedge \alpha$. Since $\mathcal{S}(\alpha) = (\forall p, q, w\ p(w) \Rightarrow q(w))$ and this is a contradiction, $\mathcal{S}(\mathcal{T}(\mathcal{T}_1(\mathcal{A}_1)))$ entails everything, in particular it entails φ.
<u>**Case 2:**</u> $\mathcal{A}_1$ is consistent with $\beta \stackrel{\text{def}}{=} \exists p, q\ \neg(p \vdash^{2} q)$.
Thus, $\mathcal{A}_1$ entails φ iff $\mathcal{A}_1 \wedge \beta$ entails φ *and* $\mathcal{A}_1 \wedge \neg\beta$ entails φ. The latter case is actually already considered in case 1. Therefore, w.l.o.g. we can assume that $\mathcal{A}_1$ entails β. By the completeness theorems for $\mathcal{T}$ we have, $\mathcal{A}_1$ entails φ iff $\mathcal{T}(\mathcal{T}_1(\mathcal{A}_1))$ entails $\mathcal{T}(\mathcal{T}_1(\varphi))$. Let $A' \stackrel{\text{def}}{=} \mathcal{T}(\mathcal{T}_1(\mathcal{A}_1))$ and $\varphi' \stackrel{\text{def}}{=} \mathcal{T}(\mathcal{T}_1(\varphi))$. We need to show, A' entails φ' iff $\mathcal{S}(A')$ entails $\mathcal{S}(\varphi')$. By assumption 1 and Lemma 4.9, $\{[p] \mid p \text{ is an } \mathcal{L}\text{-term}\}$ is a Boolean algebra. We have

$p \approx q$	iff	$\mathcal{A}_1$ entails $p \vdash^{2} q \wedge q \vdash^{2} p$
	iff	$\mathcal{A}_2$ entails $\forall w{:}W\ w \models^{\mathcal{L}} p \Leftrightarrow w \models^{\mathcal{L}} q$
	iff	in every interpretation $\Im$ of $\mathcal{A}_2$: $\lvert p\rvert = \lvert q\rvert$
		where $\lvert p\rvert = \{w \mid (w, \Im(p)) \in \Im(\models^{\mathcal{L}})\}$.

Thus, $q \in [p]$ iff for every interpretation $\Im$ of A', $|p| = |q|$. Hence, $\{|p| \mid p \in \mathit{domain}(\Im)\}$ is a Boolean algebra for every interpretation of A'. By Stone's representation theorem [Sto36], every Boolean algebra is isomorphic to a field of sets. Therefore $\{|p| \mid p \in \mathit{domain}(\Im)\}$ is isomorphic to the powerset of some base set. Because of the assumption β the Boolean algebra contains at least two elements, the base set of the corresponding powerset is therefore not empty.
Since the transformations $\mathcal{S}(A')$ and $\mathcal{S}(\varphi')$ affect only literals of the form $\ldots \models^{\subseteq} p$, where p is a variable or constant symbol, and p occurs only as the second argument of the $\models^{\subseteq}$–predicate, there is an isomorphism between the interpretations of A' and $\mathcal{S}(A')$. By structural induction on the formulae it can be shown that this isomorphism guarantees that A' and $\mathcal{S}(A')$ entail the same theorems. $\lhd$

Notice that $\mathcal{S}$ can turn a consistent axiomatization into an inconsistent axiomatization when $\mathcal{A}_1$ entails $\forall p, q\ p \models^{2} q$. This is usually an undesired phenomenon. Therefore one should check whether this formula is entailed by $\mathcal{A}_1$ or not. A model generation program like John Slaney's FINDER can do this automatically [PS90].

There is a weaker version $\mathcal{S}'$ of $\mathcal{S}$. $\mathcal{S}'$ replaces literals $w \models^{\subseteq} p$ with $w \in p$. Since the formula $\forall p, q\ \forall w\ w \in p \Rightarrow w \in q$ is consistent, whereas $\forall p, q\ \forall w\ p(w) \Rightarrow q(w)$ is inconsistent, there is a slight difference between $\mathcal{S}'$ and $\mathcal{S}$. If $\mathcal{A}_1$ does not entail $\forall p, q\ p \models^{2} q$, however, these two are equivalent. In the following we shall make this assumption and use $\mathcal{S}'$ and $\mathcal{S}$ both interchangeably.

$p \models^{2} q$ is transformed by $\mathcal{T}_1$ into $\forall w\ w \models^{\subseteq} p \Rightarrow w \models^{\subseteq} q$, and this in turn is transformed by $\mathcal{S}'$ into $\forall w\ w \in p \Rightarrow w \in q$ which means $p \subseteq q$. Thus, from now on the $\models^{2}$–relation can be interpreted as subset relation. As further consequences, we need no longer the notations $w \in |p|$ and $|p| = |q|$ etc. for *atomic* p's and q's.

The $|...|$–function is no longer necessary. $w \in |p|$ becomes $w \in p$ and $|p| = |q|$ becomes $p = q$, where p and q now denote sets. This allows considerable simplifications. For example, the formula $\mathcal{T}_3(f(\vec{p})) = \exists \vec{X}\ N'_f(w, \vec{X}) \wedge \bigwedge_i X_i = |p_i|$ simplifies to just $N'_f(w, \vec{p})$.

Example 4.12 (for Transformations with $\mathcal{S}$)
As a first example, suppose $\mathcal{A}_1$ contains $\gamma \stackrel{\text{def}}{=} \forall p\ \Box p \models^{2} \neg p$ where $\neg$ is classical negation and $(w \models \neg p) \Leftrightarrow \neg(w \models p)$ has been verified already.

$$\begin{array}{lcl}
\gamma' & \stackrel{\text{def}}{=} & \mathcal{T}_3(\mathcal{T}_1(\gamma)) = \forall p, w\ (\exists X N'_\Box(w, X) \wedge X = |p|) \Rightarrow \neg w \in |p|. \\
\mathcal{S}'(\gamma') & = & \forall p, w\ N'_\Box(w, p) \Rightarrow w \notin p. \\
\gamma'' & \stackrel{\text{def}}{=} & \mathcal{T}_5(\mathcal{T}_1(\gamma)) = \forall p, w\ (\exists X X \subseteq |p| \wedge N'_\Box(w, X)) \Rightarrow \neg w \in |p|. \\
\mathcal{S}(\gamma'')) & = & \forall p, w\ (\exists X (\forall v\ v \in X \Rightarrow p(v)) \wedge N'_\Box(w, X)) \Rightarrow \neg p(w).
\end{array}$$

This last formula is a PL2 formula which is equivalent to the PL1 formula $\forall w, X\ \neg N'_\Box(w, X)$ (try the SCAN algorithm on it). Thus, under the monotonicity assumption of $\Box$ (we used $\mathcal{T}_3$), $\Box p \models^{2} \neg p$ enforces that no world has any neighbourhood at all, not even the empty set.

As a second example, we take the monotonicity properties (34) and (35). The composition of the transformers $\mathcal{T}_1$, $\mathcal{T}_3$ and $\mathcal{S}$ translates these two formulae into:

$$\text{upward monotonicity:} \quad \forall p, q\ p \subseteq q \Rightarrow \forall \vec{s}, \vec{s}' \forall w{:}W\ N'_f(w, \vec{s}, p, \vec{s}') \Rightarrow N'_f(w, \vec{s}, q, \vec{s}') \tag{45}$$

$$\text{downward monotonicity:} \quad \forall p, q\ p \subseteq q \Rightarrow \forall \vec{s}, \vec{s}' \forall w{:}W\ N'_f(w, \vec{s}, q, \vec{s}') \Rightarrow N'_f(w, \vec{s}, p, \vec{s}'). \tag{46}$$

Since p and q are variables, we could use again the simpler notation $N(w, p)$ instead of $\exists X\ N(w, X)$ $\wedge X = |p|$ which these transformers initially produce.
Upward monotonicity thus enforces for the neighbourhood relation closedness under supersets, and downward monotonicity enforces closedness under subsets. $\lhd$

4.5 Closure Properties

In modal logic, relational possible worlds semantics can be obtained from neighbourhood semantics if the neighbourhood structure is closed under supersets *and intersections*. We generalize this result now for arbitrary connectives. The key formula which must hold in order to strengthen neighbourhood semantics to relational semantics in modal logic is $(\Box q_1 \wedge \Box q_2) \Rightarrow \Box(q_1 \wedge q_2)$. A slightly weaker property which can be formulated without the $\wedge$–connective and which is sufficient for this purpose is

$$\forall q_1, q_2, w\ w \Vdash^2 \Box q_1 \wedge w \Vdash^2 \Box q_2 \Rightarrow \exists s\ s \Vdash^2 q_1 \wedge s \Vdash^2 q_2 \wedge w \Vdash^2 \Box s. \tag{47}$$

The s replaces $q_1 \wedge q_2$ or something stronger respectively.

In line with the previous sections where we tried to find transformations eliminating particular properties of the consequence relation and connectives f, the properties we want to eliminate now, are for each upward monotonic argument position of f
(to simplify notation we assume this is the last argument position):

$$\forall \vec{p}, q_1, q_2, w\ w \Vdash^2 f(\vec{p}, q_1) \wedge w \Vdash^2 f(\vec{p}, q_2) \Rightarrow \exists s\ s \Vdash^2 q_1 \wedge s \Vdash^2 q_2 \wedge w \Vdash^2 f(\vec{p}, s) \tag{48}$$

and for each downward monotonic argument position of f
(to simplify notation we assume this is the first argument position):

$$\forall p_1, p_2, \vec{q}, w\ w \Vdash^2 f(p_1, \vec{q'}) \wedge w \Vdash^2 f(p_2, \vec{q}) \Rightarrow \exists s\ p_1 \Vdash^2 s \wedge p_2 \Vdash^2 s \wedge w \Vdash^2 f(s, \vec{q}). \tag{49}$$

The instance of (48) for the $\Box$–operator is (47). The instance of (48) for the $\rightsquigarrow$–connective is (50) and the instance of (49) for the $\rightsquigarrow$–connective is (51) below.

$$\forall p, q_1, q_2, w\ w \Vdash^2 (p \rightsquigarrow q_1) \wedge w \Vdash^2 (p \rightsquigarrow q_2) \Rightarrow \exists s\ s \Vdash^2 q_1 \wedge s \Vdash^2 q_2 \wedge w \Vdash^2 (p \rightsquigarrow s) \tag{50}$$

$$\forall p_1, p_2, q, w\ w \Vdash^2 (p_1 \rightsquigarrow q) \wedge w \Vdash^2 (p_2 \rightsquigarrow q) \Rightarrow \exists s\ p_1 \Vdash^2 s \wedge p_2 \Vdash^2 s \wedge w \Vdash^2 (s \rightsquigarrow q). \tag{51}$$

(50) is proved in Lemma A.10. (51) is proved in Lemma A.11 and (47) is proved in Lemma A.12 in the Appendix.

Induction on natural numbers shows that from the closure properties (48) and (49) the stronger versions with finite premises

$$\forall q_1, \ldots, q_m, \vec{p}, w\ (\textstyle\bigwedge_{i=1}^m w \Vdash^2 f(\vec{p}, q_i)) \Rightarrow \exists s\ (\textstyle\bigwedge_{i=1}^m s \Vdash^2 q_i) \wedge w \Vdash^2 f(\vec{p}, s) \tag{52}$$

$$\forall p_1, \ldots, p_m, \vec{q}, w\ (\textstyle\bigwedge_{i=1}^m w \Vdash^2 f(p_i, \vec{q})) \Rightarrow \exists s\ (\textstyle\bigwedge_{i=1}^m p_i \Vdash^2 s) \wedge w \Vdash^2 f(s, \vec{q}) \tag{53}$$

follow.

Lemma 4.13 (Finite Closure Properties of the Neighbourhood Relation)
If $\mathcal{A}_1$ is an extension of propositional logic then the closure formula (52) for the k^{th} argument together with the upward monotonicity for the same argument implies closedness under finite intersections of the corresponding argument position in the neighbourhood relation. Conversely, (53) and downward monotonicity implies closedness under finite unions.

Proof: We prove the first part. The proof for closedness under unions is analogous. Since the q_i and p_i are variables in (52), the composition of the transformers $\mathcal{T}_1$, $\mathcal{T}_3$ and $\mathcal{S}$, applied to (52) yields a formula which is equivalent to

$$\forall q_1, \ldots, q_m, \vec{p}, w\ \textstyle\bigwedge_{i=1}^m N'_f(w, \vec{p}, q_i) \Rightarrow \exists s\ (\textstyle\bigwedge_{i=1}^m s \subseteq q_i) \wedge N'_f(w, \vec{p}, s).$$

This in turn is equivalent to

$$\forall q_1, \ldots, q_m, \vec{p}, w\ \textstyle\bigwedge_{i=1}^m N'_f(w, \vec{p}, q_i) \Rightarrow \exists s\ s \subseteq \textstyle\bigcap_{i=1}^m q_i \wedge N'_f(w, \vec{p}, s)$$

and since the monotonicity formulae imply that the neighbourhood relation is closed under supersets in the 's-argument' (see Example 4.12), we obtain

$$\forall q_1, \ldots, q_m, \vec{p}, w\ \textstyle\bigwedge_{i=1}^m N'_f(w, \vec{p}, q_i) \Rightarrow N'_f(w, \vec{p}, \textstyle\bigcap_{i=1}^m q_i),$$

i.e. closedness under finite intersections. A similar transformation of (53) yields

$$\forall p_1, \ldots, p_m, \vec{q}, w\ \textstyle\bigwedge_{i=1}^m N'_f(w, p_i, \vec{q}) \Rightarrow N'_f(w, \textstyle\bigcup_{i=1}^m p_i, \vec{q}).$$

i.e. closedness under finite unions. $\lhd$

Unfortunately closedness under finite unions and intersections is not sufficient for our next transformation step. We need closedness under arbitrary intersections and unions respectively. As the lemma below shows, this can be assumed in fact, provided $\mathcal{A}_1$ consists of definite Horn clauses.

Lemma 4.14 ((52) for $m = \infty$)
If $\mathcal{A}_1$ consists of definite Horn clauses (they have exactly one positive literal), and (52) holds for all finite m, and φ is a finite formula, then $\mathcal{A}_1$ entails φ iff $\mathcal{A}_1 \wedge \delta$ entails φ, where δ is (52) for $m = \infty$.

Proof: The 'if'–direction is trivial. For the 'only–if'–direction suppose $\mathcal{A}_1 \wedge \delta$ entails φ. It suffices to show that φ is true in all Herbrand interpretations of $\mathcal{A}_1$. Since $\mathcal{A}_1$ consists of definite Horn clauses, this can be reduced further. It is sufficient to show that φ holds in the unique minimal Herbrand interpretation $\Im_0$, which is the intersection of all Herbrand interpretations of $\mathcal{A}_1$. We now construct an extension of $\Im_0$ as a model for $\mathcal{A}_1 \wedge \delta$. Let $A_0 \stackrel{\text{def}}{=} \mathcal{A}_1$. For $i > 0$, we construct $\Im_i$ and A_i as follows. For each term w and set $\{p_1, \ldots, p_\infty\}$ of terms for which the premise of δ holds in $\Im_{i-1}$, but the conclusion does not hold, we create a new constant symbol s and add the corresponding conclusion literals of δ to A_{i-1}. A_i is the result of this extension. $\Im_i$ is the minimal Herbrand interpretation of A_i. $\Im_i$ is a superset of $\Im_{i-1}$.
Now let $A \stackrel{\text{def}}{=} \bigcup A_i$ and $\Im \stackrel{\text{def}}{=} \bigcup \Im_i$. A consists of $\mathcal{A}_1$ and infinitely many additional unit clauses. $\Im$ is the minimal Herbrand model of A and by construction, $\Im$ satisfies (52). Therefore A entails δ. Since A also contains $\mathcal{A}_1$, A entails φ. Because of the compactness of PL1, a finite subset of A' of A already entails φ. Suppose A' contains some of the additional unit clauses $s \Vdash^2 q_i$ or $w \Vdash^2 f(\vec{p}, s)$ for some of the newly created constant symbols s. Since $\mathcal{A}_1$ implies (52) for all finite m, there is also a term s' such that $s' \Vdash^2 q_i$ and $w \Vdash^2 f(\vec{p}, s')$ is entailed by $\mathcal{A}_1$. s does not occur in φ. Therefore the clauses with s can be replaced with the corresponding clauses with s'. In other words, a subset A' of $\mathcal{A}_1$ is sufficient to derive φ, and that's what we set out to show. $\lhd$

Analogously we can show that (53) for $m = \infty$ can be assumed either. As a corollary of the above lemma and the Lemma 4.13 we get:

Corollary 4.15 (Closure Properties of the Neighbourhood Relation)
If $\mathcal{A}_1$ satisfies all the conditions of Theorem 4.11 and additionally $\mathcal{A}_1$ consists of definite Horn clauses and (48) holds for the upward monotonic argument positions and (49) holds for the downward monotonic argument positions of f then closedness under arbitrary intersections, and closedness under arbitrary unions, respectively, holds:

$$(\textstyle\bigwedge_{i \in I} N'_f(w, \vec{X}, Y_i)) \Rightarrow N'_f(w, \vec{X}, \bigcap_{i \in I} Y_i)$$
$$(\textstyle\bigwedge_{i \in I} N'_f(w, X_i, \vec{Y})) \Rightarrow N'_f(w, \bigcup_{i \in I} X_i, \vec{Y}).$$

$\lhd$

Unfortunately for the proof we used that $\mathcal{A}_1$ consists of definite Horn clauses. It is not clear whether this is an artefact of the proof technique or whether this is necessary in general.

Lemma 4.16 (Accessibility Relation)
If the neighbourhoud relation N' is *serial* for at least one argument position, for example
$\forall w, \vec{X}\ \exists Y\ \ N'(w, \vec{X}, Y)$ holds, and N' is closed under subsets and union in the first group of arguments and closed under supersets and intersections in the last group of arguments in the sense of Corollary 4.15, then there exists a relation $\mathcal{R}$ on worlds with

$$N'(w, \vec{X}, \vec{Y}) \Leftrightarrow \forall \vec{x}, \vec{y}\ \mathcal{R}(w, \vec{x}, \vec{y}) \Rightarrow (\textstyle\bigwedge_i x_i \in X_i) \Rightarrow (\bigvee_j y_j \in Y_j). \tag{54}$$

Proof: For the first part of the proof we assume the serial argument position of N' is one where N' is upward monotonic and hence closed under intersections. W.l.o.g. let this be the last argument position.
We split N's arguments into $N'(w, \vec{x}, \vec{y}', y)$ where $\vec{x}$ denotes the downward monotonic argument positions and all others are upward monotonic. All but the last argument position may be empty. $\vec{y}$ abbreviates '$\vec{y}', y$'. For a world w and sets $\vec{X}, \vec{Y}'$ we define the intersection of all neighbourhoods:

$$\sqcap(w, \vec{X}, \vec{Y}') \stackrel{\text{def}}{=} \textstyle\bigcap_Y N'(w, \vec{X}, \vec{Y}', Y). \tag{55}$$

Since N' is serial, the range of the intersection is not empty. Closedness under intersections for the downward monotonic argument positions means for all $w, \vec{X}, \vec{Y}'$

$$N'(w, \vec{X}, \vec{Y}', \sqcap(w, \vec{X}, \vec{Y}')). \tag{56}$$

Notice that, (56) is false if N' is not serial. In this case $\sqcap(w, \vec{X}, \vec{Y}')$ is the set of all worlds, but this is not neccessarily a neighbourhood.
The accessibility relation $\mathcal{R}$ is now defined as

$$\mathcal{R}(w, \vec{x}, \vec{y}) \Leftrightarrow y \in \sqcap(w, \{\vec{x}\}, \{\vec{y}'\}^c). \tag{57}$$

Recall $\{y\}^c$ denotes the complement of the set $\{y\}$.
$\Rightarrow$ part of (54).
Suppose $N'(w, \vec{X}, \vec{Y}', Y)$ is true and $\forall \vec{x}, \vec{y}\ \mathcal{R}(w, \vec{x}, \vec{y}', y) \Rightarrow (\bigwedge_i x_i \in X_i) \Rightarrow (\bigvee_j y_j \in Y'_j \vee y \in Y)$ is false. That means the following facts are assumed for some $w, \vec{X}, \vec{Y}', Y, \vec{a}, \vec{b}', b$

	(a)	$N'(w, \vec{X}, \vec{Y}', Y)$	
	(b)	$\mathcal{R}(w, \vec{a}, \vec{b}', b)$	
	(c)	$a_i \in X_i$	for the downward monotonic arguments
	(d)	$b_i \notin Y'_i$	for the first upward monotonic arguments
	(e)	$b \notin Y$	for the last upward monotonic argument
$\succ$	(f)	$b \in \sqcap(w, \{\vec{a}\}, \{\vec{b}'\}^c)$	(b) and (57)
$\succ$	(g)	$N'(\{\vec{a}\}, \{\vec{b}'\}^c, \sqcap(w, \{\vec{a}\}, \{\vec{b}'\}^c)$	(f) and (56)
	(h)	$N'(\{\vec{a}\}, \vec{Y}', Y)$	(a), (c) and downward monotonicity
	(i)	$\vec{Y}' \subseteq \{\vec{b}'\}^c$	from (d)
$\succ$	(j)	$N'(\{\vec{a}\}, \{\vec{b}'\}^c, Y)$	(h), (i) and upward monotonicity
	(k)	$\sqcap(w, \{\vec{a}\}, \{\vec{b}'\}^c) \subseteq Y$	(g), (j) and (55)
$\succ$		contradiction with (e) and (f).	
$\succ$		The $\Rightarrow$ part of (54) holds.	'$\succ$ ' denotes 'this implies'.

$\Leftarrow$ part of (54).
Suppose $\forall \vec{x}, \vec{y}', y\ \mathcal{R}(w, \vec{x}, \vec{y}', y) \Rightarrow (\bigwedge_i x_i \in X_i) \Rightarrow (\bigvee_j y'_j \in Y'_j \vee y \in Y)$ is true.
$\succ$ $\forall \vec{x}, \vec{y}', y\ y \in \sqcap(w, \{\vec{x}\}, \{\vec{y}'\}^c) \Rightarrow (y \in Y \vee \neg(\bigwedge_i x_i \in X_i) \vee \bigvee_j y'_j \in Y'_j))$ (57)
$\succ$ $\forall \vec{x}, \vec{y}'\ ((\bigwedge_i x_i \in X_i) \wedge (\bigwedge_j y'_j \notin Y'_j)) \Rightarrow \sqcap(w, \{\vec{x}\}, \{\vec{y}\}'^c) \subseteq Y$ (logic and set theory)
$\succ$ $\forall \vec{x}, \vec{y}'\ ((\bigwedge_i x_i \in X_i) \wedge (\bigwedge_j y'_j \notin Y'_j)) \Rightarrow N'(w, \{\vec{x}\}, \{\vec{y}'\}^c, Y)$ (56) and (upward monotonicity)
$\succ$ $\forall \vec{y}'\ (\bigwedge_j y'_j \notin Y'_j) \Rightarrow N'(w, \vec{X}, \{\vec{y}'\}^c, Y)$ (closedness under unions)
$\succ$ $N'(w, \vec{X}, \vec{Y}', Y)$ (closedness under intersections)
$\succ$ The $\Leftarrow$ part of (54) holds.

If the argument position where N' is serial is not upward monotonic, but downward monotonic, there is a proof for the lemma which is just dual to the proof given above. Instead of intersections, use the union of all neighbourhoods. Therefore we do not give the proof here. $\lhd$

Seriality of the neighbourhood relation is one of the conditions in the previous lemma. The next lemma shows how this can be proved by alternatively proving an appropriate lemma from $\mathcal{A}_1$.

Lemma 4.17 (Seriality of the Neighbourhood Relation)
If $\mathcal{A}_1$ satisfies the conditions of Theorem 4.11 and the formula $\forall w\ \forall \vec{p}\ \exists q\ w \Vdash^{2} f(\vec{p}, q)$ is provable from $\mathcal{A}_1$ then the neighbourhood relation is serial in the last argument.
Proof: Since p and q are variables, this formula can be translated directly into $\forall w\ \forall \vec{p}\ \exists q\ N'_f(w, \vec{p}, q)$. $\lhd$

Seriality for other argument positions are proved analogously. For our test connectives we prove $\forall w, p\ \exists q\ w \Vdash^{2} p \rightsquigarrow q$ in Lemma A.13 and $\forall w \exists q\ w \Vdash^{2} \Box q$ in Lemma A.14.

Now we are ready to define a transformer that eliminates the closure properties.

Lemma 4.18 (Transformer for Closure Properties)
If $\mathcal{A}_1$ satisfies the conditions of Theorem 4.11 and Lemmas 4.13 and 4.17 for some connective f then the following transformer

$$\mathcal{T}_6\colon \quad w \models f(\vec{p}, \vec{q}) \Leftrightarrow \forall \vec{x}, \vec{y}\ \mathcal{R}_f(w, \vec{x}, \vec{y}) \Rightarrow (\bigwedge_i x_i \models p_i) \Rightarrow (\bigvee_j y_j \models q_j) \tag{58}$$

is sound and complete for $\mathcal{A}_2$.

Proof: Extension Lemma: Soundness and completeness of the transformer $\mathcal{T}_3$ implies that there is an extension of a model for $\mathcal{A}_2$ which satisfies (28): $w \models^{\mathcal{L}} f(\vec{p}) \Leftrightarrow \exists \vec{X}\ N'_f(w, \vec{X}) \wedge \bigwedge_i X_i = |p_i|$. In this model $w \models^{\mathcal{L}} f(\vec{p}) \Leftrightarrow N'_f(w, |\vec{p}|)$ holds. But according to Lemma 4.16, $N'_f(w, |\vec{p}|)$ is equivalent to the right hand side of (58).
The other lemmas are proved as in Lemma (4.2). $\lhd$

The instances of $\mathcal{T}_6$ for our test connectives are:

$$\begin{aligned} w \models^{\mathcal{L}} (p \rightsquigarrow q) \quad &\Leftrightarrow \quad \forall x, y\ \mathcal{R}_{\rightsquigarrow}(w, x, y) \Rightarrow (x \models^{\mathcal{L}} p \Rightarrow y \models^{\mathcal{L}} q) && (59)\\ w \models^{\mathcal{L}} \Box q \quad &\Leftrightarrow \quad \forall y\ \mathcal{R}_{\Box}(w, y) \Rightarrow y \models^{\mathcal{L}} q && (60) \end{aligned}$$

and we recongize the standard relational semantics for the $\Box$–operator. If we had axiomatized the $\Box$–operator with the classical implication connective $\rightarrow$ instead of $\rightsquigarrow$, then the transformer (60) together with $\mathcal{T}_1$ would turn the two axioms (14) and (15) for the $\Box$–operator into tautologies and we had found an optimal semantics for $\Box$. In order to obtain this result also for our axiomatization, we need to show that $\rightsquigarrow$ is equivalent to $\rightarrow$. Therefore we continue the investigation of the $\rightsquigarrow$–connective in In Section 4.7.

For $\wedge$–like connectives, the seriality condition cannot be proved, because $\forall w, p\ \exists q\ w \models^{2} (p \wedge q)$ is certainly not a theorem. In this case there is no semantics in terms of the accessibility relation as above. Usually, however, $\wedge$–like connectives are defined connectives. If it is for example defined as $(p \& q) \Leftrightarrow \neg(p \rightsquigarrow \neg q)$ then the semantics for $\rightsquigarrow$ can be used to define the semantics for &:

$$w \models^{\mathcal{L}} (p \& q) \Leftrightarrow \exists x, y\ \mathcal{R}_{\rightsquigarrow}(w, x, y) \wedge x \models^{\mathcal{L}} p \wedge y \models^{\mathcal{L}} q$$

Theorem 4.19 (Soundness and Completeness of the Transformation)
If $\mathcal{A}_1$ satisfies all the conditions of 4.18 and φ is a positive formula (see 4.11) then $\mathcal{A}_1$ entails φ iff $\mathcal{T}(\mathcal{T}_1(\mathcal{A}_1))$ entails $\mathcal{T}(\mathcal{T}_1(\varphi))$ where $\mathcal{T}$ is the strongest admissible transformer. $\lhd$

4.6 Binary Accessibility Relations

As we have seen in Lemma 4.18, n–place connectives correspond to $n+1$–ary accessibility relations. It is well known that m–place relations can be decomposed into $m+1$ binary relations. This trick can be used to find a translation into normal multi–modal systems. To this end we define a further K-transformation. For all $\vec{p}, \vec{q}, w$:

$$\begin{aligned} & \forall \vec{x}, \vec{y}{:}W\ \mathcal{R}(w, \vec{x}, \vec{y}) \Rightarrow ((\textstyle\bigwedge_i \varphi_i(x_i, p_i)) \Rightarrow (\bigvee_j \varphi_j(y_j, q_j)) \\ \Leftrightarrow\quad & \forall z{:}S'\ K(w, z) \Rightarrow ((\textstyle\bigwedge_i \exists x_i{:}W\ R_i(z, x_i) \wedge \varphi_i(x_i, p_i)) \Rightarrow \\ & \qquad\qquad (\textstyle\bigvee_j \forall y_j{:}W\ R_j(z, y_j) \Rightarrow \varphi_j(y_j, q_j)). \end{aligned} \tag{61}$$

Lemma 4.20 (Soundness and Completeness of (61))
The transformation lemmas hold for the K-transformation (61), if we use as transformation strategy inside–out rewriting, applying (61) from left to right on formulae which are the result of a transformation using $\mathcal{T}_6$.

Proof: Extension Lemma: We have to show that every interpretation $\Im$ for the accessibility relation $\mathcal{R}$ can be extended to an interpretation $\Im'$ satisfying (61). We interpret the sort S' as the set of n–tuples of worlds and the predicates R_k as the k^{th} projection function for these n–tuples. This interpretation satisfies (61).
Transformation Lemma: The results of a transformation with $\mathcal{T}_6$ are just formulae with the same structure as (61)'s left hand side. Therefore the transformation is an equivalence transformation.
Elimination Lemma: We have to show that every interpretation $\Im$ for the transformed system can be extended to an interpretation $\Im'$ satisfying (61). We define $\mathcal{R}$ as follows

$$\mathcal{R}(w, \vec{x}, \vec{y}) \Leftrightarrow \exists z\ K(w, z) \wedge \textstyle\bigwedge_i R_i(z, x_i) \wedge \bigwedge_j R_j(z, x_j).$$

This implies
$\forall\vec{x},\vec{y}\ \mathcal{R}(w,\vec{x},\vec{y}) \Rightarrow ((\bigwedge_i \varphi_i(x_i,p_i)) \Rightarrow (\bigvee_j \varphi_j(y_j,q_j))$
$\Leftrightarrow \forall\vec{x},\vec{y}\ (\exists z\ K(w,z) \wedge \bigwedge_i R_i(z,x_i) \wedge \bigwedge_j R_j(z,x_j)) \Rightarrow ((\bigwedge_i \varphi_i(x_i,p_i)) \Rightarrow (\bigvee_j \varphi_j(y_j,q_j))$
$\Leftrightarrow \forall z\ K(w,z) \Rightarrow \vec{x},\vec{y}\ \bigwedge_i R_i(z,x_i) \wedge \bigwedge_j R_j(z,x_j)) \Rightarrow ((\bigwedge_i \varphi_i(x_i,p_i)) \Rightarrow (\bigvee_j \varphi_j(y_j,q_j))$
$\Leftrightarrow \forall z\ K(w,z) \Rightarrow (\bigvee_i \forall x_i\ \neg R_i(z,x_i) \vee \varphi_i(x_i,p_i)) \vee (\bigvee_j \neg R_j(z,x_j) \vee \varphi_j(y_j,q_j))$
$\Leftrightarrow \forall z\ K(w,z) \Rightarrow ((\bigwedge_i \exists x_i\ R_i(z,x_i) \wedge \varphi_i(x_i,p_i)) \Rightarrow (\bigvee_j \forall y_j\ R_j(z,y_j) \Rightarrow \varphi_j(y_j,q_j)).$ $\triangleleft$

The composition of $\mathcal{T}_6$ with the transformer (61) is

$$\mathcal{B}_3 \qquad w \models^{\mathcal{L}} f(\vec{p},\vec{q}) \Leftrightarrow \forall z{:}S'\ K(w,z) \Rightarrow ((\bigwedge_i \exists x_i{:}W\ R_i(z,x_i) \wedge \varphi_i(x_i,p_i)) \Rightarrow (\bigvee_j \forall y_j{:}W\ R_j(z,y_j) \Rightarrow \varphi_j(y_j,q_j)). \tag{62}$$

This in turn corresponds to a translation function π_3 which maps n-place connectives to modal operators:

$$\pi_3(f(\vec{p},\vec{q})) = [K]((\bigwedge_i \langle R_i\rangle \pi_3(p_i)) \Rightarrow \bigvee_j [R_j]\pi_3(q_j)). \tag{63}$$

At different places we have defined translations π_i into the language of multi–modal logic. But there is a very subtle point to be discussed here. The transformers $\mathcal{B}_1$, $\mathcal{B}_2^+$, $\mathcal{B}_2^-$ and $\mathcal{B}_3$ introduce a new sort S'. That means the worlds are split into two different kinds, worlds of sort W and worlds of sort S'. The outermost quantification of $\mathcal{T}_1(\varphi)$, however, quantifies over the sort W. Therefore the frame conditions specified by φ don't hold for all worlds, but only for the worlds of sort W.

To see what this means, suppose as an example, $\varphi = \Box p \Vdash^{2} p$. Let us assume strong neighbourhood semantics. Quantifier elimination of $\mathcal{S}(\mathcal{B}_2^+(\mathcal{T}_1(\varphi)))$ yields $\forall w{:}W\ \forall v{:}S'\ K(w,v) \Rightarrow R(v,w)$. This is the frame property corresponding to φ. Now consider $\varphi' \stackrel{\text{def}}{=} \pi_2^+(\varphi) = \langle K\rangle[R]p \Vdash^{2} p$. Quantifier elimination of $\mathcal{S}(\mathcal{T}_6(\mathcal{T}_1(\varphi')))$ yields $\forall w,v{:}W\ K(w,v) \Rightarrow R(v,w)$. That means, if we interpret the formula φ' as a normal axiom schema in normal multi modal logic, we get frame conditions for all worlds whereas the original schema φ specified a frame condition only for certain worlds. Thus, the target system for the π_i is *not* standard multi modal logic, but a refined version with some more structure.

The additional structure in the semantics is the separation into two different sets of worlds. The accessibility relations K lead from the W–type worlds to the S'–type worlds and vice versa for the R–relations. The additional structure on the syntactic side is an invariant on the nesting sequence of the modal operators. All translations π_i yield K–type operators followed by R–type operators which are again followed by K–type operators etc. Formulae like $[K][K]\ldots$ or $[R][R]\ldots$ do not occur. Therefore we can classify the formulae as K–type formula if the top–level operators are K–type connectives operating on R–type formulae. Analogously, R–type formulae have top–level operators of R–type operating on K–type formulae. Pure classical propositional formulae fall into both sets. K–type formulae are to be interpreted only at W–type worlds and R–type formulae are to be interpreted at S'–type worlds.

I omit giving a formal definition because I think the structure is simple. What is important, however, is that the axiomatization of this logic with K-axiom and Necessitation rules for each of these operators needs to be modified also. For the K–type operators, K-axiom and Necessitation rule are defined only for R–type formulae, and vice versa for the R–type operators. For example, the p's and q's in the K-axiom $[K](p \Rightarrow q) \vdash ([K]p \Rightarrow [K]q)$ for the K–type operator must be R–type formulae, otherwise the syntactic structure does not match the semantics. Let us call this logic a *separated* multi modal logic.

Exploiting soundness and completeness of $\mathcal{T}_6$ we can now prove soundness and completeness of these translations into separated multi–modal logic.

Theorem 4.21 (Translation into Multi–Modal Logic: Soundness and Completeness) The translations π_1, π_2^+, π_2^- and π_3 (defined in (33), (43), (44) and (63)) to the separated multi modal logic are sound and complete.

Proof: Let $\mathcal{T}_6'$ be the instance of $\mathcal{T}_6$ for the appropriate number of parameterized modal operators. For K–type operators, $\mathcal{T}_6'$ quantifies over worlds of sort W and for R–type operators, $\mathcal{T}_6'$ quantifies over worlds of sort S. By the definition of the π_i, $\pi_1 \circ \mathcal{T}_1 \circ \mathcal{T}_6' = \mathcal{T}_1 \circ \mathcal{B}_1$, holds $\pi_2^+ \circ \mathcal{T}_1 \circ \mathcal{T}_6' = \mathcal{T}_1 \circ \mathcal{B}_2^+$,holds $\pi_2^- \circ \mathcal{T}_1 \circ \mathcal{T}_6' = \mathcal{T}_1 \circ \mathcal{B}_2^-$ holds and also $\pi_3 \circ \mathcal{T}_1 \circ \mathcal{T}_6' = \mathcal{T}_1 \circ \mathcal{B}_3$ holds, where $\circ$ denotes composition. Let $(\pi, \mathcal{B})$ be one of these four pairs of transformers. If K is the set of K-axioms and N is the set of Necessitation rules for the different modal operators then soundness and completeness (s&c) of the transformers $\mathcal{B}$ and $\mathcal{T}_6'$ allow us to prove

	$\mathcal{A}_1 \Rightarrow \varphi$	
iff	$\mathcal{B}(\mathcal{T}_1(\mathcal{A}_1)) \Rightarrow \mathcal{B}(\mathcal{T}_1(\varphi))$	(s&c) of $\mathcal{T}_1$ and $\mathcal{B}$
iff	$\mathcal{T}_6'(\mathcal{T}_1(\pi(\mathcal{A}_1))) \Rightarrow \mathcal{T}_6'(\mathcal{T}_1(\pi(\varphi)))$	above equations
iff	$\mathcal{T}_6'(\mathcal{T}_1(K) \wedge \mathcal{T}_1(N) \wedge \mathcal{T}_1(\pi(\mathcal{A}_1))) \Rightarrow \mathcal{T}_6'(\mathcal{T}_1(\pi(\varphi)))$	$\mathcal{T}_6'(\mathcal{T}_1(K) \wedge \mathcal{T}_1(N)) = \mathit{true}$
iff	$\mathcal{T}_1(K) \wedge \mathcal{T}_1(N) \wedge \mathcal{T}_1(\pi(\mathcal{A}_1)) \Rightarrow \mathcal{T}_1(\pi(\varphi))$	(s&c) of $\mathcal{T}_6'$
iff	$K \wedge N \wedge \pi(\mathcal{A}_1) \Rightarrow \pi(\varphi)$	(s&c) of $\mathcal{T}_1$.

$\triangleleft$

Usually the direct $\mathcal{B}$–transformations using binary accessibility relations is sufficient, and we need not take a detour via multi modal logic. Therefore I have not investigated the difference between this kind of separated multi modal logic, introduced above and the standard multi modal logic. I conjecture that there is no significant difference as long as the syntactic invariant of the nested K–type and R–type formulae is guaranteed. Direct translations for modal logic with neighbourhood semantics into normal bi- or tri-modal logic respectively have been investigated by Olivier Gasquet and Andreas Herzig [GH93a, GH93b]. It turnes out that it is extremely complicated to extend the frame properties from the worlds of sort W to all worlds. There is no general scheme and each frame property has to be treated individually.

4.7 Truth Value Semantics

For a connective f with a semantics defined in terms of the accessibility relation we can prove that f is actually a classical connective by proving that the accessibility relation collapses to a point relation.

A point relation is one where general reflexivity $\forall x\ \mathcal{R}(x, \ldots, x)$ holds and all arguments of $\mathcal{R}$ 'collapse' in the following sense: $\forall \vec{x}\ \mathcal{R}(\vec{x}) \Rightarrow \bigwedge_{i,j} x_i = x_j$ holds.

For our test connective $\rightsquigarrow$ we can actually prove these properties. In order to illustrate how the translation, quantifier elimination and theorem proving techniques work together, we list all the details. First, the two axioms (12) and (13), namely

$$\begin{array}{ll} \forall p, q, r, s & ((p \rightsquigarrow q) \rightsquigarrow r) \models^2 ((r \rightsquigarrow p) \rightsquigarrow (s \rightsquigarrow p)) \\ \forall p, q, r, s & p \models^2 q \wedge (p \rightsquigarrow q) \models^2 (r \rightsquigarrow s) \Rightarrow r \models^2 s, \end{array}$$

can be translated with a composition of the transformers $\mathcal{T}_1$,$\mathcal{T}_6$ and $\mathcal{S}$. This is simple rewriting. The results are:
$\forall w\ (\forall u, v\ \mathcal{R}(w, u, v) \Rightarrow ((\forall x, y\ \mathcal{R}(u, x, y) \Rightarrow (p(x) \Rightarrow q(y))) \Rightarrow r(v)))$
$\quad \Rightarrow (\forall a, b\ \mathcal{R}(w, a, b) \Rightarrow ((\forall x, y\ \mathcal{R}(a, x, y) \Rightarrow (r(x) \Rightarrow p(y))) \Rightarrow (\forall c, d\ \mathcal{R}(b, c, d) \Rightarrow (s(c) \Rightarrow p(d)))))$
and
$((\forall w\, p(w) \Rightarrow q(w)) \wedge (\forall w\ (\forall u, v\ (\mathcal{R}(w, u, v) \Rightarrow p(u) \Rightarrow q(v))) \Rightarrow (\forall u, v\ \mathcal{R}(w, u, v) \Rightarrow (r(u) \Rightarrow s(v)))))$
$\quad \Rightarrow (\forall w\ r(w) \Rightarrow s(w))$.

All predicates except $\mathcal{R}$ are universally quantified. To simplify the formulae we apply the quantifier elimination algorithm SCAN. First we negate the formulae in order to get existentential quantifications. The negation is clausified, all existentially quantified predicates are resolved away, the quantifiers are reconstructed and the formula is negated again. Lets start with the first formula. Its negation is
$\exists w\ (\forall u, v\ \mathcal{R}(w, u, v) \Rightarrow ((\forall x, y\ \mathcal{R}(u, x, y) \Rightarrow (p(x) \Rightarrow q(y))) \Rightarrow r(v)))$
$\quad \wedge (\exists a, b\ \mathcal{R}(w, a, b) \wedge (\forall x, y\ \mathcal{R}(a, x, y) \Rightarrow (r(x) \Rightarrow p(y))) \wedge (\exists c, d\ \mathcal{R}(b, c, d) \wedge s(c) \wedge \neg p(d)))$.

The clause form is

C_1	$\neg\mathcal{R}(w, u, v), \mathcal{R}(u, f(u, v), g(u, v)), r(v)$
C_2	$\neg\mathcal{R}(w, u, v), p(f(u, v)), r(v)$
C_3	$\neg\mathcal{R}(w, u, v), \neg q(g(u, v)), r(v)$
C_4	$\mathcal{R}(w, a, b)$
C_5	$\neg\mathcal{R}(a, x, y), \neg r(x), p(y)$

C_6	$\mathcal{R}(b, c, d)$	
C_7	$s(c)$	u, v, x, y are variables.
C_8	$\neg p(d)$	all other symbols are Skolem constants or functions.

The result of resolving p, q, r, s away is

C'_1 $\neg\mathcal{R}(w, u, v), \mathcal{R}(u, f(u, v), g(u, v)), \neg\mathcal{R}(a, v, d)$
C'_2 $\neg\mathcal{R}(w, u, v), f(u, v) \neq d, \neg\mathcal{R}(a, v, d)$
C_4 $\mathcal{R}(w, a, b)$
C_6 $\mathcal{R}(b, c, d)$.

The quantifier prefix is reconstructed. f and g become existentially quantified variables again. The resulting formula is
$\exists w, a, b, c, d\, \forall u, v\, \exists x, y\; \mathcal{R}(w, a, b) \wedge \mathcal{R}(b, c, d) \wedge (\neg\mathcal{R}(w, u, v) \vee \neg\mathcal{R}(a, v, d) \vee (\mathcal{R}(u, x, y) \wedge x \neq d)$
To obtain the final result, this formula is negated.

$$\forall w, a, b, c, d\; (\mathcal{R}(w, a, b) \wedge \mathcal{R}(b, c, d)) \Rightarrow \exists u, v\; \mathcal{R}(w, u, v) \wedge \mathcal{R}(a, v, d) \wedge \forall x, y\; \mathcal{R}(u, x, y) \Rightarrow x = d. \tag{64}$$

That means in relational possible worlds semantics the Łukasiewics axiom (12) corresponds to the property (64) of the accessibility relation.

Now lets turn to the second formula. Its negation is
$((\forall w\; p(w) \Rightarrow q(w)) \wedge \forall w\; (\forall u, v\; \mathcal{R}(w, u, v) \Rightarrow (p(u) \Rightarrow q(v))) \Rightarrow (\forall u, v\; \mathcal{R}(w, u, v) \Rightarrow (r(u) \Rightarrow s(v))))$
$\wedge\; (\exists w\; r(w) \wedge \neg s(w))$.

The clause form is

C_1	$\neg p(w), q(w)$	
C_2	$\mathcal{R}(w, f(w), g(w)), \neg\mathcal{R}(w, x, y), \neg r(x), s(y)$	
C_3	$p(f(w)), \neg\mathcal{R}(w, x, y), \neg r(x), s(y)$	
C_4	$\neg q(g(w)), \neg\mathcal{R}(w, x, y), \neg r(x), s(y)$	
C_5	$r(a)$	w, x, y are variables.
C_6	$\neg s(a)$	f, g, a are Skolem symbols.

The result of resolving p, q, r, s away is

C'_2 $\mathcal{R}(w, f(w), g(w)), \neg\mathcal{R}(w, a, a)$
C'_3 $f(w) \neq g(w), \neg\mathcal{R}(w, a, a)$.

The quantifier prefix is reconstructed. f and g become existentially quantified variables. The resulting formula is $\exists a\; \forall w\; \exists u, v\; \neg\mathcal{R}(w, a, a) \vee (\mathcal{R}(w, u, v) \wedge u = v)$.
Negating this formula, we obtain the final result

$$\forall a\; \exists w\; \mathcal{R}(w, a, a) \wedge \forall u, v\; \mathcal{R}(w, u, v) \Rightarrow u = v. \tag{65}$$

That means in relational possible worlds semantics our version of Modus Ponens (13) corresponds to the property (65) of the accessibility relation.

From the two translated axioms (64) and (65), the three lemmas
$\forall x\; \mathcal{R}(x, x, x)$ and $\forall w, u, v\; R(w, u, v) \Rightarrow w = u$ and $\forall w, u, v\; R(w, u, v) \Rightarrow u = v$
follow, which shows that $\mathcal{R}$ collapses to a point relation. The proofs were found with the Otter theorem prover. They are listed in A.15 in the Appendix. The transformer
$w \models^{\mathcal{L}} (p \rightsquigarrow q) \quad\Leftrightarrow\quad \forall x, y\; \mathcal{R}_{\rightsquigarrow}(w, x, y) \Rightarrow (x \models^{\mathcal{L}} p \Rightarrow y \models^{\mathcal{L}} q)$
now simplifies to
$w \models^{\mathcal{L}} (p \rightsquigarrow q) \quad\Leftrightarrow\quad (w \models^{\mathcal{L}} p \Rightarrow w \models^{\mathcal{L}} q)$
which proves the equivalence between $\rightsquigarrow$ and $\Rightarrow$.

5 The Semantics Generation Procedure

We collect the results of the previous section and define a procedure for developing the semantics for the given non–classical operators.

We start with an arbitrary formula set $\mathcal{A}_1$ containing the $\vdash^{2}$–predicate. Naturally $\mathcal{A}_1$ should be consistent, otherwise the following steps have no meaning.

Step 1 Prove that $\vdash^{2}$ is reflexive and transitive.

Step 2 Prove the congruence properties, (17) for all connectives. If there are connectives where this is not possible, the logic is outside the scope of the current theory. If the congruence properties hold for all connectives and $\vdash^{2}$ is the only predicate in $\mathcal{A}_1$, and all formula variables are universally quantified then any of the transformers $\mathcal{T}_2$ (24), $\mathcal{T}_3$ (28) or $\mathcal{B}_1$ (32) defines a sound and complete semantics.

Step 3 Try proving for each connective f the monotonicity properties (34) and (35) from $\mathcal{A}_1$. In this step we determine for each argument position of f, whether it is upward or downward monotonic. We assume that each argument position is monotonic. If some but not all argument positions are monotonic, or one argument position is both upward and downward monotonic, there is a straightforward extension of our theory which can deal with this (we leave it to the reader to show this).

The actual semantics is now determined by one of the transformers $\mathcal{T}_4$ (41), $\mathcal{T}_5$(42), $\mathcal{B}_2^+$ (43) or $\mathcal{B}_2^-$ (44) respectively, for each monotonic connective, and one of the previous transformers for the non–monotonic connectives. All can be mixed freely.

Step 4 Check whether $\mathcal{A}_1$ contains an axiomatization of standard propositional logic. Find a model for $\mathcal{A}_1 \wedge (\exists p, q\ \neg(p \vdash^{2} q))$ to ensure that the next transformations do not produce inconsistencies. If both conditions are fulfilled, one can simplify the previously transformed axioms $\mathcal{A}$ by applying the transformer $\mathcal{S}$ which replaces literals $v \models^{\mathcal{L}} p$ in $\mathcal{A}$ with $p(v)$ and trying the quantifier elimination algorithm on these PL2 formulae. Notice that this is an optional simplification step. If the quantifier elimination algorithm does not compute an equivalent first–order formula, then the original translated first–order formula $\mathcal{A}$ is the final result of the transformation. In this way we can also deal with so–called incomplete systems.

Step 5 If the conditions of the previous step are fulfilled and $\mathcal{A}_1$ consists of definite Horn clauses only, then check for the closure properties. For the connectives which are either upward or downward monotonic for each argument position, try proving (48) for each upward monotonic argument position and (49) for each downward monotonic argument position. If this is successful then the transformer $\mathcal{T}_6$ (58) yields the classical relational possible worlds semantics. Translate $\mathcal{A}_1$ again using the strongest possible transformer for each connective. As in the previous step, these translated axioms can be simplified using the transformer $\mathcal{S}$ and quantifier elimination.

Notice that this step and the corresponding completeness proof subsumes the Sahlquist technique frequently used in modal logic to get frame properties from Hilbert axioms [Sah75, vB83].

Step 6 For connectives with relational semantics (the previous step was successful) try proving that the accessibility relation collapses to a point relation (see Section 4.7). If this is provable then the connective has a standard truth value semantics.

The completeness lemmas guarantee that a (positive) theorem can be proved from the original Hilbert axioms $\mathcal{A}_1$ if and only if the translated theorem can be proved from the resulting translated axiom system.

6 Summary

I have presented an alternative way for developing model theoretic semantics from an axiomatic specification of non–classical propositional logics. The semantics is represented as syntactic transformation rules. These transformation rules are derived from particular theorems of the original system and they translate these theorems into tautologies. Reflexivity and transitivity of a binary consequence relation yields the basic possible worlds framework. The congruence properties of the connectives yield weak neighbourhood semantics. The monotonicity properties yield a stronger neighbourhood semantics. The closure properties yield relational possible worlds semantics for the connectives.

Propositional logic as basis of the specification allows us to turn those parts which are not eliminated into second–order predicate logic formulae. In many cases these formulae can be simplified to a first–order predicate logic formula which describes the corresponding frame property. As a consequence, the correspondence problem reduces to a quantifier elimination problem which can be solved with a quantifier elimination algorithm like SCAN. All transformations preserve satisfiability and unsatisfiability. This is sufficient for the soundness and completeness of the generated semantics.

Sometimes there are alternative transformations for the same property. For example the transformations which produce neighbourhood semantics may define a neighbourhood relation between worlds or a neighbourhood relation between a world and sets of worlds. Moreover, one can define transformations which produce for n–place connectives only binary accessibility relations. We have shown how these transformation can be turned into a translation from the original logic into multi–modal logic.

With PL1 theorem provers, quantifier elimination algorithms and model finders, all steps of the transformation process can be automated.

This work can be extended in various directions. For example, one can try to find transformations for other properties than the ones considered in this paper. The starting point for our method were the congruence properties of the connectives. There are, however, quite interesting and useful connectives which do not have the congruence properties and for which our method does not work. An example is Fagin and Halpern's awareness operator [FH88].

In order to apply Stone's representation theorem we need to have a propositional logic as basis. It should be investigated whether some of the representation theorems for weaker structures than Boolean algebras can be exploited to investigate extensions of intuitionistic or relevance logics. Of particular interest are of course logics with quantifiers. It is not clear how quantifiers fit into our framework.

Although in this paper we investigated only specifications of logics, the ideas and techniques used are independent of this particular application. There might be other areas where similar manipulations of logical specifications also yield interesting results. In particular, the area of representation theorems for algebraic systems is related to this work. For example Jónsson and Tarski's representation theorem for Boolean algebras with operators [JT51, JT52] gives an alternative completeness theorem for relational possible worlds semantics. Transferring the ideas and methods developed in this paper to algebraic logic and to general algebra seems promising.

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.

[BG90] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In *CADE-10: 10th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, pages 427–441, Kaiserslautern, FRG, 1990. Springer-Verlag. Copy filed.

[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.

[Che80] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, 1980.

[FH88] Ronald Fagin and Joseph Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[GH93a] Olivier Gasquet and Andreas Herzig. Translating inaccessible worlds logic into bimodal logic. In Michael Clarke, Rudolf Kruse, and Serafín Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Proceedings of ECSQARU '93, Granada, Spain, Nov. 1993*, volume 747 of *LNCS*, pages 145–150, Granada, Spain, 8–10 November 1993.

[GH93b] Olivier Gasquet and Andreas Herzig. Translating non-normal modal logics into normal modal logics. In I. J. Jones and M. Sergot, editors, *Proc. of International Workshop on Deontic Logic in Computer Science (DEON–94)*, TANO, Oslo, December 1993.

[GO92a] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second–order predicate logic. *South African Computer Journal*, 7:35–43, July 1992. also published in [GO92b].

[GO92b] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second–order predicate logic. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Principles of Knowledge Representation and Reasoning (KR92)*, pages 425–435. Morgan Kaufmann, 1992. also published as a technical report MPI-I-92-231 of the Max-Planck-Institut für Informatik, Saarbrücken and in the South African Computer Journal, 1992.

[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.

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

[Kri63] S. A. Kripke. Semantical analysis of modal logic i, normal propositional calculi. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[Łuk70] J. Łukasiewicz. *Selected Works.* North Holland, 1970. Edited by L. Borkowski.

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

[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.

[OGP94] Hans Jürgen Ohlbach, Dov Gabbay, and David Plaisted. Killer transformations. Technical Report MPI-I-94-226, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994. To be published in Proc. of the 1993 Workshop on Proof Theory in Modal Logic, Hamburg.

[PS90] P. Pritchard and J. Slaney. Computing models of propositional logics. In *10th International Conference on Automated Deduction, CADE-10, LNCS 449*, page 685. Springer Verlag, 1990.

[Sah75] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In S. Kanger, editor, *Proceedings of the 3rd Scandinavian Logic Symposium, 1973*, pages 110–143, Amsterdam, 1975. North Holland.

[Sim94] Harold Simmons. The monotonous elimination of predicate variables. *Journal of Logic and Computation*, 4(1), 1994.

[Sto36] M. H. Stone. The theory of representations for boolean algebras. *Transactions of American Mathematical Society*, 40:37–111, 1936.

[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.

[vB83] Johan van Benthem. *The Logic of Time.* Reidel, Kluwer Academic Publishers, Dordrecht, 1983.

[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.

[Wal87] Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Pitman Ltd., London, 1987.

A Appendix: Proofs for the Test Example

We list the proofs for the key lemmas needed Section 4 for jusitfying the application of the transformations. All proofs were found by the theorem prover Otter [McC89, McC90]. The proofs of the lemmas below are almost the original output listings of Otter. Only the superfluous information has been deleted and the layout has been changed slightly. I used Otter Version 2.2xa on a Solburn machine with SuperSparc processors. Some of the proofs required several hours of CPU time, and the program generated up to 70 million clauses. Otter is a refutational theorem prover. Therefore the theorem to be proved is negated and the systems tries to find the empty clause F, the basic contradiction. '|' is Otter's symbol for disjunction. x != y means $x \neq y$. All other symbols are self explanatory.

In most cases I used hyperresolution as inference rule. Except for the weighting limit, which limits the size of the derived clauses, no other control parameter was changed from its default value. For some of the proofs the weighting limit turned out to be crucial. No proof could be found for other weighting limits. During the proof search, however, there was no interaction with the system. In most cases all formulae were put into the set of support (sos) list. This is necessary because hyperresolution is not complete together with the sos strategy.

In the protocols below, `d` is used for the binary consequence relation $\models^2$, `i` is the implication connective and `b` stands for the $\Box$–operator. `x,y,z,u,v,w` are variable symbols.

Lemma A.1 (Reflexivity of $\models^2$)
From the axioms (12) and (13) we prove that the consequence relation is reflexive.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
% negated theorem.
-(all p d(p,p)).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3  -d(c1,c1).
----> UNIT CONFLICT at   1.58 sec ----> 380 [379,3] F.
---------------- PROOF ----------------
  4 [2,1,1]     d(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u))).
  5 [4,2,4]     d(x,i(i(y,z),i(z,i(y,z)))).
  8 [5,2,1]     d(i(i(i(x,y),i(y,i(x,y))),z),i(u,z)).
 16 [8,2,1]     d(x,i(i(i(y,i(z,y)),z),i(u,z))).
 21 [16,2,1]    d(i(i(i(i(x,i(y,x)),y),i(z,y)),u),i(v,u)).
253 [21,2,4]    d(x,i(i(y,z),i(u,i(y,z)))).
259 [253,2,253] d(i(x,y),i(z,i(x,y))).
288 [259,2,1]   d(i(i(x,i(y,z)),y),i(u,y)).
310 [288,2,1]   d(i(i(x,y),i(z,i(y,u))),i(v,i(z,i(y,u)))).
369 [310,2,259] d(x,i(y,i(z,z))).
376 [369,2,369] d(x,i(y,y)).
379 [376,2,376] d(x,x).
380 [379,3]     F.
```

◁

Lemma A.2 (Transitivity of $\models^2$)
From the axioms (12) and (13) we prove the transitivity of the consequence relation.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
% negated theorem.
-(all p q r (d(p,q)&d(q,r)->d(p,r))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4   d(c3,c2).
5   d(c2,c1).
6  -d(c3,c1).
----> UNIT CONFLICT at 3612.46 sec ----> 41345 [41344,6] F.
---------------- PROOF ----------------
    7 [2,3,1]         d(i(i(x,y),x),i(z,x)).
    9 [7,2,7]         d(x,i(y,y)).
   10 [7,2,1]         d(i(i(x,y),i(y,z)),i(u,i(y,z))).
   11 [9,2,1]         d(i(i(x,x),y),i(z,y)).
   20 [10,2,1]        d(i(i(x,i(y,z)),i(u,y)),i(v,i(u,y))).
   30 [20,2,1]        d(x,i(i(i(i(y,z),u),z),i(y,z))).
   31 [20,2,1]        d(i(i(x,i(y,z)),i(u,i(z,v))),i(w,i(u,i(z,v)))).
   32 [30,2,30]       d(i(i(i(x,y),z),y),i(x,y)).
   34 [32,2,20]       d(x,i(y,i(z,y))).
   35 [32,2,7]        d(x,i(y,x)).
   37 [32,2,1]        d(i(i(x,y),i(i(x,y),z)),i(u,i(i(x,y),z))).
   47 [35,2,5]        d(x,i(c2,c1)).
   48 [35,2,4]        d(x,i(c3,c2)).
   50 [35,2,1]        d(i(i(x,i(y,z)),y),i(u,y)).
   51 [47,2,35]       d(x,i(y,i(c2,c1))).
   52 [47,2,1]        d(i(i(c2,c1),x),i(y,x)).
   53 [48,2,35]       d(x,i(y,i(c3,c2))).
   54 [48,2,1]        d(i(i(c3,c2),x),i(y,x)).
   56 [34,2,1]        d(i(i(x,i(y,x)),z),i(u,z)).
   58 [51,2,1]        d(i(i(x,i(c2,c1)),y),i(z,y)).
   60 [53,2,1]        d(i(i(x,i(c3,c2)),y),i(z,y)).
  267 [37,2,60]       d(x,i(i(i(y,i(c3,c2)),z),z)).
  268 [37,2,58]       d(x,i(i(i(y,i(c2,c1)),z),z)).
  269 [37,2,56]       d(x,i(i(i(y,i(z,y)),u),u)).
  270 [37,2,54]       d(x,i(i(i(c3,c2),y),y)).
  271 [37,2,52]       d(x,i(i(i(c2,c1),y),y)).
  272 [37,2,50]       d(x,i(i(i(y,i(z,u)),z),z)).
  284 [37,2,11]       d(x,i(i(i(y,y),z),z)).
  287 [37,2,7]        d(x,i(i(i(y,z),y),y)).
  294 [270,2,270]     d(i(i(c3,c2),x),x).
  297 [271,2,294]     d(i(i(c2,c1),x),x).
  300 [284,2,297]     d(i(i(x,x),y),y).
  303 [287,2,300]     d(i(i(x,y),x),x).
  306 [303,2,1]       d(i(x,i(x,y)),i(z,i(x,y))).
  307 [267,2,303]     d(i(i(x,i(c3,c2)),y),y).
  311 [268,2,307]     d(i(i(x,i(c2,c1)),y),y).
  315 [269,2,311]     d(i(i(x,i(y,x)),z),z).
  319 [272,2,315]     d(i(i(x,i(y,z)),y),y).
  323 [319,2,1]       d(i(x,i(y,i(x,z))),i(u,i(y,i(x,z)))).
  339 [306,2,306]     d(x,i(i(y,i(y,z)),i(y,z))).
  342 [339,2,339]     d(i(x,i(x,y)),i(x,y)).
  364 [342,2,10]      d(i(i(x,y),i(y,z)),i(y,z)).
  371 [364,2,1]       d(i(i(x,y),i(z,x)),i(u,i(z,x))).
  458 [371,2,32]      d(x,i(y,i(z,x))).
  605 [323,2,1]       d(x,i(i(y,z),i(i(i(z,u),y),z))).
  609 [605,2,605]     d(i(x,y),i(i(i(y,z),x),y)).
  622 [609,2,458]     d(i(i(i(x,i(y,z)),u),z),i(x,i(y,z))).
  674 [609,2,54]      d(i(i(i(x,y),z),i(i(c3,c2),y)),i(x,y)).
  676 [609,2,50]      d(i(i(i(x,y),z),i(i(u,i(y,v)),y)),i(x,y)).
  690 [609,2,7]       d(i(i(i(x,y),z),i(i(y,u),y)),i(x,y)).
 3887 [31,2,1]        d(x,i(i(i(y,z),u),i(z,u))).
 3899 [3887,2,3887]   d(i(i(x,y),z),i(y,z)).
 3908 [3899,2,1]      d(x,i(i(x,y),i(z,y))).
 4217 [3908,2,3899]   d(x,i(i(i(y,x),z),i(u,z))).
 4225 [3908,2,609]    d(i(i(i(i(x,y),i(z,y)),u),x),i(i(x,y),i(z,y))).
 4229 [3908,2,1]      d(i(i(i(i(x,y),z),i(u,z)),x),i(v,x)).
 4604 [4217,2,1]      d(i(i(i(i(x,i(y,z)),u),i(v,u)),y),i(w,y)).
 5104 [4229,2,342]    d(i(i(i(i(x,y),z),i(u,z)),x),x).
 5118 [5104,2,676]    d(i(i(i(i(x,i(i(y,z),u)),i(y,z)),v),z),i(y,z)).
 6337 [4604,2,342]    d(i(i(i(i(x,i(y,z)),u),i(v,u)),y),y).
 6354 [6337,2,674]    d(i(i(x,i(i(i(c3,c2),i(y,z)),u)),z),i(y,z)).
16572 [4225,2,6354]   d(i(i(x,y),z),i(i(i(c3,c2),i(x,y)),z)).
18364 [16572,2,1]     d(i(i(c3,c2),i(i(x,y),z)),i(i(z,x),i(u,x))).
39219 [5118,2,690]    d(i(i(x,i(i(i(y,z),y),u)),i(i(y,z),y)),y).
39255 [39219,2,4225]  d(i(x,y),i(i(i(x,z),x),y)).
40733 [39255,2,5]     d(i(i(c2,x),c2),c1).
40751 [39255,2,18364] d(i(c2,i(c3,x)),i(y,i(c3,x))).
40917 [40733,2,622]   d(c2,i(x,c1)).
41296 [40751,2,40917] d(x,i(c3,c1)).
41344 [41296,2,41296] d(c3,c1).
41345 [41344,6]       F.
```

$\triangleleft$

Lemma A.3 (First Congruence Lemma for $\rightsquigarrow$)
From the axioms (12) and (13) we prove the first congruence lemma (19) for $\rightsquigarrow$:

$$\forall p, q\ (p \stackrel{2}{\vdash} q \wedge q \stackrel{2}{\vdash} p) \Rightarrow \forall s\ p \rightsquigarrow s \stackrel{2}{\vdash} q \rightsquigarrow s).$$

As lemmas we use reflexivity and transitivity of $\stackrel{2}{\vdash}$.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
% negated theorem.
-(all p q (d(p,q)&d(q,p)-> (all s d(i(p,s),i(q,s))))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5   d(c3,c2).
6   d(c2,c3).
7  -d(i(c3,c1),i(c2,c1)).
----> UNIT CONFLICT at 886.29 sec ----> 35208 [35207,7] F.
---------------- PROOF ----------------
    9 [2,3,1]       d(i(i(x,y),x),i(z,x)).
   11 [9,4,1]       d(i(i(x,y),i(z,x)),i(u,i(z,x))).
   13 [9,2,9]       d(x,i(y,y)).
   16 [13,2,1]      d(i(i(x,x),y),i(z,y)).
   18 [16,4,1]      d(i(i(x,y),x),i(z,i(u,x))).
   31 [18,2,9]      d(x,i(y,i(z,y))).
   34 [31,2,31]     d(x,i(y,x)).
   37 [34,4,6]      d(c2,i(x,c3)).
   40 [34,4,1]      d(x,i(i(x,y),i(z,y))).
   43 [34,2,5]      d(x,i(c3,c2)).
   47 [37,4,1]      d(c2,i(i(c3,x),i(y,x))).
   55 [43,2,1]      d(i(i(c3,c2),x),i(y,x)).
  770 [11,2,40]     d(x,i(i(i(i(y,z),u),z),i(y,z))).
 2355 [770,2,770]   d(i(i(i(x,y),z),y),i(x,y)).
 2421 [2355,4,55]   d(i(i(i(i(c3,c2),x),y),x),i(z,x)).
 2468 [2421,2,9]    d(x,i(i(i(c3,c2),y),y)).
 2479 [2468,2,2468] d(i(i(c3,c2),x),x).
 2519 [2479,4,11]   d(i(i(x,y),i(z,x)),i(z,x)).
 2520 [2479,4,9]    d(i(i(x,y),x),x).
 2845 [2520,4,47]   d(i(i(c2,x),c2),i(i(c3,y),i(z,y))).
35207 [2845,2,2519] d(i(c3,x),i(c2,x)).
35208 [35207,7]     F.
```

◁

Lemma A.4 (Second Congruence Lemma for $\rightsquigarrow$)
From the axioms (12) and (13) we prove the second congruence lemma (20) for $\rightsquigarrow$:
$$\forall p,q\ (p \Vdash^2 q \land q \Vdash^2 p) \Rightarrow \forall s\ s \rightsquigarrow p \Vdash^2 s \rightsquigarrow p).$$
As lemmas we use again reflexivity and transitivity of $\Vdash^2$.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
% negated theorem.
-(all p q (d(p,q)&d(q,p)-> (all s d(i(s,p),i(s,q))))).
end_of_list.
-------> sos clausifies to:
1     d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2    -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3     d(x9,x9).
4    -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5     d(c3,c2).
6     d(c2,c3).
7    -d(i(c1,c3),i(c1,c2)).
----> UNIT CONFLICT at 10394.18 sec ----> 126271 [126270,7] F.
---------------- PROOF ----------------
     9 [2,3,1]          d(i(i(x,y),x),i(z,x)).
    11 [9,4,1]          d(i(i(x,y),i(z,x)),i(u,i(z,x))).
    13 [9,2,9]          d(x,i(y,y)).
    16 [13,2,1]         d(i(i(x,x),y),i(z,y)).
    18 [16,4,1]         d(i(i(x,y),x),i(z,i(u,x))).
    31 [18,2,9]         d(x,i(y,i(z,y))).
    34 [31,2,31]        d(x,i(y,x)).
    38 [34,4,5]         d(c3,i(x,c2)).
    40 [34,4,1]         d(x,i(i(x,y),i(z,y))).
    43 [34,2,5]         d(x,i(c3,c2)).
    45 [34,2,1]         d(i(i(x,i(y,z)),y),i(u,y)).
    51 [38,2,34]        d(x,i(c3,i(y,c2))).
    55 [43,2,1]         d(i(i(c3,c2),x),i(y,x)).
    83 [51,2,1]         d(i(i(c3,i(x,c2)),y),i(z,y)).
   327 [11,2,40]        d(x,i(i(i(i(y,z),u),z),i(y,z))).
   450 [327,2,327]      d(i(i(i(x,y),z),y),i(x,y)).
   498 [450,4,55]       d(i(i(i(i(c3,c2),x),y),x),i(z,x)).
   528 [498,2,9]        d(x,i(i(i(c3,c2),y),y)).
   534 [528,2,528]      d(i(i(c3,c2),x),x).
   564 [534,4,83]       d(i(i(c3,i(x,c2)),y),y).
   568 [534,4,45]       d(i(i(x,i(y,z)),y),y).
   574 [534,4,9]        d(i(i(x,y),x),x).
   829 [574,4,450]      d(i(i(i(i(x,y),x),z),x),x).
   905 [574,2,1]        d(i(x,i(x,y)),i(z,i(x,y))).
  5733 [905,4,829]      d(i(x,i(x,y)),i(x,y)).
  6649 [5733,4,1]       d(i(i(x,y),z),i(i(z,x),x)).
 16808 [6649,4,564]     d(i(i(i(x,c2),y),c3),i(x,c2)).
 16814 [6649,4,1]       d(i(i(x,y),z),i(i(x,z),i(u,z))).
 16845 [6649,2,568]     d(i(x,i(y,i(x,z))),i(y,i(x,z))).
124104 [16845,2,16814]  d(i(x,y),i(i(i(x,z),y),y)).
126270 [124104,4,16808] d(i(x,c3),i(x,c2)).
126271 [126270,7]       F.
```

$\triangleleft$

Lemma A.5 (Congruence Lemma for $\Box$)
From the axioms (12) to (15) we prove the congruence lemma (21) for $\Box$:

$$\forall p, q\ (p \models^2 q \land q \models^2 p) \Rightarrow \Box p \models^2 \Box q.$$

Reflexivity and transitivity of $\models^2$ are again used as lemmas.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
all p ((all x d(x,p))-> (all x d(x,b(p)))).
all p q d(b(i(p,q)),i(b(p),b(q))).
% negated theorem.
-(all p q (d(p,q)&d(q,p)->d(b(p),b(q)))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5  -d(f1(x13),x13)|d(x,b(x13)).
6   d(b(i(x14,x15)),i(b(x14),b(x15))).
7   d(c2,c1).
8   d(c1,c2).
9  -d(b(c2),b(c1)).
----> UNIT CONFLICT at   0.71 sec ----> 209 [208,9] F.
 11 [2,3,1]     d(i(i(x,y),x),i(z,x)).
 15 [11,2,11]   d(x,i(y,y)).
 19 [15,2,1]    d(i(i(x,x),y),i(z,y)).
 27 [19,4,1]    d(i(i(x,y),x),i(z,i(u,x))).
 90 [27,2,11]   d(x,i(y,i(z,y))).
 94 [90,2,90]   d(x,i(y,x)).
103 [94,2,8]    d(x,i(c1,c2)).
104 [94,2,7]    d(x,i(c2,c1)).
114 [103,5]     d(x,b(i(c1,c2))).
117 [104,5]     d(x,b(i(c2,c1))).
122 [114,4,6]   d(x,i(b(c1),b(c2))).
126 [117,4,6]   d(x,i(b(c2),b(c1))).
197 [122,2,122] d(b(c1),b(c2)).
208 [126,2,197] d(b(c2),b(c1)).
209 [208,9]     F.
```

$\triangleleft$

Lemma A.6 (Upwards Monotonicity Lemma for $\rightsquigarrow$)
From the axioms (12) and (13) we prove the upward monotonicity lemma for $\rightsquigarrow$ (36):

$$p \mathrel{|\!\!\!\overset{2}{}} q \Rightarrow \forall s\ (s \rightsquigarrow p) \mathrel{|\!\!\!\overset{2}{}} (s \rightsquigarrow q).$$

Reflexivity and transitivity of $\mathrel{|\!\!\!\overset{2}{}}$ are again used as lemmas.

```
set(hyper_res).
assign(max_weight,13).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
% negated theorem.
-(all p q (d(p,q)-> (all s d(i(s,p),i(s,q))))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5   d(c3,c2).
6  -d(i(c1,c3),i(c1,c2)).
----> UNIT CONFLICT at 1041.75 sec ----> 54166 [54165,6] F.
    8 [2,3,1]        d(i(i(x,y),x),i(z,x)).
   10 [8,4,1]        d(i(i(x,y),i(z,x)),i(u,i(z,x))).
   12 [8,2,8]        d(x,i(y,y)).
   15 [12,2,1]       d(i(i(x,x),y),i(z,y)).
   17 [15,4,1]       d(i(i(x,y),x),i(z,i(u,x))).
   30 [17,2,8]       d(x,i(y,i(z,y))).
   33 [30,2,30]      d(x,i(y,x)).
   36 [33,4,5]       d(c3,i(x,c2)).
   38 [33,4,1]       d(x,i(i(x,y),i(z,y))).
   40 [33,2,5]       d(x,i(c3,c2)).
   42 [33,2,1]       d(i(i(x,i(y,z)),y),i(u,y)).
   45 [36,2,33]      d(x,i(c3,i(y,c2))).
   47 [40,2,1]       d(i(i(c3,c2),x),i(y,x)).
   71 [45,2,1]       d(i(i(c3,i(x,c2)),y),i(z,y)).
  345 [10,2,38]      d(x,i(i(i(i(y,z),u),z),i(y,z))).
  491 [345,2,345]    d(i(i(i(x,y),z),y),i(x,y)).
  528 [491,4,47]     d(i(i(i(i(c3,c2),x),y),x),i(z,x)).
  563 [528,2,8]      d(x,i(i(i(c3,c2),y),y)).
  573 [563,2,563]    d(i(i(c3,c2),x),x).
  592 [573,4,71]     d(i(i(c3,i(x,c2)),y),y).
  594 [573,4,42]     d(i(i(x,i(y,z)),y),y).
  600 [573,4,8]      d(i(i(x,y),x),x).
  714 [600,4,491]    d(i(i(i(i(x,y),x),z),x),x).
  775 [600,2,1]      d(i(x,i(x,y)),i(z,i(x,y))).
 3013 [775,4,714]    d(i(x,i(x,y)),i(x,y)).
 3642 [3013,4,1]     d(i(i(x,y),z),i(i(z,x),x)).
 7874 [3642,4,592]   d(i(i(i(x,c2),y),c3),i(x,c2)).
 7882 [3642,4,1]     d(i(i(x,y),z),i(i(x,z),i(u,z))).
 7900 [3642,2,594]   d(i(x,i(y,i(x,z))),i(y,i(x,z))).
53109 [7900,2,7882]  d(i(x,y),i(i(i(x,z),y),y)).
54165 [53109,4,7874] d(i(x,c3),i(x,c2)).
54166 [54165,6]      F.
```

◁

Lemma A.7 (Downwards Monotonicity Lemma for $\rightsquigarrow$)
From the axioms (12) and (13) we prove the downward monotonicity lemma for $\rightsquigarrow$ (37):

$$p \Vdash^2 q \Rightarrow \forall s\ (q \rightsquigarrow s) \Vdash^2 (p \rightsquigarrow s).$$
As lemmas we use again reflexivity and transitivity of $\Vdash^2$.

```
set(hyper_res).
assign(max_weight,13).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s)))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
% negated theorem.
-(all p q (d(p,q)-> (all s d(i(q,s),i(p,s))))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5   d(c3,c2).
6  -d(i(c2,c1),i(c3,c1)).
----> UNIT CONFLICT at  82.77 sec ----> 11511 [11510,6] F.
    8 [2,3,1]     d(i(i(x,y),x),i(z,x)).
   10 [8,4,1]     d(i(i(x,y),i(z,x)),i(u,i(z,x))).
   12 [8,2,8]     d(x,i(y,y)).
   15 [12,2,1]    d(i(i(x,x),y),i(z,y)).
   17 [15,4,1]    d(i(i(x,y),x),i(z,i(u,x))).
   30 [17,2,8]    d(x,i(y,i(z,y))).
   33 [30,2,30]   d(x,i(y,x)).
   36 [33,4,5]    d(c3,i(x,c2)).
   38 [33,4,1]    d(x,i(i(x,y),i(z,y))).
   40 [33,2,5]    d(x,i(c3,c2)).
   44 [36,4,1]    d(c3,i(i(c2,x),i(y,x))).
   47 [40,2,1]    d(i(i(c3,c2),x),i(y,x)).
  345 [10,2,38]   d(x,i(i(i(i(y,z),u),z),i(y,z))).
  491 [345,2,345] d(i(i(i(x,y),z),y),i(x,y)).
  528 [491,4,47]  d(i(i(i(i(c3,c2),x),y),x),i(z,x)).
  563 [528,2,8]   d(x,i(i(i(c3,c2),y),y)).
  573 [563,2,563] d(i(i(c3,c2),x),x).
  599 [573,4,10]  d(i(i(x,y),i(z,x)),i(z,x)).
  600 [573,4,8]   d(i(i(x,y),x),x).
  750 [600,4,44]  d(i(i(c3,x),c3),i(i(c2,y),i(z,y))).
11510 [750,2,599] d(i(c2,x),i(c3,x)).
11511 [11510,6]   F.
```

$\triangleleft$

Lemma A.8 (Upwards Monotonicity Lemma for $\Box$)
From the axioms (12) to (15) we prove the upward monotonicity lemma for $\Box$ (38):
$$p \Vdash^2 q \Rightarrow \Box p \Vdash^2 \Box q.$$
Reflexivity and transitivity of $\Vdash^2$ are used as lemmas.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
all p ((all x d(x,p))-> (all x d(x,b(p)))).
all p q d(b(i(p,q)),i(b(p),b(q))).
% negated theorem.
```

```
-(all p q (d(p,q)->d(b(p),b(q)))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5  -d(f1(x13),x13)|d(x,b(x13)).
6   d(b(i(x14,x15)),i(b(x14),b(x15))).
7   d(c2,c1).
8  -d(b(c2),b(c1)).
----> UNIT CONFLICT at   0.54 sec ----> 170 [169,8] F.
 10 [2,3,1]     d(i(i(x,y),x),i(z,x)).
 14 [10,2,10]   d(x,i(y,y)).
 18 [14,2,1]    d(i(i(x,x),y),i(z,y)).
 26 [18,4,1]    d(i(i(x,y),x),i(z,i(u,x))).
 89 [26,2,10]   d(x,i(y,i(z,y))).
 93 [89,2,89]   d(x,i(y,x)).
101 [93,2,7]    d(x,i(c2,c1)).
108 [101,5]     d(x,b(i(c2,c1))).
113 [108,4,6]   d(x,i(b(c2),b(c1))).
169 [113,2,113] d(b(c2),b(c1)).
170 [169,8]     F.
```

$\triangleleft$

Lemma A.9 (Basis for Transformer $\mathcal{T}_4$)
From the monotonicity properties (40) and (39) we prove
$w \models f(\vec{p},\vec{q}) \Leftrightarrow \forall\vec{x} \bigwedge_i (\forall v{:}W\ v \models x_i \Rightarrow v \models p_i) \Rightarrow \exists\vec{y} \bigwedge_j (\forall v{:}W\ v \models y_j \Rightarrow v \models q_j) \wedge w \models f(\vec{x},\vec{y})$.
It is sufficient to prove it for two arguments of f, the first one being downward monotonic and the second one being upward monotonic. 'm(w,p)' means $w \models p$.
This time we use binary resolution and the set of support strategy.

```
set(binary_res).
set(factor).
formula_list(usable).
all p q ((all w (m(w,p)->m(w,q)))-> (all r w (m(w,f(r,p))->m(w,f(r,q))))).
all p q ((all w (m(w,p)->m(w,q)))-> (all r w (m(w,f(q,r))->m(w,f(p,r))))).
end_of_list.
-------> usable clausifies to:
 1   m(f1(x1,x2),x1)| -m(w,f(x3,x1))|m(w,f(x3,x2)).
 2  -m(f1(x1,x2),x2)| -m(w,f(x3,x1))|m(w,f(x3,x2)).
 3   m(f2(x4,x5),x4)| -m(w,f(x5,x6))|m(w,f(x4,x6)).
 4  -m(f2(x4,x5),x5)| -m(w,f(x5,x6))|m(w,f(x4,x6)).
formula_list(sos).
% negated theorem.
-(all p q w (m(w,f(p,q)) <->
   (all x ((all v (m(v,x)->m(v,p))) ->
     (exists y ((all v (m(v,y)->m(v,q))) & m(w,f(x,y)))))))).
end_of_list.
-------> sos clausifies to:
 5   m(c2,f(c4,c3))|m(f3(x),x)| -m(v,f4(x))|m(v,c3).
 6   m(c2,f(c4,c3))|m(f3(x),x)|m(c2,f(x,f4(x))).
 7   m(c2,f(c4,c3))| -m(f3(x),c4)| -m(v,f4(x))|m(v,c3).
 8   m(c2,f(c4,c3))| -m(f3(x),c4)|m(c2,f(x,f4(x))).
 9  -m(c2,f(c4,c3))| -m(x7,c1)|m(x7,c4).
10  -m(c2,f(c4,c3))|m(f5(y),y)| -m(c2,f(c1,y)).
11  -m(c2,f(c4,c3))| -m(f5(y),c3)| -m(c2,f(c1,y)).
----> UNIT CONFLICT at  10.65 sec ----> 1090 [1089,164] F.
---------------- PROOF ----------------
```

```
  47 [11,10]   -m(c2,f(c4,c3))| -m(c2,f(c1,c3)).
  60 [47,4]    -m(c2,f(c4,c3))| -m(f2(c1,x),x)| -m(c2,f(x,c3)).
  61 [47,3]    -m(c2,f(c4,c3))|m(f2(c1,x),c1)| -m(c2,f(x,c3)).
  66 [60]      -m(c2,f(c4,c3))| -m(f2(c1,c4),c4).
  67 [61]      -m(c2,f(c4,c3))|m(f2(c1,c4),c1).
  80 [66,9]    -m(c2,f(c4,c3))| -m(f2(c1,c4),c1).
  90 [80,67]   -m(c2,f(c4,c3)).
  93 [90,2]    -m(f1(x,c3),c3)| -m(c2,f(c4,x)).
  94 [90,1]     m(f1(x,c3),x)| -m(c2,f(c4,x)).
 119 [6,90]     m(f3(x),x)|m(c2,f(x,f4(x))).
 144 [8,90]    -m(f3(x),c4)|m(c2,f(x,f4(x))).
 156 [144,119]  m(c2,f(c4,f4(c4))).
 163 [156,94]   m(f1(f4(c4),c3),f4(c4)).
 164 [156,93]  -m(f1(f4(c4),c3),c3).
 455 [5,90]     m(f3(x),x)| -m(y,f4(x))|m(y,c3).
 546 [455,163]  m(f3(c4),c4)|m(f1(f4(c4),c3),c3).
 616 [546,164]  m(f3(c4),c4).
 844 [7,90]    -m(f3(x),c4)| -m(y,f4(x))|m(y,c3).
 968 [844,616] -m(x,f4(c4))|m(x,c3).
1089 [968,163]  m(f1(f4(c4),c3),c3).
1090 [1089,164] F.
```

$\triangleleft$

Lemma A.10 (Intersection Closure Lemma for $\rightsquigarrow$)
From (11) to (13) we prove the closure lemma (50) for $\rightsquigarrow$:

$$\forall p, q, r, w\ w \Vdash^2 (r \rightsquigarrow p) \wedge w \Vdash^2 (r \rightsquigarrow q) \Rightarrow \exists s\ s \Vdash^2 p \wedge s \Vdash^2 q \wedge w \Vdash^2 (r \rightsquigarrow s).$$

Reflexivity and transitivity of $\Vdash^2$ and the monotonicity properties for $\rightsquigarrow$ are used as lemmas.

```
set(hyper_res).
assign(max_weight,13).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
all p d(bot,p).
all p q (d(q,p)-> (all s d(i(s,q),i(s,p)))).
all p q (d(p,q)-> (all s d(i(q,s),i(p,s)))).
% negated theorem.
-(all p q r w (d(w,i(r,p))&d(w,i(r,q))->
   (exists s (d(s,p)&d(s,q)&d(w,i(r,s)))))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5   d(bot,x13).
6  -d(x14,x15)|d(i(x16,x14),i(x16,x15)).
7  -d(x17,x18)|d(i(x18,x19),i(x17,x19)).
8   d(c1,i(c2,c4)).
9   d(c1,i(c2,c3)).
10  -d(x20,c4)| -d(x20,c3)| -d(c1,i(c2,x20)).
-----> EMPTY CLAUSE at 4234.45 sec ----> 8834 [8482,10,8384,5493] F.
---------------- PROOF ----------------
  13 [6,5]         d(i(x,bot),i(x,y)).
  14 [13,6]        d(i(x,i(y,bot)),i(x,i(y,z))).
  17 [7,13]        d(i(i(x,y),z),i(i(x,bot),z)).
  22 [7,5]         d(i(x,y),i(bot,y)).
  25 [22,4,13]      d(i(x,bot),i(bot,y)).
  93 [17,4,13]      d(i(i(x,y),bot),i(i(x,bot),z)).
 129 [1,4,17]       d(i(i(x,y),z),i(i(z,bot),i(u,x))).
 148 [2,25,1]       d(i(i(bot,x),y),i(z,y)).
 150 [2,13,1]       d(i(i(x,y),x),i(z,x)).
 151 [148,7]        d(i(i(x,y),z),i(i(i(bot,u),y),z)).
 161 [148,4,1]      d(i(i(x,y),bot),i(z,i(u,x))).
 169 [150,4,93]     d(i(i(x,y),bot),i(z,x)).
 226 [161,2,150]    d(x,i(y,i(z,y))).
 235 [226,2,226]    d(x,i(y,x)).
 237 [235,6]        d(i(x,y),i(x,i(z,y))).
 266 [235,4,1]      d(x,i(i(x,y),i(z,y))).
 381 [237,7]        d(i(i(x,i(y,z)),u),i(i(x,z),u)).
 800 [151,2,150]    d(x,i(i(i(bot,y),z),z)).
 813 [800,2,800]    d(i(i(bot,x),y),y).
 816 [813,6]        d(i(x,i(i(bot,y),z)),i(x,z)).
 848 [813,4,169]    d(i(i(x,y),bot),x).
 852 [813,4,150]    d(i(i(x,y),x),x).
 914 [848,7]        d(i(x,y),i(i(i(x,z),bot),y)).
 954 [852,7]        d(i(x,y),i(i(i(x,z),x),y)).
 964 [852,4,129]    d(i(i(x,y),i(z,x)),i(z,x)).
1340 [816,4,266]    d(x,i(i(x,y),y)).
1353 [816,4,129]    d(i(i(x,y),z),i(i(z,bot),x)).
1361 [816,4,1]      d(i(i(x,y),z),i(i(z,x),x)).
2816 [914,4,1340]   d(x,i(i(i(i(x,y),z),bot),y)).
3735 [954,4,9]      d(c1,i(i(i(c2,x),c2),c3)).
3736 [954,4,8]      d(c1,i(i(i(c2,x),c2),c4)).
3901 [964,4,954]    d(i(x,i(y,i(x,z))),i(y,i(x,z))).
3911 [964,4,14]     d(i(i(x,y),i(z,bot)),i(z,x)).
4150 [1361,4,3736] d(c1,i(i(c4,i(c2,x)),i(c2,x))).
4151 [1361,4,3735] d(c1,i(i(c3,i(c2,x)),i(c2,x))).
5493 [3911,4,2816] d(x,i(y,i(i(x,i(y,bot)),z))).
6675 [381,4,4151]   d(c1,i(i(c3,x),i(c2,x))).
6676 [381,4,4150]   d(c1,i(i(c4,x),i(c2,x))).
6754 [6675,4,237]   d(c1,i(i(c3,x),i(y,i(c2,x)))).
6781 [6676,4,237]   d(c1,i(i(c4,x),i(y,i(c2,x)))).
7782 [3901,2,6781] d(i(c4,x),i(c1,i(c2,x))).
7783 [3901,2,6754] d(i(c3,x),i(c1,i(c2,x))).
8384 [7782,2,1353] d(i(i(c1,i(c2,x)),bot),c4).
8482 [7783,2,1353] d(i(i(c1,i(c2,x)),bot),c3).
8834 [8482,10,8384,5493] F.
```

◁

Lemma A.11 (Union Closure Lemma for $\rightsquigarrow$)
From (11) to (13) we prove the closure lemma (51) for $\rightsquigarrow$.

$$\forall p, q, r, w\ w \Vdash^2 (p \rightsquigarrow r) \wedge w \Vdash^2 (q \rightsquigarrow r) \Rightarrow \exists s\ p \Vdash^2 s \wedge q \Vdash^2 s \wedge w \Vdash^2 (s \rightsquigarrow r).$$

As lemmas we use reflexivity and transitivity of $\Vdash^2$ and the monotonicity properties for $\rightsquigarrow$.

```
set(hyper_res).
assign(max_weight,13).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
all p q (d(q,p)-> (all s d(i(s,q),i(s,p)))).
all p q (d(p,q)-> (all s d(i(q,s),i(p,s)))).
% negated theorem.
-(all p q r x (d(x,i(p,r))&d(x,i(q,r))->
    (exists s (d(p,s)&d(q,s)&d(x,i(s,r)))))).
end_of_list.
-------> sos clausifies to:
1  d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3  d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5  -d(x13,x14)|d(i(x15,x13),i(x15,x14)).
6  -d(x16,x17)|d(i(x17,x18),i(x16,x18)).
7  d(c1,i(c4,c2)).
8  d(c1,i(c3,c2)).
9  -d(c4,x19)| -d(c3,x19)| -d(c1,i(x19,c2)).
-----> EMPTY CLAUSE at  15.92 sec ----> 819 [726,9,728,727] F.
---------------- PROOF ----------------
 16 [6,8]       d(i(i(c3,c2),x),i(c1,x)).
 17 [6,7]       d(i(i(c4,c2),x),i(c1,x)).
 26 [1,4,17]    d(i(i(c2,x),c4),i(c1,i(y,c2))).
 27 [1,4,16]    d(i(i(c2,x),c3),i(c1,i(y,c2))).
 32 [2,3,1]     d(i(i(x,y),x),i(z,x)).
 33 [32,6]      d(i(i(x,y),z),i(i(i(y,u),y),z)).
 38 [32,4,27]   d(i(i(c3,x),c3),i(c1,i(y,c2))).
 39 [32,4,26]   d(i(i(c4,x),c4),i(c1,i(y,c2))).
 45 [32,2,32]   d(x,i(y,y)).
 47 [45,6]      d(i(i(x,x),y),i(z,y)).
 61 [47,4,1]    d(i(i(x,y),x),i(z,i(u,x))).
 93 [61,2,32]   d(x,i(y,i(z,y))).
 98 [93,2,93]   d(x,i(y,x)).
112 [98,4,39]   d(c4,i(c1,i(x,c2))).
113 [98,4,38]   d(c3,i(c1,i(x,c2))).
114 [98,4,1]    d(x,i(i(x,y),i(z,y))).
467 [33,2,32]   d(x,i(i(i(y,z),y),y)).
479 [467,2,467] d(i(i(x,y),x),x).
480 [479,6]     d(i(x,y),i(i(i(x,z),x),y)).
681 [480,4,479] d(i(x,i(x,y)),i(x,y)).
726 [681,4,114] d(x,i(i(x,y),y)).
727 [681,4,113] d(c3,i(c1,c2)).
728 [681,4,112] d(c4,i(c1,c2)).
819 [726,9,728,727] F.
```

◁

Lemma A.12 (Closure Lemma (47) for □)

From (11) to (15) we prove the closure lemma (47) for □.

$$\forall p, q, w\ w \Vdash^2 \Box p \wedge w \Vdash^2 \Box q \Rightarrow \exists s\ s \Vdash^2 p \wedge s \Vdash^2 q \wedge w \Vdash^2 \Box s.$$

Reflexivity and transitivity of $\Vdash^2$ and the monotonicity properties for $\rightsquigarrow$ and □ are used as lemmas.

```
set(hyper_res).
assign(max_weight,13).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
all p d(bot,p).
all p q (d(p,q)-> (all s d(i(s,p),i(s,q)))).
all p q (d(p,q)-> (all s d(i(q,s),i(p,s)))).
all p ((all x d(x,p))-> (all x d(x,b(p)))).
all p q d(b(i(p,q)),i(b(p),b(q))).
all p q (d(p,q)->d(b(p),b(q))).
% negated theorem.
```

```
-(all p q w (d(w,b(p))&d(w,b(q))-> (exists s (d(s,p)&d(s,q)&d(w,b(s)))))).
end_of_list.
-------> sos clausifies to:
 1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
 2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
 3   d(x9,x9).
 4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
 5   d(bot,x13).
 6  -d(x14,x15)|d(i(x16,x14),i(x16,x15)).
 7  -d(x17,x18)|d(i(x18,x19),i(x17,x19)).
 8  -d(f1(x20),x20)|d(x,b(x20)).
 9   d(b(i(x21,x22)),i(b(x21),b(x22))).
10  -d(x23,x24)|d(b(x23),b(x24)).
11   d(c1,b(c3)).
12   d(c1,b(c2)).
13  -d(x25,c3)| -d(x25,c2)| -d(c1,b(x25)).
-----> EMPTY CLAUSE at 14759.62 sec ----> 11001 [11000,13,7739,7383] F.
---------------- PROOF ----------------
   33 [6,5]          d(i(x,bot),i(x,y)).
   45 [7,33]         d(i(i(x,y),z),i(i(x,bot),z)).
   56 [7,12]         d(i(b(c2),x),i(c1,x)).
   58 [7,5]          d(i(x,y),i(bot,y)).
   62 [58,4,33]      d(i(x,bot),i(bot,y)).
  134 [9,4,56]       d(b(i(c2,x)),i(c1,b(x))).
  158 [134,6]        d(i(x,b(i(c2,y))),i(x,i(c1,b(y)))).
  266 [45,4,33]      d(i(i(x,y),bot),i(i(x,bot),z)).
  579 [1,4,45]       d(i(i(x,y),z),i(i(z,bot),i(u,x))).
  617 [2,62,1]       d(i(i(bot,x),y),i(z,y)).
  621 [2,33,1]       d(i(i(x,y),x),i(z,x)).
  623 [617,7]        d(i(i(x,y),z),i(i(i(bot,u),y),z)).
  639 [617,4,1]      d(i(i(x,y),bot),i(z,i(u,x))).
  690 [621,4,266]    d(i(i(x,y),bot),i(z,x)).
  711 [621,2,621]    d(x,i(y,y)).
  715 [711,7]        d(i(i(x,x),y),i(z,y)).
  830 [715,4,33]     d(i(i(x,x),bot),i(y,z)).
 1171 [639,2,621]    d(x,i(y,i(z,y))).
 1198 [1171,2,1171]  d(x,i(y,x)).
 1365 [1198,4,1]     d(x,i(i(x,y),i(z,y))).
 1450 [1198,2,1]     d(i(i(x,i(y,z)),y),i(u,y)).
 1879 [1365,4,45]    d(x,i(i(x,bot),i(y,z))).
 3472 [623,2,621]    d(x,i(i(i(bot,y),z),z)).
 3494 [3472,2,3472]  d(i(i(bot,x),y),y).
 3498 [3494,6]       d(i(x,i(i(bot,y),z)),i(x,z)).
 3590 [3494,4,1450]  d(i(i(x,i(y,z)),y),y).
 3623 [3494,4,830]   d(i(i(x,x),bot),y).
 3646 [3494,4,690]   d(i(i(x,y),bot),x).
 3662 [3494,4,621]   d(i(i(x,y),x),x).
 3698 [3623,6]       d(i(x,i(i(y,y),bot)),i(x,z)).
 3775 [3646,7]       d(i(x,y),i(i(i(x,z),bot),y)).
 3776 [3646,6]       d(i(x,i(i(y,z),bot)),i(x,y)).
 5420 [3498,4,1879]  d(x,i(i(x,bot),y)).
 5434 [3498,4,1365]  d(x,i(i(x,y),y)).
 5438 [3498,4,579]   d(i(i(x,y),z),i(i(z,bot),x)).
 5471 [5420,7]       d(i(i(i(x,bot),y),z),i(x,z)).
 5768 [5434,10]      d(b(x),b(i(i(x,y),y))).
 6822 [5768,4,11]    d(c1,b(i(i(c3,x),x))).
 7383 [3698,4,3590]  d(i(i(x,i(y,z)),i(i(u,u),bot)),y).
 7455 [3775,10]      d(b(i(x,y)),b(i(i(i(x,z),bot),y))).
 7480 [3775,4,3662]  d(i(x,i(x,y)),i(x,y)).
 7513 [7480,4,158]   d(i(c1,b(i(c2,x))),i(c1,b(x))).
 7739 [3776,4,3662]  d(i(i(x,y),i(i(x,z),bot)),x).
 8290 [5438,10]      d(b(i(i(x,y),z)),b(i(i(z,bot),x))).
 8487 [5471,10]      d(b(i(i(i(x,bot),y),z)),b(i(x,z))).
10097 [7455,4,6822]  d(c1,b(i(i(i(i(c3,x),y),bot),x))).
10955 [8290,4,10097] d(c1,b(i(i(x,bot),i(i(c3,x),y)))).
10989 [8487,4,10955] d(c1,b(i(x,i(i(c3,i(x,bot)),y)))).
11000 [10989,2,7513] d(c1,b(i(i(c3,i(c2,bot)),x))).
11001 [11000,13,7739,7383] F.
```

$\triangleleft$

Lemma A.13 (Seriality for $\rightsquigarrow$)
We prove $\forall w, p\ \exists q\ w \Vdash^2 p \rightsquigarrow q$.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
% negated theorem.
-(all w p exists q d(w,i(p,q))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5  -d(c2,i(c1,x13)).
----> UNIT CONFLICT at   0.06 sec ----> 12 [11,5] F.
 7 [2,3,1] d(i(i(x,y),x),i(z,x)).
11 [7,2,7] d(x,i(y,y)).
12 [11,5]  F.
```

$\triangleleft$

Lemma A.14 (Seriality for $\Box$)
We prove $\forall w\ \exists q\ w \Vdash^2 \Box q$.

```
set(hyper_res).
formula_list(sos).
all r p q s d(i(i(p,q),r),i(i(r,p),i(s,p))).
all p q r s (d(p,q)&d(i(p,q),i(r,s))->d(r,s)).
all p d(p,p).
all p q r (d(p,q)&d(q,r)->d(p,r)).
all p ((all x d(x,p))-> (all x d(x,b(p)))).
all p q d(b(i(p,q)),i(b(p),b(q))).
all p q (d(p,q)->d(b(p),b(q))).
-(all w exists p d(w,b(p))).
end_of_list.
-------> sos clausifies to:
1   d(i(i(x1,x2),x3),i(i(x3,x1),i(x4,x1))).
2  -d(x5,x6)| -d(i(x5,x6),i(x7,x8))|d(x7,x8).
3   d(x9,x9).
4  -d(x10,x11)| -d(x11,x12)|d(x10,x12).
5  -d(f1(x13),x13)|d(x,b(x13)).
6   d(b(i(x14,x15)),i(b(x14),b(x15))).
7  -d(x16,x17)|d(b(x16),b(x17)).
8  -d(c1,b(x18)).
----> UNIT CONFLICT at   0.11 sec ----> 23 [22,8] F.
14 [2,3,1]   d(i(i(x,y),x),i(z,x)).
```

```
19 [14,2,14] d(x,i(y,y)).
22 [19,5]    d(x,b(i(y,y))).
23 [22,8]    F.
```

◁

Lemma A.15 (Collapsing Lemmas for $\rightsquigarrow$)
We prove from (64) and (65) the general reflexivity $\forall x\ \mathcal{R}(x,x,x)$. Instead of the built–in equality handling we use equality axioms together with the unit resulting (UR) strategy. Other parameter settings did not work.

```
set(ur_res).
formula_list(sos).
all w exists x (R(x,w,w)& (all u v (R(x,u,v)->u=v))).
all w a b c d (R(w,a,b)&R(b,c,d)->
   (exists u v (R(w,u,v)&R(a,v,d)& (all x y (R(u,x,y)->x=d))))).
% equality axioms
all x (x=x).
all x y z u v w (x=y&z=u&v=w&R(x,z,v)->R(y,u,w)).
% negated theorem
-(all x R(x,x,x)).
end_of_list.
-------> sos clausifies to:
1  R(f(w),w,w).
2  -R(f(w),u,v)|u=v.
3  -R(w,x1,x2)| -R(x2,x3,x4)|R(w,h(w,x1,x2,x3,x4),g(w,x1,x2,x3,x4)).
4  -R(w,x1,x2)| -R(x2,x3,x4)|R(x1,g(w,x1,x2,x3,x4),x4).
5  -R(w,x1,x2)| -R(x2,x3,x4)| -R(h(w,x1,x2,x3,x4),x,y)|x=x4.
6  x=x.
7  x!=y|z!=u|v!=w| -R(x,z,v)|R(y,u,w).
8  -R(c,c,c).
----> UNIT CONFLICT at   1.89 sec ----> 297 [296,287] F.
abbreviations:
    f(f(x))               := k(x)
    g(k(x),f(x),f(x),x,x) := g1(x)
    f(c)                  := d
    f(d) = f(f(c))        := e = k(c)
---------------- PROOF ----------------
  9 [4,1,1]         R(f(x),g1(x),x).
 13 [9,2]           g1(x)=x.
 15 [7,13,6,9,8]    d!=c.
 24 [7,6,13,13,1]   R(f(g1(x)),x,x).
 27 [15,2]         -R(f(x),d,c).
 30 [27,7,6,6,1]    c!=d.
192 [5,1,1,30]     -R(h(f(e),e,e,d,d),c,x).
244 [192,7,13,6,9]  d!=h(f(e),e,e,d,d).
258 [244,2]        -R(f(x),d,h(f(e),e,e,d,d)).
261 [258,7,6,6,24]  h(f(e),e,e,d,d)!=d.
279 [261,2]        -R(f(x),h(f(e),e,e,d,d),d).
287 [279,7,6,6,13] -R(f(x),h(f(e),e,e,d,d),g(f(e),e,e,d,d)).
296 [3,1,1]         R(k(x),h(k(x),f(x),f(x),x,x),g1(x)).
297 [296,287]       F.
```

Now we prove from (64), (65) and the general reflexivity that the first two arguments of $\mathcal{R}$ collapse; $\forall w,u,v\ \mathcal{R}(w,u,v) \Rightarrow w = u$.

```
set(ur_res).
assign(max_weight,40).
formula_list(sos).
all w exists x (R(x,w,w)& (all u v (R(x,u,v)->u=v))).
all w a b c d (R(w,a,b)&R(b,c,d)->
   (exists u v (R(w,u,v)&R(a,v,d)& (all x y (R(u,x,y)->x=d))))).
all x (x=x).
% equality axioms
all x y z u v w (x=y&z=u&v=w&R(x,z,v)->R(y,u,w)).
all x R(x,x,x).
% negated theorem
-(all w u v (R(w,u,v)->w=u)).
end_of_list.
-------> sos clausifies to:
 1   R(f(w),w,w).
 2  -R(f(w),u,v)|u=v.
 3  -R(w,x1,x2)| -R(x2,x3,x4)|R(w,h(w,x1,x2,x3,x4),g(w,x1,x2,x3,x4)).
 4  -R(w,x1,x2)| -R(x2,x3,x4)|R(x1,g(w,x1,x2,x3,x4),x4).
 5  -R(w,x1,x2)| -R(x2,x3,x4)| -R(h(w,x1,x2,x3,x4),x,y)|x=x4.
 6   x=x.
 7   x!=y|z!=u|v!=w| -R(x,z,v)|R(y,u,w).
 8   R(x,x,x).
 9   R(e,d,c).
10   e!=d.
----> UNIT CONFLICT at 435.17 sec ----> 8504 [binary,8503,7238] F.
---------------- PROOF ----------------
  11 [2,10]             -R(f(x),e,d).
  15 [4,8,1]             R(f(x),g(f(x),f(x),f(x),x,x),x).
  18 [4,1,1]             R(f(x),g(f(f(x)),f(x),f(x),x,x),x).
  69 [15,2]              g(f(x),f(x),f(x),x,x)=x.
  81 [18,2]              g(f(f(x)),f(x),f(x),x,x)=x.
 247 [7,6,6,1,11]        d!=e.
 294 [7,81,81,6,8]       R(x,x,g(f(f(x)),f(x),f(x),x,x)).
 297 [7,69,69,6,8]       R(x,x,g(f(x),f(x),f(x),x,x)).
 463 [294,2]             f(x)=g(f(f(f(x))),f(f(x)),f(f(x)),f(x),f(x)).
1079 [463,7,81,6,18]     R(g(f(f(f(x))),f(f(x)),f(f(x)),f(x),f(x)),x,x).
4609 [5,8,8,247]        -R(h(e,e,e,e,e),d,x).
7001 [5,8,8,297]         h(x,x,x,x,x)=x.
7238 [7001,7,81,6,1079] R(f(h(x,x,x,x,x)),x,h(x,x,x,x,x)).
8481 [4609,7,6,6,9]      e!=h(e,e,e,e,e).
8503 [8481,2]           -R(f(x),e,h(e,e,e,e,e)).
8504 [8503,7238]         F.
```

Now we prove from (64), (65), the general reflexivity, and collapsing of the first two arguments of $\mathcal{R}$, that the last two arguments collapse as well: $\forall w, u, v\ \mathcal{R}(w, u, v) \Rightarrow u = v$.

```
set(ur_res).
assign(max_weight,40).
formula_list(sos).
all w exists x (R(x,w,w)& (all u v (R(x,u,v)->u=v))).
all w a b c d (R(w,a,b)&R(b,c,d)->
   (exists u v (R(w,u,v)&R(a,v,d)& (all x y (R(u,x,y)->x=d))))).
% equality axioms
all x (x=x).
all x y z u v w (x=y&z=u&v=w&R(x,z,v)->R(y,u,w)).

all x R(x,x,x).
all w u v (R(w,u,v)->w=u).
%negated theorem
-(all w u v (R(w,u,v)->u=v)).
end_of_list.
-------> sos clausifies to:
1  R(f(w),w,w).
2  -R(f(w),u,v)|u=v.
3  -R(w,x1,x2)| -R(x2,x3,x4)|R(w,h(w,x1,x2,x3,x4),g(w,x1,x2,x3,x4)).
```

```
4  -R(w,x1,x2)| -R(x2,x3,x4)|R(x1,g(w,x1,x2,x3,x4),x4).
5  -R(w,x1,x2)| -R(x2,x3,x4)| -R(h(w,x1,x2,x3,x4),x,y)|x=x4.
6  x=x.
7  x!=y|z!=u|v!=w| -R(x,z,v)|R(y,u,w).
8  R(x,x,x).
9  -R(w,u,v)|w=u.
10  R(e,d,c).
11  d!=c.
----> UNIT CONFLICT at 555.50 sec ----> 9689 [9688,8630] F.
---------------- PROOF ----------------
  14 [9,1]           f(x)=x.
  15 [2,11]         -R(f(x),d,c).
  21 [4,1,8]         R(x,g(f(x),x,x,x,x),x).
  66 [21,2]          g(f(f(x)),f(x),f(x),f(x),f(x))=f(x).
 275 [7,6,6,10,15]  e!=f(x).
8630 [7,14,66,6,21] R(x,f(x),f(x)).
9688 [275,9]        -R(e,f(x),y).
9689 [9688,8630]    F.
```

◁