A DEDUCTIVE SYSTEM FOR THE SEMANTICS OF INTENSIONAL LOGIC

Frank M. Brown, Nelson Bishop,
Jack Woodward

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

TR-132 April 1983

Table of Contents

1. INTRODUCTION **2**
2. DESCRIPTION OF THE THEOREM PROVER **3**
2.1 Logical Knowledge 3
2.1.1 Assertion Schemata 4
2.1.2 Goal Schemata 4
2.1.3 Replica Creation Schemata 5
2.1.4 The Unification Schema 5
2.1.5 Other Logical Schemata 6
2.2 Theory of Modality 6
2.2.1 Notation 7
2.2.2 The Logical Axioms of Modal Logic 7
2.2.3 The Possibility Problem 7
2.3 A Sequent Calculus for Modal Logic 8
2.3.1 Rules of the Sequent Calculus 8
2.3.2 The Standard Form 9
2.3.3 Completeness and Expressibility 10
2.3.4 Rules Defining Modal Concepts 11
3. RELATIONAL SEMANTICS **12**
3.1 Definitions of Relational Semantics 12
3.2 Basic Theorems of Relational Semantics 12
3.3 Results 19
4. FUNCTIONAL SEMANTICS **21**
4.1 Definitions of Functional Semantics 21
4.2 Basic Theorems of Functional Semantics 22
4.3 Examples 22
4.4 Results 30
5. REFERENCES **32**

ABSTRACT

We give an axiomatization of the modal quantificational logic which captures the notions of logical truth. This modal logic is stronger than S5. Next we describe a sequent calculus for this modal logic, and show that it is complete in the traditional sense.

We then show how two semantical systems for intensional logics can be represented in our modal logic, thus allowing our sequent calculus to prove theorems about the semantics of intensional logics.

Finally we describe an automatic theorem prover for proving theorems about the semantics of intensional logics. These new semantical system involve no set theoretic concepts. Instead, they are based on a formal calculus consisting of a very strong modal logic. We believe this calculus more closely corresponds to intuitive common sense semantical concepts. The effectiveness of this theorem prover, and the semantical systems on which it is built, is exemplified by its proof of a number of theorems in relational and functional semantics of intensional logics.

1. INTRODUCTION

This report describes an implementation of a deductive system for modal logic which is capable of proving theorems about the semantics of intensional logic. This deductive system is similar to earlier theorem provers by Brown and described in [Brown 1, 2, 3, 4, 6, 9, 17, 18]. This theorem prover is based on a modal logic and some semantical systems which have recently been developed in [Brown 7, 8, 12, 16, 19].

This deductive for modal logic system is described in Section 2. In Section 3 we discuss one of these semantical systems: Relational Semantics and list a number of theorems our automatic theorem prover has proven. Some example theorems are also discussed. In Section 4 we discuss the other semantical system: Functional Semantics and list some results and examples. Finally, in Section 5 we compare these two semantical systems and draw a few conclusions.

2. DESCRIPTION OF THE THEOREM PROVER

Our theorem prover consists of an interpreter for mathematical expressions and many items of mathematical knowledge. This interpreter is a fairly complex mechanism, but it may be viewed as applying items of mathematical knowledge of the form: $\phi \leftrightarrow \psi$ or $\phi = \psi$ to the theorem being proven, in the following manner. The interpreter evaluates the theorem recursively in a call-by-need manner. That is, if $(fa_1...a_n)$ is a sub-expression being evaluated, then the interpreter tries to apply its items of knowledge to that sub-expression before evaluating the arguments $a_1...a_n$. for each sub-expression that the interpreter evaluates, in turn it tries to match the ϕ expression of an item to that sub-expression. If, however, during the application process an argument a_i does not match the corresponding argument of the ϕ expression, then a_i is evaluated, and the system then tries to match the result of that evaluation. If ever the interpreter finds a sub-expression $\phi\theta$ which is an instance of ϕ of some item, then it replaces that expression by the corresponding instance $\psi\theta$ of ψ. At this point all memory of the sub-expression $\phi\theta$ is immediately lost and the interpreter now evaluates $\psi\theta$. If no items can be applied to a sub-expression then the sub-expression is not evaluated again but is simply returned.

Sometimes it will be the case that our interpreter will need to use items which are valid only in certain domains Π. In such a case we could represent the item as a conditional item of the form:

$$\Pi x \rightarrow (\phi x \leftrightarrow \psi x)$$
or $$\Pi x \rightarrow (\phi x = \psi x)$$

The interpreter handles conditional items in the same way in which it handles non-conditional items until it has found a $\phi\theta$ which matches the sub-expression being evaluated. At this point on a conditional item, the interpreter tries to match each element in the conjunction Πx with some expression which it believes to be true. If such matches are found with substitution $\theta\sigma$ then $\psi\theta\sigma$ is returned. Otherwise the interpreter tries to apply another item as previously described.

2.1 Logical Knowledge

The symbols of classical logic are listed below with their English translations:

$p \wedge q$	p and q
$p \vee q$	p or q
$p \rightarrow q$	if p then q
$p \leftrightarrow q$	p iff q
$\sim p$	not p
T	true
F	false
$\forall x\ \phi x$	for all objects x, ϕx holds
$\exists x\ \phi x$	for some objects x, ϕx holds
$\forall p\ \phi p$	for all propositions p, ϕp holds
$\exists p\ \phi p$	for some propositions p, ϕp holds.

Letters such as x, y, z, range over objects, whereas letters such as p, q, r, w, u, range over propositions.

Our theorem prover has knowledge about twelve logical symbols which are listed below with their English translations:

=	equal

$\Rightarrow$	implies (this symbol is called a sequent arrow)
and	and (this symbol is used to form an implicit conjunction sequents)

The sequent arrow may be defined as follows:

$$p_1,...,p_n \Rightarrow q_1,...,q_m = \text{df } (p_1 \wedge ... \wedge p_n) \rightarrow (q_1 \vee ... \vee q_m)$$

where p_i and q_j are sentences. Thus a sequent may be thought of as being a database of statements $p_1,...,p_n$ called assertions which occur before the sequent arrow, and statements $q_1,...,q_m$ called goals which occur after the sequent arrow. The implicit conjunction of different sequents may be thought of as being a group of different databases.

The items of logical knowledge, which are all schemata because they involve ellipses (i.e. dots representing arbitrary expressions), are listed below:

2.1.1 Assertion Schemata

$T\Rightarrow$: $(..T..\Rightarrow..) \leftrightarrow (.. ..\Rightarrow..)$

$F\Rightarrow$: $(..F..\Rightarrow..) \leftrightarrow T$

$\sim\ \Rightarrow$: $(..\sim p..\Rightarrow..) \leftrightarrow (.. ..\Rightarrow p..)$

$\wedge \Rightarrow$: $(..p\wedge q..\Rightarrow..) \leftrightarrow (..p,q..\Rightarrow..)$

$\vee \Rightarrow$: $(..p\vee q..\Rightarrow..) \leftrightarrow (..p..\Rightarrow..)$ and $(..q..\Rightarrow..)$

$\rightarrow\ \Rightarrow$: $(..p\rightarrow q..\Rightarrow..) \leftrightarrow (.. ..\Rightarrow p..)$ and $(..q..\Rightarrow..)$

$\leftrightarrow\ \Rightarrow$: $(..p\leftrightarrow q.. ..) \leftrightarrow (..p,q..\Rightarrow..)$ and $(.. .. \Rightarrow p,q..)$

$\exists \Rightarrow$: $(..\exists x\ \phi x..\Rightarrow..) \leftrightarrow (..\phi(f{*}_1..{*}_n)..\Rightarrow..)$
where f is a new skolem function and ${*}_1...{*}_n$ are all the unification variables which occur in ϕx.

$\forall \Rightarrow$: $(..\forall x\ \phi x..\Rightarrow..) \leftrightarrow (..\forall(x^*)\phi x,\phi^*..\Rightarrow..)$
where * is a new unification variable

$= \Rightarrow$: $(\Pi a...\{{a=t \atop t=a}\}..\lceil a\Rightarrow\phi a...\psi a) \leftrightarrow (\Pi t..\lceil t\Rightarrow\phi t..\psi t)$
where a is of the form $(f{*}_1...{*}_n)$ and f is a skolem function not occurring in t.
This is our version of the law of Leibniz.

2.1.2 Goal Schemata

$\Rightarrow T$: $(..\Rightarrow..T..) \leftrightarrow T$

$\Rightarrow F$: $(..\Rightarrow..F..) \leftrightarrow (..\Rightarrow.. ..)$

$\Rightarrow \sim$: $(..\Rightarrow..\sim p..) \leftrightarrow (..p\Rightarrow.. ..)$

$\Rightarrow \wedge$: $(..\Rightarrow..p\wedge q..) \leftrightarrow (..\Rightarrow..p..)$ and $(..\Rightarrow..q..)$

$\Rightarrow \vee$: $(..\Rightarrow..p\vee q..) \leftrightarrow (..\Rightarrow..p,q..)$

$\Rightarrow \rightarrow$: $(..\Rightarrow..p\rightarrow q..) \leftrightarrow (..p\Rightarrow..q..)$

$\Rightarrow \leftrightarrow$: $(..\Rightarrow p\leftrightarrow q..) \leftrightarrow (..p\Rightarrow..q..)$ and $(..q\Rightarrow..p..)$

$\Rightarrow \forall$: $(..\Rightarrow..\forall x\ \phi x..) \leftrightarrow (..\Rightarrow..\phi(f{*}_1...{*}_n)..)$
where f is a new skolem function and ${*}_1...{*}_n$ are all the unification variables which occur in ϕx.

$\Rightarrow \exists$: $(..\Rightarrow..\exists x\ \phi x..) \leftrightarrow (..\Rightarrow..\exists(x^*)\phi x,\phi^*..)$
where * is a new unification variable.

2.1.3 Replica Creation Schemata

$\forall()\Rightarrow$: $(..\forall(x...)\phi x..\Rightarrow..) \leftrightarrow (..\forall(x...*)\phi x,\phi^*..\Rightarrow..)$

where * is a new unification variable and no more than one unification variable occurs in (x...).

$\Rightarrow\exists()$: $(..\Rightarrow..\exists(x...)\phi x..) \leftrightarrow (..\Rightarrow..\exists(x...*)\phi x,\phi^*..)$

where * is a new unification variable and no more than one unification variable occurs in (x...).

The items $\forall()\Rightarrow$ and $\Rightarrow\exists()$ are used to create additional replicas: ϕ^*, a universally quantified assertion $\forall(x...)\phi x$, and an existentially quantified goal $\exists(x...)\phi x$. The replica ϕ^* is exactly like the original formula except that the initial quantifier is deleted and the bound variable associated with that quantifier is replaced by a new free unification variable.

A *Unification Variable* is a free variable which is created by $\forall\Rightarrow$, $\Rightarrow\exists$, $\forall()\Rightarrow$, or $\Rightarrow\exists()$ items, and which may later be instantiated to some term by the unification item (see Section 2.1.4). Unification variables are written as a star sign: * possibly followed by a number such as: *1, *2, *3.

In these four items we have seen formulae of the form $\forall(x...)\phi x$ and $\exists(x...)\phi x$ which are not usually thought of as being well formed sentences of logic. Such formulae should be interpreted as respectively $\forall x\phi x$ and $\exists x\phi x$ which are well formed sentences of logic. The ... list, which is called the *replica instance list*, is used merely to store certain pragmatic information used by the deductive system. This information is basically the list of unification variables (or more precisely the list of instantiations of the unification variables, see Section 2.1.4) that were produced from this quantifier by applications of the $\forall\Rightarrow$, $\Rightarrow\exists$, $\forall()\Rightarrow$, and $\Rightarrow\exists()$ items.

2.1.4 The Unification Schema

$$\text{Unify: } [(..p_1..\Rightarrow..q_1..) \text{ and .. and .. } (..p_n..\Rightarrow..q_n..)] \leftrightarrow [(..p_i..\Rightarrow..q_i) \text{ and .. and .. } (..p_n..\Rightarrow..q_n..)]\ \theta$$

where $1\leq i\leq n$ and θ is any one of the sets of substitutions of terms for unification variables which satisfy both the forcing restriction and the instantiation restriction. These two restrictions are described below.

The *forcing restriction* is the requirement that the substitution makes tautologous the greatest number of sequents starting with the first sequent and progressing towards the nth sequent.

In the case that there actually is some substitution which will make all the sequents tautologous, without further unification variables being created by the $\forall\Rightarrow$ and $\Rightarrow\exists$ items, then θ will be one such substitution. As a minor point, if θ makes all the sequents tautologous, then the unification schema is defined to return.

The *instantiation restriction* is the requirement that no unification variable be instantiated to a term which already occurs in the replica instance list of the quantifier of the given sequent which contains the unification variable. The rationale behind this restriction is that if a term t occurs in the replica instance list of a quantifier such as $\forall$ in $(..\forall x\phi x..\Rightarrow..)$ then the sub-formulae of ϕt must already occur in some sequent which must be proven in order to prove the theorem.

2.1.5 Other Logical Schemata

atom: (..p..⇒..p) ↔ T
and: (..and T and..) ↔ (.. and ..)

The logical items are not all used at the same time. In particular the ∀()⇒, ⇒∃(), and unify items are used in a special way. Initially, the interpreter evaluates each sequent trying to apply items in the following order:

1. Non splitting assertion items:
   T⇒, F⇒, ∼⇒, ∧⇒, ∃⇒, = ⇒
2. Non splitting goal items:
   ⇒T, ⇒F, ⇒∼, ⇒∨, ⇒→, ⇒∀
3. Non logical items
4. The atom and "and" items
5. Splitting goal items: ⇒∧, ⇒↔
6. Splitting assertion items: ∨⇒, →⇒, ↔⇒
7. ⇒∃
8. ∀⇒

After the above items have been applied as many times as possible, the interpreter then tries to apply the unify item to the resulting conjunction of sequents.

If the application of the unification item results in T then the processes terminates because the theorem has been proven. But, if the application of the unification item does not result in T, then the interpreter applies the ∀()⇒ and ⇒∃() items to certain formulas, and then repeats the whole process starting at step 1.

One major difference between this theorem and our previous sequent logic theorem provers [Brown 1, 3, 4, 6] is that the ⇒∃ and ∀⇒ have been inserted into the initial evaluation procedure as steps 7 and 8, and thus one instance of every quantifier is initially created before the unify rule is every applied. The reason for this is due to the vast numbers of trivial quantifiers produced by the modal sequent logic described in Section 2.2. It was found that unless instances of these quantifiers were produced before unification takes place, many important bindings would not be found quickly enough and irrelevant bindings would be produced by the forcing restriction, due to the fact that the relevant formulae would not be available for matching.

2.2 Theory of Modality

We first describe a very strong modal logic which captures the notion of logical truth of the meaning of object language sentences. That is, for example, letting ⊢ be the modal symbol for logical truth, and M be our recursive meaning function [1] then ⊢(M S) states that the meaning of the object language sentence S is logically true. Thus, amazing as it may seem, we can construct a definition of logical truth without the use of any set-theoretic concepts.

After explaining in Section 2.2.1 the logical notation that we use, we give in Section 2.2 the logical axioms of our modal logic. In Section 2.3 we discuss what we call the Possibility Problem of modal logic, and explain why any theory formulated in modal logic should also include certain specific non-logical axioms about possibility.

2.2.1 Notation

We now explain our notation.

The symbols of modal logic are:

$\vdash p$	p is logically true
$\vdash p\ q$	p entails q
$\diamond p$	p is possible
(World p)	p is a world

The last three modal symbols are defined in terms of the first one as follows:

$\vdash p\ q$	$=$ df $\vdash(p \rightarrow q)$
$\diamond p$	$=$ df $\sim \vdash \sim p$
(World p)	$=$ df $(\diamond p) \wedge \forall q(\vdash p\ q) \wedge (\vdash p(\sim q))$

2.2.2 The Logical Axioms of Modal Logic

Our theory of modality is based on a very strong modal logic which is described in [4, 5]. It consists of a single primitive unary symbol: $\vdash$ which is interpreted as logical truth. This modal logic is stronger than S5 and can be described by the following minimal set of inference rules and axioms:

R0: from p infer $\vdash p$
A1: $\vdash p \rightarrow p$
A2: $\vdash(p \rightarrow q) \rightarrow (\vdash p \rightarrow \vdash q)$
A3: $\vdash p \vee \vdash \sim \vdash p$
A4: $(\forall q\ \text{World}^* q \rightarrow \vdash qp) \rightarrow \vdash p$

The inference rule R0 and the axioms A1, A2 and A3 are essentially the S5 modal logic. The last axiom A4 expresses Leibniz's intuition that something is logically true only if it is true in all worlds. The World* and $\vdash$ symbols are essentially the same as World and $\vdash$ which are defined later in Section 2.2.2.

Our automatic theorem prover does not use the above axioms but is based on the sequent calculus derived from these axioms which is described in [5]. We describe this modal sequent calculus in Section 2.2.1 and then list some definitions of modal concepts used by the theorem prover in Section 2.2.2. Finally, in Section 2.2.3 we discuss the possibility problem of modal logic.

This modal logic consists of the symbols and laws of classical quantificational logic plus the unary necessity symbol $\vdash$ and the following axioms and inference rules:

R0, A1, A2, and A3 are essentially the inference rule and axioms of S5 modal logic. Axiom A4 which we call Leibniz's postulate expresses his intuition that something is logically true if it is true in all possible worlds. A good introduction to modal logic is given in Hughes and Cresswell [10].

2.2.3 The Possibility Problem

The possibility problem of modal logic is that from the logical axioms of modal logic we cannot prove certain elementary facts about the possibility of conjunctions of distinct possible negated atomic expressions consisting of non-logical symbols. For example, if we have a theory formulated in our modal logic which contains the non-logical atomic expression (ON A B) then since $\sim$(ON A B) is not logically true, it follows that (ON A B) must be possible. Yet $\diamond$(ON A B) is not a theorem of our modal logic.

Thus, for any theory expressed in modal logic, a certain number of non-logical axioms dealing with possibility should also be added. For example, in the case of the propositional logic, or in the case of the quantificational logic over a finite domain since it reduces to propositional logic, one sufficient but inefficient axiomatization would be to assert the possibility of all consistent disjunctions of conjunctions of literals as additional non-logical axioms:

$\diamond(\vee(\wedge \text{Literals}))$

A more computationally efficient axiomatization which is obtained by noting that the possibility of a disjunction of sentences is implied by the possibility of any one of those sentences:

$\diamond p \rightarrow \diamond(p \wedge q)$

is to assert only the possibility of all consistent conjunctions of literals:

$\diamond(\wedge \text{literals})$

Using our meaning function [1] this may be done in a finite manner by adding the single axiom:

(Conj S) ∧ (Consist S) ⇒ ⋄(M S)

where Conj and Consist are recursive functions defined as follows:

(Conj S)	= df (Lit S) ∧ ∃T ∃R (S=[T'∧R] ∧ (Lit T) ∧ (Conj R))
(Lit S)	= df (∃T S=['∼ T] ∧ (Atomicsent T)) ∨ (Atomicsent S)
(Consist [ ])	= df F
(Consist [S.L])	= df (Consist2 S L) ∧ (Consist L)
(Consist2 S [ ])	= df F
(Consist2 S[T.L]	= df ∼(Opp S T) ∧ (Consist2 S L)
(Opp S T)	= df (∃R S=['∼R] ∧ T=R) ∧ (∃R T=['∼R] ∧ S=R)

The methods for representing object language expressions in our logic and for obtaining their meanings are defined in [1, 7] [1977d]. For example [T'∧R] is essentially a structural descriptive name of (M T) ∧ (M R).

2.3 A Sequent Calculus for Modal Logic

We give in Section 2.3.1 some theorems of our modal logic which when used as rewrite rules will form the basis of a method for translating every expression of our modal logic into the standard form described in Section 2.3.2. In Section 2.3.3 we show that these rewrite rules form the basis of a complete proof procedure for modal logic. In Section 2.3.4 we list some definitions which are also to be used as rewrite rules. These rules are intended to be added to a classical sequent calculus such as: [Brown 1, 3, 4, 6].

2.3.1 Rules of the Sequent Calculus

We list below thirteen theorems of our modal logic of the form p↔q or r⇒(p↔q) which may be used to rewrite rules replacing p by q in any context in which r is a hypothesis. The symbols World* and ⊢ have the same meaning respectively as world and ⊢, but are never to be replaced by their definitions in a proof procedure using these rules. Furthermore any initial theorem given to such a proof procedure must not itself contain the world* or ⊢ symbols, although it could of course contain the World and ⊢ symbols.

⊢:	(⊢p)	↔	∀w(World*w)	⇒	⊢w p)
⊢ ∧:	(World*w)	→	((⊢w(p∧q))	↔	(⊢w p ∧ ⊢w q))
⊢ ∨:	(World*w)	→	((⊢w(p∨q))	↔	(⊢w p ∨ ⊢w q))
⊢ →:	(World*w)	→	((⊢w(p→q))	↔	(⊢w p → ⊢w q))
⊢ ↔:	(World*w)	→	((⊢w(p↔q))	↔	(⊢w p ↔ ⊢w q))
⊢ ∼:	(World*w)	→	((⊢w(∼p))	↔	(∼ ⊢w p)

$\vdash$ T:	(World*w)	$\rightarrow$	$((\vdash w\ T)$	$\leftrightarrow$	T
$\vdash$ F:	(World*w)	$\rightarrow$	$((\vdash w\ F)$	$\leftrightarrow$	F
$\vdash \forall$:	(World*w)	$\rightarrow$	$((\vdash w(\forall x\ \phi x))$	$\leftrightarrow$	$(\forall x \vdash w \phi x))$
$\vdash \exists$:	(World*w)	$\rightarrow$	$((\vdash w(\exists x\ \phi x))$	$\leftrightarrow$	$(\exists x \vdash w \phi x))$
$\vdash$ a:	(World*w)	$\rightarrow$	$((\vdash w(\forall p\ \phi p))$	$\leftrightarrow$	$(\forall p \vdash w \phi p))$
$\vdash$ e:	(World*w)	$\rightarrow$	$((\vdash w(\exists p\ \phi p))$	$\leftrightarrow$	$(\exists p \vdash w \phi p))$
$\vdash \vdash$:	(World*w)	$\rightarrow$	$((\vdash w(\vdash p))$	$\leftrightarrow$	$\vdash p$

The $\vdash\forall$ and $\vdash\exists$ theorems pertain to quantifiers of object language variables whereas the $\vdash$a, $\vdash$e theorems pertain to quantifiers for propositional variables. The $\vdash\forall$ and $\vdash\exists$ theorems are equivalent to the fact that something is an object iff it is logically true that it is an object. All these theorems hold regardless of whether propositions are objects or not.

In order to try to prove a theorem ψ with a proof procedure using these rules, sometimes it must actually try to prove $\vdash\psi$ instead. There is a deep and beautiful reason for this which is basically that this initial $\vdash$ inserted before ψ is a symbol of the metalanguage of this logic as are all the $\vdash$ and World* symbols. Essentially $\vdash\psi$ is the statement in the metatheory that ψ is logically true, and it is this rather than ψ itself which we are trying to prove.

Unlike sequent calculi for weaker modal logics [11] our sequent calculus leads to a very efficient proof procedure, as can be seen from the fact that the $\vdash$ law is an explicit definition of $\vdash$ in terms of $\vdash$w, and that the remaining twelve laws are essentially a contextual definition which eliminates all occurrences of the $\vdash$w symbol which do not occur immediately before an atomic sentence. Thus in a proof procedure based on this sequent calculus it makes no essential difference as to which of the thirteen laws is first applied. Furthermore it makes no essential difference to which subformula of the theorem being proven a law is first applied. All possible strategies of applying these laws, so long as they are applied as many times as possible will led to the standard form described in Section 3.2.

2.3.2 The Standard Form

If the rewrite rules given in Section 3.1 are supplemented by enough laws of classical logic which when used as rewrite rules suffice to put the sentences of classical logic into skolemized prenex conjunctive normal form, then every sentence of our modal logic will be rewritten to an equivalent sentence in the following standard form:

1. First a sequence of universal quantifiers (i.e. skolem functions) consisting of:
   (a) object variable quantifiers: $\forall$x
   (b) proposition variable quantifiers: $\forall$p (except those introduced by the $\vdash$ and $\vdash\vdash$ rules)
   (c) propositional variable quantifiers introduced by the $\vdash$ rule. These quantifiers are essentially treated as world quantifiers as the (World*w) hypothesis is always kept next to the quantifiers: ($\exists$w(World*w) $\Rightarrow$...).

2. Second a sequence of existential quantifiers (i.e. unification variables) consisting of:
   (a) object variable quantifiers: $\exists$x
   (b) proposition variable quantifiers: $\exists$p (except those introduced by the $\vdash$ and $\vdash\vdash$ rules)
   (c) propositional variable quantifiers introduced by the $\vdash$ rule. These quantifiers are essentially treated as World quantifiers as the (World*w) hypothesis is always kept next to the quantifier:
   ($\exists$w(World*w)...).

3. And finally a matrix consisting of:
   (a) a conjunction: $\wedge$
   (b) of a disjunction: $\vee$ (i.e. of sequents) (c) of negated: $\sim$ or unnegated

(d) atoms consisting of an entailment symbol $\vdash w$ whose first argument is a variable quantified by a quantifier of type (1c) or (2c), and whose second argument is either (i) a variable quantified by a quantifier of type (1b) or (2b), or (ii) a nonlogical atomic sentence containing no variables of type (1c) or (2c).

Schematically this standard form can be represented as:

$\forall X \forall p \forall w(\text{World}^* w) \Rightarrow \exists x \exists p \exists w(\text{World}^* w) \wedge$ Matrix

where the matrix is of the form:

$\wedge \vee \{\sim\} \vdash w \{ {}^{p}_{(\phi px)} \}$

where ϕ is a non-logical symbol.

The fact that sorted quantifiers can be pulled out of the matrix and skolemized can be justified by the following theorems of classical logic:

$\Rightarrow \forall w$: $(\forall w(\text{World}^* w) \Rightarrow (s \wedge (t \vee \phi w))) \leftrightarrow (s \wedge (t \vee (\forall w(\text{World}^* w) \Rightarrow \phi w)))$
$\Rightarrow \exists w$: $(\exists w(\text{World}^* w) \wedge (s \wedge (t \vee \phi w))) \leftrightarrow s \wedge (t \vee (\exists w W \text{World}^* w) \wedge \phi w)))$
Skolem: $(\exists x \Pi x \wedge (\forall a\ a \Rightarrow \phi xa)) \leftrightarrow (\forall a(\forall x \Pi x \Rightarrow (ax)) \Rightarrow (\exists x \Pi x \wedge \phi x(ax)))$

The $\Rightarrow \forall w$ and $\Rightarrow \exists w$ theorems depend on the truth of the theorem: $\exists w(\text{World } w)$. The "a" in "(ax)" of the Skolem theorem represents a skolem function.

2.3.3 Completeness and Expressibility

We consider the fragment of our modal logic which does not contain any propositional variables and where propositions are not objects. Given the standard form described in Section 3.2 we can prove the completeness of this fragment relative to classical state logic. Classical state logic is a classical quantificational logic containing two distinct sorts, namely objects and worlds such tat each non-logical symbol contains exactly one argument position which contains a world variable. This completeness proof is carried out merely by forming an isomorphism between expressions of our modal logic and expressions of state logic by translating each atom $\vdash w(\phi^n X_1 ... X_n)$ of our modal logic containing an n-ary non-logical symbol ϕ into an (n+1)-ary atom of state logic: $(\phi^{n+1} X_1 ... X_n w)$.

We now state the Completeness Theorem:

Completeness Theorem

For every sentence (a) of this fragment of our modal logic, there is an equivalent sentence (b) of our modal logic such that there exists a sentence (s) of classical state logic which is isomorphic to (b). There fore (s) is provable iff (b) and (a) are provable. Using this isomorphism we may also obtain an Expressibility Theorem for our modal logic:

Expressibility Theorem

For every sentence (t) of state logic there is an equivalent sentence (s) of state logic in skolemized prenex conjunctive normal form such that there exists a sentence (b) of our modal logic which is isomorphic to (s).

The expressibility theorem shows that everything expressible in state logic is expressible in our modal logic.

2.3.4 Rules Defining Modal Concepts

The laws are all of the form $p \leftrightarrow q$ and are to be used only to replace an expression of the form $p\theta$ by an expression of the form $q\theta$.

D$\vdash$:	$\vdash$p q	$\leftrightarrow \vdash(p \rightarrow q)$	"p entails q"
D$\equiv$:	$p \equiv q$	$\leftrightarrow \vdash(p \leftrightarrow q)$	"p is synonymous to q"
D$\diamond$:	$\diamond p$	$\leftrightarrow \sim\vdash\sim p$	"p is possible"
Ddet :	(Det p q)	$\leftrightarrow \vdash$p q$\vee\vdash$p$\sim$q	"p determines q"
Dcom :	(Complete p)	$\leftrightarrow \forall$q(Det p q)	"p is complete"
Dwor :	(World p)	$\leftrightarrow \diamond p \wedge$(Complete p)	"p is a world"

3. RELATIONAL SEMANTICS

The basic idea of Relational Semantics [8] is to define the proposition that p is necessary holds in a world w as the proposition that p holds in all worlds u related to w by some binary relation R:

DR: $\vdash w \sqcap p \leftrightarrow d\ f\ (\forall u.Rwu \rightarrow \vdash up)$

From this definition it is clear that the necessity symbol for various modal logics will be easily definable by assuming various axioms for the R-relation. The R-relation will be assumed to be an intensional symbol, and thus the following axiom shall hold.

RA1: $(p \leftrightarrow q) \rightarrow (\forall rRpr \leftrightarrow Rqr) \wedge (\forall rRrp \leftrightarrow Rrq)$

In particular this axiom is used to prove theorems RT1, RT4, RT5, and RT6 in Section 3.2. Also it does not seem unreasonable to interpret the R relation as being the same in all worlds.

RA2: $\diamond Rxy \rightarrow \vdash Rxy$

Static Relational Semantics is a degenerate case of Relational Semantics that may be obtained by assuming either the intensional logic sentence:

$\vdash \forall p\ (\sqcap p \rightarrow \vdash \sqcap p)$

or the semantic condition of theorem RT11:

$\forall u \forall v \forall w (Rvw \rightarrow Rvw)$

Essentially such an assumption makes the first argument of the relation R irrelevant to its meaning. For this reason; alternatively the same effect could be achieved by simply replacing the definition DR by the definition DSR:

DSR: $\vdash w \sqcap p \leftrightarrow (\forall uRu \rightarrow \vdash up)$

involving a unary predicate R whose argument is the second argument of the relation R. The utility of the DSR rule lies in the fact that if some intentional concept satisfies the axiom $\vdash \forall p \sqcap p \rightarrow \vdash \sqcap p$ then it will be more efficient to use the single law DSR as a rewrite rule rather than use both DR and that axiom.

After defining some basic concepts in Section 3.1, we list in Section 3.2 a number of theorems derivable from the axiom definitions DR and DSR. It will be seen that the theorems which relate laws of intensional logic to their semantic conditions on the R relation in DR or the R predicate in DSR are generalizations of many well known results about Relational Semantics. Proof of all these theorems are given in [9]. Some example proofs are given in Section 3.3. In Section 3.4 statistics of our automatic theorem prover's attempt to prove each of these theorems is given.

3.1 Definitions of Relational Semantics

We first make a few definitions:

D1:	Δp	$\leftrightarrow$	$\sim \sqcap \sim p$
D2:	($\sqcap$-det p q)	$\leftrightarrow$	$\sqcap(p \rightarrow q) \vee \sqcap(p \rightarrow \sim q)$
D3:	($\sqcap$-complete p)	$\leftrightarrow$	$\forall q$($\sqcap$-det p q)
D4:	($\sqcap$-world p)	$\leftrightarrow$	$\Delta p \wedge$($\sqcap$-complete p)
D5:	($\sqcap$-valid p)	$\leftrightarrow$	$\forall q$($\sqcap$-world q) $\rightarrow \sqcap(q \rightarrow p)$

3.2 Basic Theorems of Relational Semantics

The main theorems of Relational Semantics are listed below. Variables u, v, w, range over worlds.

RT0: $\forall p(\vdash p \rightarrow \vdash \sqcap p) \quad \leftrightarrow \quad T$

RT1:	$\vdash(\sqcap(p \rightarrow q) \rightarrow (\sqcap p \rightarrow \sqcap q)$	$\leftrightarrow$	T	
RT2:	$\vdash((\sqcap\text{-valid } p) \rightarrow \sqcap p)$	$\leftrightarrow$	T	
RT3:	$\vdash(\sqcap\forall x\phi x \leftrightarrow \forall x\, \sqcap\phi x)$	$\leftrightarrow$	T[1]	
RT4:	$\vdash(\exists x\sqcap\phi x\rightarrow \sqcap\exists x\phi x)$	$\leftrightarrow$	T	
RT5:	$\vdash \forall p(\sqcap p \rightarrow \Delta p)$	$\leftrightarrow$	$\forall w\exists uRwu$	"deontic" existence
RT6:	$\vdash \forall p(\Delta p) \rightarrow \sqcap p)$	$\leftrightarrow$	$\exists u\forall v\forall w(Rwu\land Rwv \rightarrow u\equiv v)$	uniqueness
RT7:	$\vdash \forall p(\sqcap p \rightarrow p)$	$\leftrightarrow$	$\forall wRww$	"T" reflexive
RT8:	$\vdash \forall p(p \rightarrow \sqcap\Delta p)$	$\leftrightarrow$	$\forall u\forall v(Ruv \rightarrow Rvu)$	"B" symmetric
RT9:	$\vdash \forall p(\sqcap p \rightarrow \sqcap\sqcap p)$	$\leftrightarrow$	$\forall u\ \forall v\ \forall w(Ruv \land Rvw \rightarrow Ruw)$	"S4" transitive
RT10:	$\vdash \forall p(\sqcap p \lor \sqcap \sim \sqcap p)$	$\leftrightarrow$	$\forall u\forall v\forall w(Ruv \land Ruw \rightarrow Rvw)$	"S5"
RT11:	$\vdash \forall p(\sqcap p \rightarrow \vdash\sqcap p)$	$\leftrightarrow$	$\forall u\forall v\forall w(Ruw \rightarrow Rvw)$	"Static"

SRT0:	$\forall p \vdash p \rightarrow \vdash \sqcap p$			
SRT1:	$\vdash(\sqcap(p \rightarrow q) \rightarrow (\sqcap p \rightarrow \sqcap q)$			
SRT2:	$\vdash((\sqcap\text{-valid } p) \rightarrow \sqcap p$			
SRT3:	$\vdash(\sqcap\forall x\phi x \leftrightarrow \forall x\sqcap\phi x)$			
SRT4:	$\vdash(\exists x\ \sqcap\phi x \rightarrow \sqcap\exists x\phi x)$			
SRT5:	$\vdash(\forall p(\sqcap p \rightarrow \Delta p)$	$\leftrightarrow$	$\exists uRu$	"Deontic" existence
SRT6:	$\vdash(\forall p(\Delta p \rightarrow \sqcap p)$	$\leftrightarrow$	$\forall u\forall v\ Ru \land Rv \rightarrow u\equiv v$	uniqueness
SRT7:	$\vdash(\forall p(\sqcap p \rightarrow p)$	$\leftrightarrow$	$\forall wRw$	"T" reflexive
SRT8:	$\vdash(\forall p(p \rightarrow \sqcap\ \Delta p)$	$\leftrightarrow$	$\exists vRv \rightarrow \forall uRu$	"B" symmetric
SRT9:	$\vdash(\forall p(\sqcap p \rightarrow \sqcap\sqcap p)$	$\leftrightarrow$	T	"S4" transitive
SRT10:	$\vdash(\forall p(\sqcap p \lor \sqcap\sim\sqcap p)$	$\leftrightarrow$	T	"S5"
SRT11:	$\vdash(\forall p(\sqcap p \rightarrow \vdash\sqcap p)$	$\leftrightarrow$	T	"Static"

It is interesting to note that the modal laws of the SRT9 and SRT10 theorems become true without any restrictions on the R verb. Also it follows that the modal laws of the SRT7 become equivalent to the conjunction of the modal laws of SRT5 and SRT8:

$\vdash\forall p(\sqcap p \rightarrow p \leftrightarrow (\vdash\forall p\sqcap p \rightarrow \Delta p \land \vdash\forall p(p \Rightarrow \sqcap\Delta p))$

$\forall wRw \leftrightarrow (\exists wRw \land (\exists wRw \rightarrow \forall wRw))$

$\forall wRw \leftrightarrow (\exists wRw \land \forall wRw)$

$\forall wRw \leftrightarrow \forall wRw$

reflexitivity $\leftrightarrow$ existence $\land$ symmetry

[1]The reader should not be disturbed by the fact that **RT3** holds in our Relational Semantics where as in Kripke's [8] work it did not; because this is merely a consequence of Kripke's way of defining the classical logic operation of substitution. In [9] we show how to obtain quantifiers such that **RT3** does not hold.

3.3 Examples

```
RT5: ⊢∀p(⊓p→△p) ↔ ∀w∃uRwu                                                    Rules
   Θ0:
  RT5A ⇒ ∀w∃uRwu → ⊢∀p (⊓p→△p)                                               :⇒→,⊢
    ∀w∃uRwu ⇒ ∀ x1 ⊢ x1 ∀p (⊓p→ △p)                                          :⇒∀,⊢∀
    ∀w∃uRwu ⇒ ∀p ⊢ c1 (⊓p→ △p)                                               :⇒∀,⊢→
    ∀w∃uRwu ⇒⊢c1 (⊓ c2) → ⊢ c1 (△c2)                                         :⇒→,DR
    ∀ x2 (R c1 x2 → ⊢ x2 c2), ∀w∃uRwu ⇒ ⊢ c1 (△ c2)                          :D1,⊢∼
    ∀ x2 (R c1 x2 → ⊢ x2 c2), ∀w∃uRwu ⇒ ∼ ⊢ c1 (⊓ ∼ c2)                      :⇒∼,DR
    ∀ x3 (R c1 x3 → ⊢ x3 (∼ c2)), ∀ x2 (R c1 x2 → ⊢ x2 c2), ∀w∃uRwu ⇒        :∀⇒, →⇒

      ∀(x3 *1)(R c1 x3 → ⊢ x3 (∼ c2)), ⊢ *1(∼ c2),
         ∀ x2(R c1 x2 → ⊢ x2 c2), ∀w∃uRwu ⇒                                  :⊢∼

      (1), ∼⊢ *1 c2, ∀ x2 (R c1 x2 → ⊢ x2 c2), ∀w∃uRwu                       :∼⇒,∀⇒,→⇒

         (1), ∀(x2 *2)(R c1 x2 → ⊢ *2 c2), ⊢ *2 c2, ∀w∃uRwu ⇒ ⊢ *1 c2        :∀⇒, ∃⇒

         (1),(2), ⊢ *2 c2, ∀(w *3) ∃uRwu, R *3 (c3 *3) ⇒ ⊢ *1 c2             :unify
                T (*1 = *2)
         (1),(2),∀w∃uRwu ⇒ ⊢ *1 c2, R c1 *2                                  :∀⇒,∃⇒
         (1),(2),∀(w *4)∃uRwu, R *4 (c4 *4) ⇒ ⊢ *1 c2, R c1 *2               :unify
                T (*1 = (c4 *4), *4 = c1)
      (1),∀ x2 (R c1 x2 → ⊢ x2 c2), ∀w∃uRwu ⇒ R c1 *1                        :∀⇒,→⇒

         (1),∀(x2 *5)(R c1 x2 → ⊢ x2 c2), ⊢ *5 c2, ∀w∃uRwu ⇒ R c1 *1         :∀⇒, ∃⇒
         (1),(3),⊢ *5 c2, ∀(w *6) ∃uRwu, R *6 (c5 *6) ⇒ R c1 *1              :unify
                Θ1 (fail-substitute, reinstate, try again)
         (1),(3),∀w∃uRwu ⇒ R c1 *1, R c1 *5                                  :∀⇒,∃⇒
         (1),(3),∀(w *7)∃uRwu, R *7 (c6 *7) ⇒ R c1 *1, R c1 *5               :unify
                T (*5 = (c6 *7), *7 = c1)
  Θ1: ∀(x3(c4 *4))(R c1 x3 → ⊢ x3 (∼c2)), ∀(x2(c6 *7))(R c1 x2 → ⊢ x2 c2),
    ⊢(c6 *7) c2, ∀(w *6) ∃uRwu, R *6 (c5 *6) ⇒ R c1 (c4 *4)                  :reinstate
    ∀(x3(c4 *4)*9)(R c1 x3 → ⊢ x3(∼c2)), R c1 *9 → ⊢ *9 (∼ c2),
      ∀(x2(c6 *7)*8)(R c1 x2 → ⊢ x2 c2), R c1 *8 → ⊢ *8 c2, ⊢(c6 *7) c2,
      (4),R *6(c5 *6) ⇒ R c1(c4 *4)                                           :→⇒


   (5),R c1 *9 → ⊢ *9 (∼c2),(6),⊢ *8 c2, ⊢(c6 *7) c2,(4),R *6(c5 *6)
      ⇒ R c1(c4 *4)                                                           :→⇒

      (5),⊢ *9(∼c2),(6),⊢ *8 c2, ⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4)    :⊢∼
      (5),∼⊢ *9 c2,(6),⊢ *8 c2,⊢(c6 *7) c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4)     :∼⇒
      (5),(6),⊢ *8 c2,⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4), ⊢ *9 c2      :unify
                T (*8 = *9) or (*9 = (c6 *7))
      (5),(6),⊢ *8 c2,⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4), R c1 *9      :unify
                T (*6 = c1, *9 = (c5 *6) ≠ (c6 *7))
   (5),R c1 *9 → ⊢ *9(∼c2),(6),⊢(c6 *7)c2,(4),R *6(c5 *6)
      ⇒ R c1(c4 *4), R c1 *8                                                  :→⇒

      (5),⊢ *9(∼c2),(6),⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4), R c1 *8    :⊢∼
      (5),∼ ⊢ *9 c2,(6),⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4), R c1 *8    :∼⇒
      (5),(6),⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4), R c1 *8, ⊢ *9 c2     :unify
                T
      (5),(6),⊢(c6 *7)c2,(4),R *6(c5 *6) ⇒ R c1(c4 *4), R c1 *8, R c1 *9     :unify
                T
```

```
θ0:
RT5B ⇒ ⊢ ∀p(⊓p→△p)→∀w∃uRwu                                                    :⇒→,⊢
  ∀ x1 ⊢ x1(∀p(⊓p→△p)) ⇒ ∀w∃uRwu                                              :⇒∀,⇒∃,∀⇒,⊢∀
  ∀(x1 *2)⊢ x1(∀p(⊓p→△p)),∀p ⊢ *2(⊓p→△p) ⇒ ∃(u *1) R c1 u, R c1 *1           :∀⇒,⊢→

   (2),∀(p *3)⊢ *2(⊓p→△p), ⊢ *2(⊓ *3)→ ⊢ *2(△ *3) ⇒ (1),R c1 *1              :→⇒


      (2),(3),⊢ *2(△ *3) ⇒(1),R c1 *1                                         :D1,⊢∼
      (2),(3),∼ ⊢ *2(⊓∼*3) ⇒(1),R c1 *1                                       :∼⇒,DR
      (2),(3)⇒(1), R c1 *1, ∀ x2 (R *2 x2 →⊢ x2(∼ *3))                         :⇒∀,⇒→,⊢∼
      (2),(3),R *2(c2 *3 *2) ⇒ (1),R c1 *1, ∼⊢(c2 *3 *2) *3                   :⇒∼
      ⊢(c2 *3 *2) *3,(2),(3),R *2(c2 *3 *2) ⇒ (1),R c1 *1                     :unify
                     T (*1 = (c2 *3 *2), *2 = c1)
      (2),(3)⇒(1),R c1 *1, ⊢ *2(⊓ *3)                                          :DR
      (2),(3)⇒(1),R c1 *1, ∀ x3(R *2 x3 →⊢ x3 *3)                              :⇒∀,⇒,→
      R *2(c3 *3 *2),(2),(3)⇒(1), R c1 *1, ⊢(c3 *3 *2) *3                     :unify
                     θ1 (fail, try again)

θ1: R c1(c3 *3),∀(x1 c1)⊢ x1(∀p(⊓p→△p)), ∀(p *3)⊢ c1(⊓p→△p)
   ⇒ ∃(u(c2 *3 *2))R c1 u, R c1(c2 *3 *2), ⊢(c3 *3) *3                       :reinstate
    R c1(c3 *3), ∀(x1 c1 *4) ⊢ x1(∀p(⊓p→△p)), ⊢ *4(∀p(⊓p→△p)),(3)
      ⇒ ∃(u(c2 *3 *2) *5) R c1 u, R c1 *5, R c1(c2 *3 *2), ⊢(c3 *3) *3        :∀⇒,⊢→
    R c1(c3 *3),(4),∀(p *6)⊢ *4(⊓p→△p),⊢ *4(⊓ *6)→⊢ *4(△ *6),(3)
      ⇒(5), R c1 *5, R c1(c2 *3 *2), ⊢(c3 *3) *3                              :→⇒

      R c1(c3 *3),(4),(6),⊢ *4(△ *6),(3)
        ⇒(5), R c1 *5, R c1(c2 *3 *2), ⊢(c3 *3) *3                            :D1,⊢∼
      R c1(c3 *3),(4),(6),∼⊢ *4(⊓∼ *6),(3)
        ⇒(5), R c1 *5, R c1(c2 *3 *2), ⊢(c3 *3) *3                            :∼⇒,DR
      R c1(c3 *3),(4),(6),(3)⇒(5),R c1 *5, R c1(c2 *3 *2),⊢(c3 *3) *3,
        ∀ x4(R *4 x4 → ⊢ x4(∼ *6))                                             :⇒∀,⇒→,⊢∼
      R *4(c4 *6 *4), R c1(c3 *3),(4),(6),(3)
        ⇒(5),R c1 *5, R c1(c2 *3 *2),⊢(c3 *3) *3, ∼⊢(c4 *6 *4) *6            :⊢∼
      ⊢(c4 *6 *4)*6, R *4(c4 *6 *4), R c1(c3 *3),(4),(6),(3)
         ⇒ R c1 *5, R c1(c2 *3 *2), ⊢(c3 *3) *3                               :unify
                     T (*5 = (c3 *3))
      R c1(c3 *3),(4),(6),(3)⇒(5),R c1 *5, R c1(c2 *3 *2),
        ⊢(c3 *3)*3, ⊢ *4(⊓ *6)                                                 :DR
      R c1(c3 *3),(4),(6),(3)⇒(5),R c1 *5, R c1(c2 *3 *2),
        ⊢(c3 *3)*3, ∀ x5(R *4 x5 →⊢ x5 *6)                                     :⇒∀,⇒→
      R *4(c5 *6 *4),R c1(c3 *3),(4),(6),(3)
        ⇒(5),R c1 *5, R c1(c2 *3 *2), ⊢(c3 *3) *3, ⊢(c5 *6 *4) *6            :unify
                     T
```

Observe that in the "A" part of this proof, much effort could be saved by reducing ∀w∃uRwu before the proof begins to split.

```
RT7: ⊢ ∀p(⊓p→p) ↔ ∀wRww
   Θ_0:
  RT7A ⇒ ∀wRww → ⊢ ∀p(⊓p→p)                                        :⇒→,⊢
     ∀wRww ⇒ ∀ x1 ⊢ x1(∀p(⊓p→p))                                   :⇒∀,⊢∀
     ∀wRww ⇒ ∀p(⊢ c1(⊓p→p))                                        :⇒∀,⊢→
     ∀wRww ⇒ ⊢ c1(⊓ c2) → ⊢ c1 c2                                  :⇒→,DR
     ∀ x2(r c1 x2 → ⊢ x2 c2), ∀wRww ⇒ ⊢ c1 c2                      :∀⇒,→,⇒

        ∀(x2 *1)(R c1 x2 → ⊢ x2 c2), ⊢ *1 c2, ∀wRww ⇒ ⊢ c1 c2      :∀⇒

        (1),⊢ *1 c2, ∀(w *2) Rww, R *2 *2 ⇒ ⊢ c1 c2                :unify
                   T (*1 = c1)

        ∀wRww,(1)⇒ ⊢ c1 c2, R c1 *1                                :∀⇒
        ∀(w *3) Rww, R *3 *3, (1) ⇒ ⊢ c1 c2, R c1 *1               :unify
                   T (*3 = c1)

   Θ_0:
  RT7B ⇒ ⊢(∀p(⊓p → p)) → ∀wRww
     ∀ x1 ⊢ x1(∀p(⊓p → p)) → ∀wRww                                 :⇒→,⊢
     ∀(x1 *1) ⊢ x1(∀p(⊓p → p)), ∀p ⊢ *1(⊓p → p) ⇒ R c1 c1          :⇒∀,∀⇒,⊢∀

      (1),∀(p *2) ⊢ *1(⊓p → p), ⊢ *1(⊓ *2)→ ⊢ *1 *2 ⇒ R c1 c1      :∀⇒,⊢→

        (1),(2),⊢ *1 *2 ⇒ R c1 c1                                  :unify
                   Θ_1 (fail-cannot reinstate)
        (1),(2) ⇒ R c1 c1, ⊢ *1(⊓ *2)                              :DR
        (1),(2) ⇒ R c1 c1, ∀ x2(R *1 x2 → ⊢ x2 *2)                 :⇒∀,⇒→
        R *1(c2 *2 *1), (1),(2) ⇒ R c1 c1, ⊢(c2 *2 *1) *2          :unify
                   Θ_2 (fail-cannot reinstate)
```

RT9: ⊢ ∀p(⊓p → ⊓⊓p) ↔ ∀u∀v∀w((Ruv ∧ Rvw) → Ruw) **Rules**

Θ_0:

RT9a ⇒ ∀u∀v∀w((Ruv ∧ Rvw) → Ruw) → ⊢ ∀p(⊓p → ⊓⊓p) :⇒→,⊢
∀u∀v∀w((Ruv ∧ Rvw) → Ruw) ⇒ ∀ x1 ⊢ x1(∀p(⊓p → ⊓⊓p)) :⇒∀,⊢∀
∀u∀v∀w((Ruv ∧ Rvw) → Ruw) ⇒ ∀ p ⊢ c1(⊓p → ⊓⊓p)) :⇒∀,⊢→
∀u∀v∀w((Ruv ∧ Rvw) → Ruw) ⇒ ⊢ c1(⊓ c2) → ⊢ c1(⊓⊓p) :⇒→,DR
∀ x2(R c1 x2 → ⊢ x2 c2), ∀u∀v∀w((Ruv ∧ Rvw) → Ruw) ⇒ ⊢ c1(⊓⊓ c2) :DR
∀ x2(R c1 x2 → ⊢ x2 c2), ∀u∀v∀w((Ruv ∧ Rvw) → Ruw)
⇒ ∀ x3(R c1 x3 → ⊢ x3(⊓ c2)) :⇒∀,⇒→,DR
R c1 c3, ∀ x2(R c1 x2 → ⊢ x2 c2), ∀u∀v∀w((Ruv ∧ Rvw) → Ruw)
⇒ ∀ x4(R c3 x4 → ⊢ x4 c2) :⇒∀,⇒→,∀⇒,→,⇒

R c3 c4,R c1 c3,∀(x2 *1)(R c1 x2 → ⊢ x2 c2),⊢ *1 c2,∀u∀v∀w((Ruv ∧ Rvw) → Ruw)
⇒ ⊢ c4 c2 :∀⇒,∀⇒,∀⇒,→⇒

R c3 c4,R c1 c3,(1),⊢ *1 c2,∀(u *2)∀v∀w((Ruv ∧ Rvw) → Ruw),
∀(v *3)∀w((R *2 v ∧ Rvw)→ R *2 w),
∀(w *4)((R *2 *3 ∧ R *3 w) → R *2 w),R *2 *4 ⇒ ⊢ c4 c2 :unify
T (*1 = c4)
R c3 c4,R c1 c3,(1),⊢ *1 c2,(2),(3),(4) ⇒ ⊢ c4 c2,R *2 *3 ∧ R *3 *4 :⇒∧

R c3 c4,R c1 c3,(1),⊢ *1 c2,(2),(3),(4) ⇒ ⊢ c4 c2,R *2 *3 :unify
T
R c3 c4,R c1 c3,(1),⊢ *1 c2,(2),(3),(4) ⇒ ⊢ c4 c2,R *3 *4 :unify
T
R c3 c4,R c1 c3,(1),∀u∀v∀w((Ruv ∧ Rvw) → Ruw) ⇒ ⊢ c4 c2,R c1 *1 :∀⇒,∀⇒,∀⇒,→⇒

R c3 c4,R c1 c3,(1),∀(u *5)((Ruv ∧ Rvw) → Ruw),∀(v *6)∀w((R *5 v ∧ Rvw) → R *5 w),
∀(w *7)((R *5 *6 ∧ R *6 w) → R *5 w),R *5 *7 ⇒ ⊢ c4 c2,R c1 *1 :unify
T (*5 = c1, *7 = c4)
R c3 c4,R c1 c3,(1),(5),(6),(7) ⇒ ⊢ c4 c2,R c1 *1,R *5 *6 ∧ R *6 *7 :⇒∧

R c3 c4,R c1 c3,(1),(5),(6),(7) ⇒ ⊢ c4 c2,R c1 *1,R *5 *6 :unify
T (*6 = c3)
R c3 c4,R c1 c3,(1),(5),(6),(7) ⇒ ⊢ c4 c2,R c1 *1,R *6 *7 :unify
T

Θ_0:

RT9b ⇒ ⊢ ∀p(⊓p → ⊓⊓p) → ∀u∀v∀w((Ruv ∧ Rvw) → Ruw) :⇒→,⊢
∀ x1 ⊢ x1(∀p(⊓p → ⊓⊓p)) ⇒ ∀u∀v∀w((Ruv ∧ Rvw) → Ruw) :⇒∀,⇒∀,⇒∀,⇒→,∧⇒,∀⇒,⊢
R c2 c3,R c1 c2,∀(x1 *1) ⊢ x1(∀p(⊓p → ⊓⊓p)),∀p ⊢ *1(⊓p → ⊓⊓p) ⇒ R c1 c3 :∀⇒,⊢–
R c2 c3,R c1 c2,(1),∀(p *2) ⊢ *1(⊓p → ⊓⊓p),⊢ *1(⊓ *2) → ⊢ *1(⊓⊓ *2) ⇒ R c1 c3 :→=

R c2 c3,R c1 c2,(1),(2),⊢ *1(⊓⊓ *2) ⇒ R c1 c3 :D
R c2 c3,R c1 c2,(1),(2),∀ x2(R *1 x2 → ⊢ x2(⊓ *2)) ⇒ R c1 c3 :∀⇒,→⇒

R c2 c3,R c1 c2,(1),(2),∀(x2 *3)(R *1 *2 → ⊢ x2(⊓ *2)),⊢ *2(⊓ *2) ⇒ R c1 c3 :D
R c2 c3,R c1 c2,(1),(2),(3),∀ x3(R *3 x3 → ⊢ x3 *2) ⇒ R c1 c3 :∀⇒,→=

R c2 c3,R c1 c2,(1),(2),(3),∀(x3 *4)(R *3 x3 → ⊢ x3 *2),⊢ *4 *2 ⇒ R c1 c3 :unif
Θ_1 (fail-try again)
R c2 c3,R c1 c2,(1),(2),(3),(4) ⇒ R c1 c3,R *3 *4 :unif
T (*3 = c1, *4 = c2) or (*3 = c2, *4 = c3)

R c2 c3,R c1 c2,(1),(2),(3) ⇒ R c1 c3,R *1 *3 :unif
T (*1 = c1, *3 ≠ c1)

R c2 c3,R c1 c2,(1),(2) ⇒ R c1 c3, ⊢ *1(⊓ *2) :D
R c2 c3,R c1 c2,(1),(2) ⇒ R c1 c3, ∀ x4(R *1 x4 → ⊢ x4 *2) :⇒∀,⇒–

R *1(c4 *2 *1),R c2 c3,R c1 c2,(1),(2) ⇒ R c1 c3,⊢(c4 *2 *1) *2 :unify
Θ_2 (fail-try again)

Θ_2:
R c1(c4 *2),R c2 c3,R c1 c2,∀(x1 c1) ⊢ x1(∀p(∏p → ∏∏p)),∀(p *2) ⊢ c1(∏p → ∏∏p)
⇒ R c1 c3,⊢(c4 *2) :reinstate
R c1(c4 *2),R c2 c3,R c1 c2,∀(x1 c1 *5) ⊢ x1(∀p(∏p → ∏∏p)),⊢ *5(∀p(∏p → ∏∏p)),(5)
⇒ R c1 c3,⊢(c4 *2) *2 :⊢∀
R c1(c4 *2),R c2 c3,R c1 c2,(6),∀p ⊢ *5(∏p → ∏∏p),(5)
⇒ R c1 c3,⊢(c4 *2) *2 :∀⇒,⊢→
R c1(c4 *2),R c2 c3,R c1 c2,(6),∀(p *6) ⊢ *5(∏p → ∏∏p),⊢ *5(∏∏ *6),(5)
⇒ R c1 c3,⊢(c4 *2) *2 :→⇒

R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),⊢ *5(∏∏ *6),(5) ⇒ R c1 c3,⊢(c4 *2) *2 :DI
R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),∀ x5(R *5 x5 → ⊢ x5(∏ *6)),(5)
⇒ R c1 c3,⊢(c4 *2) *2 :∀⇒,→⇒

R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),∀(x5 *7)(R *5 x5 → ⊢ x5(∏ *6)),⊢ *7(∏ *6),(5)
⇒ R c1 c3,⊢(c4 *2) *2 :DI
R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),(8),∀ x6(R *7 x6 → ⊢ x6 *6),(5)
⇒ R c1 c3, ⊢(c4 *2) *2 :∀⇒,→⇒

R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),(8),∀(x6 *8)(R *7 x6 → ⊢ x6 *6),⊢ *8 *6,(5)
⇒ R c1 c3,⊢(c4 *2) *2 :unif
T (*6 = *2, *8 = (c4 *2))
R c1(c4 *2),R c2 c3,R c1 c3,(6),(7),(8),(9),(5) ⇒ R c1 c3,⊢(c4 *2) *2,R *7 *8 :unif
T (*7 = c1)

R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),(8),(5) ⇒ R c1 c3,⊢(c4 *2),R *5 *7 :unif
Θ_3 (fail-do not try again)

R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),(5) ⇒ R c1 c3,⊢(c4 *2) *2,⊢ *5(∏ *6) :DI
R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),(5) ⇒ R c1 c3,⊢(c4 *2) *2,
∀ x7(R *5 x7 → ⊢ x7 *6) :⇒∀,⇒→
R *5(c5 *6 *5),R c1(c4 *2),R c2 c3,R c1 c2,(6),(7),(5)
⇒ R c1 c3,⊢(c4 *2) *2,⊢(c5 *6 *5) *6 :unif
Θ_4 (fail-do not try again)

3.3 Results

All of the theorems of Relational Semantics and static Relational Semantics were tested on the theorem prover. The results of this testing are recorded in the tables which follow. Although a number of the proofs failed, the results are instructive, as noted in the examples. Even the failures terminate in a reasonable amount of time.

THEOREM	OUTCOME	TIME*	GC	CONSES	SEQUENTS	THETAKT
RT0	T	138	0	407	12	0
RT1	T	575	0	1556	31	0
RT2	F	2709	1	4057	54	1
RT3	T	524	0	1485	32	0
RT4	T	342	0	836	18	0
RT5A	T	2258	1	3534	50	1
RT5B	T	878	0	2322	43	1
RT6A	T	5500	2	6783	88	1
RT6B	F	447	0	1283	22	1
RT7A	T	251	0	761	17	0
RT7B	F	266	0	845	15	1
RT8A	T	484	0	1375	31	0
RT8B	F	2035	1	3098	44	1
RT9A	T	624	0	1865	37	0
RT9B	F	4715	2	7360	47	1
RT10A	T	714	0	1906	39	0
RT10B	F	8721	4	10834	79	1
RT11A	T	556	0	1529	32	0
RT11B	F	1745	1	2413	26	1
Avg. of	twelve	successes:		2030	36	0

*In milliseconds

THEOREM	OUTCOME	TIME	GC	CONSES	SEQUENTS	THETAKT
SRT0	T	147	0	402	12	0
SRT1	T	491	0	1537	31	0
SRT2	F	2545	1	4070	54	1
SRT3	T	498	0	1465	32	0
SRT4	T	323	0	870	18	0
SRT5A	T	405	0	1168	27	0
SRT5B	T	484	0	1288	24	1
SRT6A	T	1027	0	2841	44	1
SRT6B	F	2163	1	3368	33	1
SRT7A	T	258	0	749	17	0
SRT7B	F	282	0	820	15	1
SRT8A	T	494	0	1368	32	0
SRT8B	F	572	0	1733	26	1
SRT9	T	298	0	849	20	0
SRT10	T	314	0	879	22	0
SRT11	T	277	0	849	19	0
Avg. of	twelve	successes:		1189	25	0

4. FUNCTIONAL SEMANTICS

The basic idea of Functional Semantics is to define the proposition that p is necessary holds in a world w as the proposition that p is entailed by the strongest proposition which is necessary in that world w. We let (f w) be the propositional function value which represents the strongest proposition which is necessary in that world. Thus we say:

DF: $\vdash \sqcap w\ p \leftrightarrow \vdash (f\ w) p$

From this definition it will be possible to define the necessity symbol for various modal logics by assuming various axioms for the (f w) function value.

Static Functional Semantics is a degenerate case of Functional Semantics that may be obtained by assuming either the intensional logic axiom:

$\forall p \sqcap p \rightarrow \vdash \sqcap p$

or by assuming the semantic condition

$\forall u \forall w \vdash fu\ fv$

Essentially such an assumption makes the propositional function f a propositional constant since it must then have the same meaning for all worlds. For this reason, alternatively the same effect could be achieved by simply replacing the definition DF by the definition DSF:

DSF: $\vdash w \sqcap p \leftrightarrow \vdash fp$

involving a propositional constant f. It should be noted that if a particular concept satisfies the axiom $\forall u \forall v \vdash fu\ fv$ then it will be easier to prove theorems about it using DSF instead of using both DR and that axiom.

After defining some basic concepts in Section 4.1, we list in Section 4.2 some theorems which can be derived from the axiom definition DF and some theorems which can be obtained from the axiom definition DSF. It should be noted that in each case the theorems are similar to those which can be obtained from the Relational Semantics described in Section 3. The advantage of our new Functional Semantics over Relational Semantics is however that it is usually much easier to obtain proof in this new semantic system. Evidence supporting this assertion is given in Section 4.3 where we present a few example proofs obtained by our automatic theorem prover. Proofs of all these theorems are given in [9]. Statistics of the automatic theorem provers attempt to prove each of these theorems is given in Section 4.3.

4.1 Definitions of Functional Semantics

We first make a few definitions:

D1:	Δp	$\leftrightarrow \sim\sqcap\sim p$
D2:	($\sqcap$-det p q)	$\leftrightarrow \sqcap(p\rightarrow q) \vee \sqcap(p\rightarrow\sim q)$
D3:	($\sqcap$-complete p)	$\leftrightarrow \forall q(\sqcap\text{-det } p\ q)$
D4:	($\sqcap$-world p)	$\leftrightarrow \Delta p \wedge(\sqcap\text{-complete } p)$
D5:	($\sqcap$-valid p)	$\leftrightarrow \forall q(\sqcap\text{-world } q) \rightarrow \sqcap(q\rightarrow p)$

4.2 Basic Theorems of Functional Semantics

We assume the definitions D1-D5 given in Section 2.1. The main theorems of Functional-semantics are listed below.

FT0:	$\forall p \vdash p \rightarrow \vdash \sqcap p$	$\leftrightarrow$ T	"T"	
FT1:	$\vdash \sqcap(p \rightarrow q) \rightarrow (\sqcap p \rightarrow \sqcap q)$	$\leftrightarrow$ T	"T"	
FT2:	$\vdash ((\sqcap\text{-valid } p) \rightarrow \sqcap p$	$\leftrightarrow$ T	"atomic"	
FT3:	$\vdash (\sqcap \forall x\ \phi\ x \rightarrow \forall x\ \sqcap \phi x)$	$\leftrightarrow$ T	"complete"	
FT4:	$\vdash (\exists x\ \sqcap\ \phi\ x \rightarrow \sqcap\ \exists x \phi x)$	$\leftrightarrow$ T		
FT5:	$\vdash \forall p(\sqcap p \rightarrow \Delta p) \leftrightarrow (\forall w \diamond fw)$	"Detonic existance"		
FT5:*	$\vdash \forall p(\sqcap p \rightarrow \Delta p) \leftrightarrow \forall w \exists u \vdash ufw)$	"Detonic existance"		
FT6:	$\vdash \forall p(\Delta p \rightarrow \sqcap p) \leftrightarrow (\forall w \text{ complete } (fw))$		"uniqueness"	
FT6:*	$\vdash \forall p(\Delta p \rightarrow \sqcap p) \leftrightarrow \forall w \forall r \forall v \vdash ufw \wedge \vdash vfw \rightarrow u \equiv v$			
FT7:	$\vdash \forall p(\sqcap p \rightarrow p) \leftrightarrow \forall w \vdash wfw)$	"T" "reflexive"		
FT8:	$\vdash \forall p(p \rightarrow \sqcap\ \Delta p) \leftrightarrow \forall a \forall b \vdash bfa \rightarrow \vdash afb$		"B"	"symmetric"
FT9:	$\vdash \forall p(\ \sqcap p \rightarrow \sqcap\ \sqcap p) \leftrightarrow \forall a \forall b \forall c \vdash cfb \wedge \vdash bfa \rightarrow \vdash cfa$		"S4"	"transitive"
FT9:*	$\vdash \forall p(\sqcap p \rightarrow \sqcap\sqcap p) \leftrightarrow \forall a \forall b \forall c \vdash cfb \wedge \vdash bfa \rightarrow \vdash cfa$		"S4"	"transitive"
FT10:	$\vdash \forall p(\ \sqcap p \vee \sqcap \sim \sqcap p \leftrightarrow \forall a \forall b \vdash bfa \rightarrow \vdash fafb$		"S5"	
FT10:*	$\vdash \forall p\ (\sqcap p \vee \sqcap \sim \sqcap p \leftrightarrow \forall a \forall b \forall c \wedge \vdash cfa \rightarrow \vdash cfb$		"S5"	
FT11:	$\vdash \forall(\sqcap p \rightarrow \vdash \sqcap p) \leftrightarrow \forall u \forall v \vdash fu\ fv$	"static"		
FT11:*	$\vdash \forall(\sqcap p \rightarrow \vdash \sqcap p) \leftrightarrow \forall u \forall v \forall w \vdash w\ fu \rightarrow \vdash w\ fv$		"static"	

It is interesting to note that if $\sqcap$ is interpreted as $\vdash$ and if fw is interpreted as true than all these theorems will be theorems of our original modal logic.

SFT0:	$\forall p \vdash p \rightarrow \vdash \sqcap p$			
SFT1:	$\vdash \sqcap(p \rightarrow q) \rightarrow (\sqcap p \rightarrow \sqcap q)$			
SFT2:	$\vdash ((\sqcap \text{valid } p) \rightarrow \sqcap p)$			
SFT3:	$\vdash (\sqcap \forall x \phi x \leftrightarrow \forall x\ \sqcap\ \phi x)$			
SFT4:	$\vdash (\exists x\ \sqcap\ \phi x \rightarrow \sqcap \exists x \phi x)$			
SFT5:	$\vdash \forall p(\sqcap p \rightarrow \Delta p)$	$\leftrightarrow \diamond f$	"deontic" existence	
SFT6:	$\vdash \forall p(\Delta p \rightarrow \sqcap p)$	$\leftrightarrow \exists utuf$	(complete$\sqcap$) uniqueness	
SFT7:	$\vdash \forall p(\sqcap p \rightarrow p)$	$\leftrightarrow f$	"T" reflexive	
SFT8:	$\vdash \forall p(p \rightarrow \sqcap \Delta p)$	$\leftrightarrow (f \rightarrow f)$	"B" symmetric	
SFT9:	$\vdash \forall p(\sqcap p \rightarrow \sqcap\sqcap p)$	$\leftrightarrow$ T	"S4" transitive	
SFT10:	$\vdash \forall p(\sqcap p \vee \sqcap \sim \sqcap p)$		$\leftrightarrow$ T	"S5"
SFT11:	$\vdash \forall p(\sqcap p \rightarrow \vdash \sqcap p)$	$\leftrightarrow$ T	"static"	

4.3 Examples

We now give some example proofs of some theorems of Functional Semantics that were be obtained by our sequent logic theorem prover using the rewrite rules for our modal logic and the axioms of the semantical system.

FT5:

```
FT5A:
1 ∀w ◇fw ⇒ ⊢ ∀p (⊓p → △p)                                              :⇒→, D⊢
3 ∀w ◇fw ⇒ ∀x1 ⊦ x1 ∀p (⊓p → △p)                                       :⇒∀, ⊢∀
5 ∀w ◇fw ⇒ ∀x1 ⊦ x1 ∀p (⊓p → △p)                                       :⇒∀, ⊢→
7 ∀w ◇ fw ⇒ ∀p ⊦ c1 (⊓p → △p)                                          :⇒→, ⊢⊓
9 ⊢ f c1 c2, ∀w ◇ fw ⇒ ⊦ c1 △ c2                                       :D⊢
10 ⊢ (f c1 → c2), ∀w ◇ fw ⇒ ⊦ c1 △ c2                                  :D⊢
11 ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ fw ⇒ ⊦ c1 △ c2                          :D△
12 ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ f2 ⇒ ∼ ⊦ c1 ⊓ ∼ c2                      :→∼, ⊢⊓
14 ⊢ f c1 ∼ c2, ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ fw ⇒                        :D⊢
15 ⊢ (f c1 → ∼ c2), ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ fw ⇒                   :D⊢
16 ∀ x3 ⊦ x3 (f c1 → ∼ c2), ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ f2 ⇒           :∀⇒, ⊢→

18 (1), ⊦ *1 f c1 → ⊦ *1 ∼ c2, ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ fw ⇒       :→⇒

19    (1), ⊦ *1 ∼ c2, ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ fw ⇒                  :⊢∼
20    (1), ∼ ⊦ *1 c2, ∀ x2 ⊦ x2 (f c1 → c2), ∀w ◇ f2 ⇒                  :∼⇒, ∀⇒, ⊢→

24    (1),(2), ⊦ *2 f c1 → ⊦ *2 c2, ∀w ◇ f2 ⇒ ⊦ *1 c2                    :→⇒

25          (1),(2), ⊦ *2 c2, ∀w ◇ f2 ⇒ ⊦ *1 c2                          :∀⇒, D◇

28          (1),(2),(3), ⊦ *2 c2, ∼ ⊢ ∼ f *3 ⇒ ⊦ *1 c2                   :∼⇒, D⊢
30          (1),(2),(3), ⊦ *2 c2, ⇒ ∀ x4 ⊦ x4 ∼ f *3, ⊦ *1 c2            :⇒∀, ⊢∼
32          (1),(2),(3), ⊦ *2 c2 ⇒ ∼ ⊦ (c3 *3) f *3, ⊦ *1 c2             :⇒∼

33          (1),(2),(3), ⊦ (c3 *3) f *3, ⊦ *2 c2 ⇒ ⊦ *1 c2               :unify
                T *2 := *1
35          (1),(2), ∀w ◇ f2 ⇒ ⊦ *2 c2, ⊦ *1 c2                          :∀⇒, D◇

37          (1),(2),(4), ∼ ⊢ ∼ f *4 ⇒ ⊦ *2 c2, ⊦ *1 c2                   :∼⇒, D⊢
39          (1),(2),(4), ⇒ ∀ x5 ⊦ x5 ∼ f *4, ⊦ *2 c2, ⊦ *1 c2            :⇒∀, ⊢∼
41          (1),(2),(4), ⇒ ∼ ⊦ (c4 *4) f *4, ⊦ *2 c2, ⊦ *1 c2            :⇒∼

            (1),(2),(4), ⊦ (c4 *4) f *4 ⇒ ⊦ *2 c2, ⊦ *1 c2               :unify
                T *4 := c1, *1 := (c4 *4), *2 := *1

44    (1), ∀ x2 ⊦ x2 ⊢ x2 (f c1 → c2), ∀w ◇ f2 ⇒ ⊦ *1 f c1              :∀⇒, ⊢→

46    (1),(5), ⊦ *5 f c1 → ⊦ *5 c2, ∀w ◇ f2 ⇒ ⊦ *1 f c1                  :→⇒

47          (1),(5), ⊦ *5 c2, ∀w ◇ f2, ⇒ ⊦ *1 f c1                       :∀⇒, D◇

50          (1),(5),(6), ⊦ *5 c2, ∼ ⊢ ∼ f *6 ⇒ ⊦ *1 f c1                 :∼⇒, D⊢
52          (1),(5),(6), ⊦ *5 c2 ⇒ ∀ x6 ⊦ x6 ∼ f *6, ⊦ *1 f c1           :⇒∀, ⊢∼
54          (1),(5),(6), ⊦ *5 c2 ⇒ ∼ ⊦ x6 f *6, ⊦ *1 f c1                :⇒∼
```

55 (1),(5),(6), ⊢ (c5 *6) f *6, ⊢ *5 c2 ⇒ ⊢ *1 f c1 :unify

57 (1),(5), ∀w ⋄ f2 ⇒ ⊢ *5 f c1, ⊢ *1 f c1 :∀⇒, D⋄

59 (1),(5),(7), ∼ ⊢ ∼ f *7 ⇒ ⊢ *5 f c1, ⊢ *1 f c1 :∼⇒, D⊢
61 (1),(5),(7), ⇒ ∀ x7 ⊢ x7 ∼ f *7, ⊢ *5 f c1, ⊢ *1 f c1 :⇒∀, ⊢∼
63 (1),(5),(7), ⇒ ∼ ⊢ (c6 *7) f *7, ⊢ *5 f c1, ⊢ *1 f c1 :⇒∼

64 (1),(5),(7), ⊢ (c5 *6) f *6 ⇒ ⊢ *5 f c1, ⊢ *1 f c1 :unify
*7 := c1, *5 := (c6 *7)
T

θ starts here
(6), ∀(x2 (6 *8) ⊢ x2 f c1 → c2, ∀(x3 c4 *9) ⊢ x3 f c1 → ∼ c2,

⊢ (c5 *6) f *6, ⊢ c6 c2, ⊢ *8 (f c1 → c2), ⊢ *9 (f c1 → ∼ c2) ⇒ ⊢ c4 f c1

66 (6),(8),(9), ⊢ c6 c2, ⊢ *9 (f c1 → ∼ c2), ⊢ *8 f c1 → ⊢ *8 c2 ⇒ ⊢ c4 f c1 :⊢→
67 (6),(8),(9), ⊢ (c5 *6) f *6,⊢ c6 c2,⊢ *9 f c1 → ⊢ *9 ∼ c2, ⊢ *8 f c1 →
⊢ *8 c2 ⇒ ⊢ c4 f c1 :→⇒

68 (6),(8),(9), ⊢ (c5 *6) f *6, ⊢ c6 c2, ⊢ *9 ∼ c2, ⊢ *8 f c1, ⊢ *8 c2 ⇒ ⊢ c4 f c1 :⊢∼
69 (6),(8),(9),⊢ (c5 *6) f *6, ⊢ c6 c2, ∼ ⊢ *9 c2, ⊢ *8 f c1, ⊢ *8 c2 ⇒ ⊢ c4 f c1 :∼⇒,→⇒

(6),(8),(9), ⊢ (c5 *6) f *6, ⊢ c6 c2, ⊢ *8 c2, ⇒ ⊢ *9 c2, ⊢ c4 f c1 :unify
*9 := c6, *8 := *9
T
(6),(8),(9), ⊢ (c5 *6) f *6, ⊢ c6 c2 ⇒ ⊢ *8 f c1, ⊢ *9 c2, ⊢ c4 f c1 :unify
*6 := c6, *9 := (c5 *6), *8 := *9
T
(6), (8), (9),⊢ (c5 *6) f *6, ⊢ c6 c2,⊢ *8 f c1 → ⊢ *8 c2 ⇒ c4 f c1, ⊢ *9 f c1 :→⇒

(6),(8),(9), ⊢ (c5 *6) f *6, ⊢ c6 c2, ⊢ *8 c2 ⇒ ⊢ c4 f c1, ⊢ *9 f c1 :unify
*6 := c1, *9 := (c5 *6), *8 := *9
T

(6),(8),(9), ⊢ (c5 *6) f *6, ⊢ c6 c2 ⇒ ⊢ *8 f c1, ⊢ *9 f c1, ⊢ c4 f c1 :unify
*6 := c1, *9 := (c5 *6), *8 := *9
T

FT5B:

```
1 ⊢ (∀p ⊓p → △p) ⇒ ∀w ⋄ f2                                         : ⇒→, D⊢
3 ∀ x1 ⊢ x1 (∀p ⊓p → △p) ⇒ ∀w ⋄ fw                                  : ∀, D⋄
5 ∀ x1 ⊢ x1 (∀p ⊓p → △p) ⇒ ∼ ⊢ ∼ f c1                               : ⇒∼, D⊢
7 ∀ x2 ⊢ x2 ∼ f c1, ∀ x1 ⊢ x1 (∀p ⊓p → △p) ⇒                        : ∀⇒, ⊢∼

9 (1), ∼ ⊢ *1 f c1, ∀ x1 ⊢ x1 (∀p ⊓p → △p) ⇒                        : ∼⇒, ∀⇒, ⊢∀

13 (1),(2), ∀p ⊢ *2 (⊓p → △p) ⇒ ⊢ *1 f c1                           : ∀⇒, ⊢→

15 (1),(2),(3), ⊢ *2 ⊓ *3 → ⊢ *2 △ *3 ⇒ ⊢ *1 f c1                   : →⇒

16    (1),(2),(3) ⊢ *2 △ *3 ⇒ ⊢ *1 f c1                             : D△
17    (1),(2),(3) ∼ ⊢ *2 ⊓ ∼ *3 ⇒ ⊢ *1 f c1                         : ∼⇒, ⊢⊓
19    (1),(2),(3) ⇒ ⊢ f *2 ∼ *3, ⊢ *1 f c1                          : D⊢
20    (1),(2),(3) ⇒ ⊢ (f *2 → ∼ *3), ⊢ *1 f c1                      : D⊢
21    (1),(2),(3) ⇒ ∀ x3 ⊢ x3 (f *2 → ∼ *3), ⊢ *1 f c1              : ⇒∀, ⊢→
23    (1),(2),(3) ⇒ ⊢ (c2 *3 *2) f *2 → ⊢ (c2 *3 *2) ∼ *3, ⊢ *1 f c1   : ⇒→, ⊢∼
26    (1),(2),(3) ⊢ (c2 *3 *2) f *2 ⇒ ∼ ⊢ (c2 *3 *2) *3, ⊢ *1 f c1    : ⇒∼
27    (1),(2),(3) ⊢ (c2 *3 *2) *3, ⊢ (c2 *3 *2) f *2 ⇒ ⊢ *1 f c1      : unify
          *2 := c1, *1 := (c2 *3 *2), *3 := f c1
          T

28    (1),(2),(3) ⇒ ⊢ *2 ⊓ *3, ⊢ *1 f c1                            : ⊢⊓
29    (1),(2),(3) ⇒ ⊢ f *2 *3, ⊢ *1 f c1                            : D⊢
30    (1),(2),(3) ⇒ ⊢ (f *2 → *3), ⊢ *1 f c1                        : D⊢
31    (1),(2),(3) ⇒ ∀ x4 ⊢ x4 (f *2 → *3), ⊢ *1 f c1                : ⇒∀, ⊢→
33    (1),(2),(3) ⊢ (c3 *3 *2) f *2 → ⊢ (c3 *3 *2) *3, ⊢ *1 f c1      : ⇒→
35    (1),(2),(3) ⊢ (c3 *3 *2) f *2 → ⊢ (c3 *3 *2) *3, ⊢ *1 f c1      : unify
          *3 := f *2, *2 := c1, *1 := (c2 *3 *2)
          T
```

FT7:

FT7A:

```
1 ∀w ⊢ w f w ⇒ ⊢ (∀p ⊓p →p)                          : ⇒→, D⊢
3 ∀w ⊢ w f w ⇒ ∀ x1 ⊢ x1 (∀p ⊓p →p)                  : ⇒∀, ⊢∀
5 ∀w ⊢ w f w ⇒ ∀ p ⊢ c1 ⊓p → ⊢ c1 p                  : ⇒∀, ⊢→
7 ∀w ⊢ w f w ⇒ ⊢ c1 ⊓ c2 → ⊢ c1 c2                   : ⇒→, ⊢⊓
9 ⊢ f c1 c2, ∀w ⊢ w fw ⇒ ⊢ c1 c2                     : D⊢
10 ⊢ (f c1 → c2), ∀w ⊢ w fw ⇒ ⊢ c1 c2                : D⊢
11 ∀ x2 ⊢ x2 (f c1 → c2) ∀w ⊢ w fw ⇒ ⊢ c1 c2         : ∀⇒, ⊢→

14 (1), ⊢ *1 f c1 → ⊢ *1 c2, ∀w ⊢ w fw ⇒ ⊢ c1 c2     : →⇒

15     (1), ⊢ *1 c2, ∀w ⊢w fw ⇒ ⊢ c1 c2              : ∀⇒

17         (1),(2), ⊢ *1 c2, ⊢ *1 c2 ⇒ ⊢ c1 c2       : unify
               T *1 := c1

20         (1), ∀w ⊢w fw ⇒ ⊢ *1 f c1, ⊢ c1 c2        : ∀⇒

21         (1),(3) ⇒ ⊢ *1 f c1, ⊢ c1 c2              : unify
               T *3 := c1, *1 := c1
```

FT7B:

```
1 ⊢ (∀p ⊓p →p) ⇒ ∀w ⊢w f2                                              : ⇒→, D⊢
3 ∀ x1 ⊢ x1 (∀p ⊓p →p) ⇒ ∀w ⊢w fw                                     : ⇒∀, ∀⇒, ∀⊢

7 (1), ∀p ⊢ *1 (⊓p →p) ⇒ ⊢ c1 f c1                                    : ∀⇒, ⊢→

9 (1),(2), ⊢ *1 ⊓ *2 → ⊢ *1 *2 ⇒ ⊢ c1 f c1                             : →⇒

10    (1),(2), ⊢ *1 *2 ⇒ ⊢ c1 f c1                                     : unify
         *1 := c1, *2 := f c1
         T

11    (1),(2) ⇒ ⊢ *1 ⊓ *2, ⊢ c1 f c1                                   : ⊢⊓
12    (1),(2) ⇒ ⊢ f *1 *2, ⊢ c1 f c1                                   : D⊢
13    (1),(2) ⇒ ⊢ (f *1 → *2), ⊢ c1 f c1                               : D⊢
14    (1),(2) ⇒ ∀ x2 ⊢ x2 (f *1 → *2), ⊢ c1 f c1                       : ⇒∀, ⊢→
16    (1),(2) ⇒ ⊢ (c2 *2 *1) f *1 → ⊢ (c2 *2 *1) *2, ⊢ c1 f c1         : ⇒→
18    (1),(2) ⇒ ⊢ (c2 *2 *1) f *1 ⇒ ⊢ (c2 *2 *1) *2, ⊢ c1 f c1         : unify
         T *2 := f c1, *1 := c1
```

Note: although it is obvious that we now have a proof, the theorem prover must reinstantiate and go through another theta level because it doesn't check to see if *2 and *1 are equal.

FT9:

FT9A:

```
1 ∀a ∀b (⊢b fa → ⊢ fb fa) → ⊢ ∀p ⊓p → ⊓⊓p                                          :⇒→, D⊢
3 ∀a ∀b (⊢b fa → ⊢ fb fa) ⇒ ∀x ⊢ x1 (∀p ⊓p → ⊓⊓p)                                    :⇒∀, ⊢∀
5 ∀a ∀b (⊢b fa → ⊢ fb fa) ⇒ ∀p ⊢ c1 (⊓p → ⊓⊓p)                                       :⇒∀, ⊢→
7 ∀a ∀b (⊢b fa → ⊢ fb fa) ⇒ ⊢ c1 (⊓p → ⊓⊓p)                                          :⇒→, ⊢⊓
9 ⊢ f c1 c2, ∀a ∀b (⊢b fa → ⊢ fb fa) ⇒ ⊢ c1 ⊓⊓ c2                                    :D⊢
10 ⊢ (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fb fa) ⇒ ⊢ c1 ⊓⊓ c2                               :D⊢
11 ∀ x2 ⊢ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ c1 ⊓⊓ c2                       :⊢⊓
12 ∀ x2 ⊢ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ f c1 ⊓ c2                      :D⊢
13 ∀ x2 ⊢ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ (f c1 → ⊓ c2)                  :D⊢
14 ∀ x2 ⊢ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ∀ x3 ⊢ x3 (f c1 → ⊓ c2)          :⇒∀, ⊢→
16 ∀ x2 ⊢ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ c3 f c1 → ⊢ c3 ⊓ c2            :⇒→, ⊢⊓
19 ⊢ c3 f c1, ∀ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ fa fb) ⇒ ⊢ f c3 c2       :D⊢
20 ⊢ c3 f c1, ∀ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ fa fb) ⇒ ⊢ (f c3 → c2)  :D⊢
21 ⊢ c3 f c1, ∀ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ∀ x4 ⊢ x4 (f c3 → c2)     :⇒∀, ⊢→
23 ⊢ c3 f c1, ∀ x2 (f c1 → c2), ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ c4 f c3 → ⊢ c4 c2      :⇒→, ∀⇒, ⊢→

28 ⊢ c4 f c3, ⊢ c3 f c1, (1), ⊢ *1 f c1 → ⊢ *1 c2, ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ c4 c2   :→⇒

29   (1), ⊢ c4 f c3, ⊢ c3 f c1, ⊢ *1 c2, ∀a ∀b (⊢b fa → ⊢ fa fb) ⇒ ⊢ c4 c2   :∀⇒, ∀⇒, →⇒

33      (1),(2),(3), ⊢ *1 c2, ⊢ c4 f c3, ⊢ c3 f c1, ⊢ f *3 f *2 ⇒ ⊢ c4 c2                    :D⊢
34      (1),(2),(3), ⊢ *1 c2, ⊢ c4 f c3, ⊢ c3 f c1, ⊢ (f *3 → f *2) ⇒ ⊢ c4 c2                :D⊢
35      (1),(2),(3), ⊢ *1 c2, ⊢ c4 f c3, ⊢ c3 f c1, ∀ x5 ⊢ x5 (f *3 → f *2) ⇒ ⊢ c4 c2        :∀⇒, ⊢→

37      (1),(2),(3), ⊢ *1 c2, ⊢ c4 f c3, ⊢ c3 f c1, ⊢ *4 f *3 → ⊢ *4 f *2 ⇒ ⊢ c4 c2          :→⇒

38      (1),(2),(3), ⊢ *1 c2, ⊢ c4 f c3, ⊢ c3 f c1, ⊢ *4 f *2 ⇒ ⊢ c4 c2                      :unify
          T *1 := c4
39      (1),(2),(3), ⊢ *1 c2, ⊢ c4 f c3, ⊢ c3 f c1, ⇒ ⊢ *3 f *2, ⊢ c4 c2                     :unify
          T *1 := c4
40      (1),(2),(3), ⊢ *1 c2, ⊢ c4 c3, ⊢ c3 f c1, ⇒ ⊢ *3 f *2, ⊢ c4 c2                       :unify
          T *1 := c4
42   (1), ⊢ c4 f c3, ⊢ c3 f c1, ∀a ∀b (⊢b fa → ⊢ fb fa) ⇒ ⊢ c4 c2, ⊢ c4 c2       :∀⇒, ∀⇒, →⇒

45       (1),(5),(6), ⊢ c4 f c3, ⊢ c3 f c1, ⊢ f *6 f *5 ⇒ ⊢ *1 f c1, ⊢ c4 c2                 :D⊢
46       (1),(5),(6), ⊢ c4 f c3, ⊢ c3 f c1, ⊢ (f *6 → f *5) ⇒ ⊢ *1 f c1, ⊢ c4 c2             :D⊢
47       (1),(5),(6), ⊢ c4 f c3, ⊢ c3 f c1, ∀ x6 ⊢ x6 (f *6 → f *5) ⇒ ⊢ *1 f c1, ⊢ c4 c2     :∀⇒, ⊢→

49       (1),(5),(6),(7), ⊢ c4 f c3, ⊢ c3 f c1, ⊢ *7 f *6 → ⊢ *7 f *5 ⇒ ⊢ *1 f c1, ⊢ c4 c2   :→⇒

50           (1),(5),(6),(7), ⊢ c4 f c3, ⊢ c3 f c1, ⊢ *7 f *6 ⇒ ⊢ *1 f c1, ⊢ c4 c2            :unify
               T *5 := c1, *7 := c4, *1 := c4

             (1),(5),(6),(7), ⊢ c4 f c3, ⊢ c3 f c1 ⇒  ⊢ *7 f *6, ⊢ *1 f c1, ⊢ c4 c2           :unify
                T *6 := c3, *5 := c1, *7 := c4, *1 := c4
         (1),(5),(6), ⊢ c4 f c3, ⊢ c3 f c1 ⇒  ⊢ *6 f *5, ⊢ *1 f c1, ⊢ c4 c2                    :unify
            T *6 := c3, *5 := c1, *7 := c4, *1 := c4
```

FT9B:

```
1 ⊢ (∀p ⊓p → ⊓⊓p) ⇒ ∀a ∀b (⊢b fa → fb fa)                                    :⇒→, D⊢
3 ∀ x1 ⊢ x1 (∀p ⊓p → ⊓⊓p) ⇒ ∀a ∀p (⊢b fa → ⊢ fb fa)                         :⇒∀, ⇒∀, ⇒→, D⊢
8 ⊢ c2 f c1, ∀ x1 ⊢ x1 (∀p ⊓p → ⊓⊓p) ⇒ ⊢ (f c2 → f c1)                        :D⊢
9 ⊢ c2 f c1, ∀ x1 ⊢ x1 (∀p ⊓p → ⊓⊓p) ⇒ ∀ x2 ⊢ x2 (f c2 → f c1)                 :⇒∀, ⊢⇒
11 ⊢ c2 f c1, ∀ x1 ⊢ x1 (∀p ⊓p → ⊓⊓p) ⇒ ⊢ c3 f c2 → ⊢ c3 f c1                 :⇒→, ∀⇒, ⊢∀

16 (1), ⊢ c3 f c2, ⊢ c2 f c1, ∀p ⊢ *1 (⊓p → ⊓⊓p), ⇒ ⊢ c3 f c1                  :∀⇒, ⊢→

18 (1),(2), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ =1 ⊓ *2 → ⊢ *1 ⊓⊓ *2 ⇒ ⊢ c3 f c1            :→⇒

19    (1),(2), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ *1 ⊓⊓ *2 ⇒ ⊢ c3 c1                       :⊢⊓
20    (1),(2), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ f *1 ⊓ *2 ⇒ ⊢ c3 c1                      :D⊢
21    (1),(2), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ (f *1 → ⊓ *2) ⇒ ⊢ c3 c1                  :D⊢
22    (1),(2), ⊢ c3 f c2, ⊢ c2 f c1, ∀ x3 ⊢ x3 (f *1 → ⊓ *2) ⇒ ⊢ c3 c1          :∀⇒, ⊢→

24    (1),(2), ⊢ c3 f c2, ⊢ c2 f c1, ≃ *3 f *1 → ⊢ *3 ⊓ *2 ⇒ ⊢ c3 c1           :→⇒

25        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ *3 ⊓ *2 ⇒ ⊢ c3 c1                :⊢⊓
26        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ f *3 *2 ⇒ ⊢ c3 c1                :D⊢
27        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ (f *3 → *2) ⇒ ⊢ c3 c1            :D⊢
28        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1, ∀ x4 ⊢ x4 (f *3 → *2) ⇒ ⊢ c3 c1    :∀⇒, ⊢→

30        (1),(2),(3),(4), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ *4 f *3 → ⊢ *4 *2 ⇒ ⊢ c3 c1  :→⇒

31            (1),(2),(3),(4), ⊢ c3 f c2, ⊢ c2 f c1, ⊢ *4 *2 ⇒ ⊢ c3 c1          :unify
                 T *3 := c2, *2 := f c1, *4 := c3

33            (1),(2),(3),(4), ⊢ c3 f c2, ⊢ c2 f c1, ⇒ ⊢ *4 f *3, ⊢ c3 f c1
                 T *1 := c1, *3 := c2, *2 := f c1, *4 := c3
34        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1 ⇒ ⊢ *1 ⊓ *2, ⊢ c3 f c1               :⊢⊓
35        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1 ⇒ ⊢ f *1 *2, ⊢ c3 f c1               :D⊢
36        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1 ⇒ ⊢ (f *1 → *2), ⊢ c3 f c1           :D⊢
37        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1 ⇒ ∀ x5 ⊢ x5 (f *1 → *2), ⊢ c3 f c1   :⇒→
39        (1),(2),(3), ⊢ c3 f c2, ⊢ c2 f c1 ⇒ ⊢ (c4 *2 *1) f *1 → ⊢ (c4 *2 *1) *2, ⊢ c3 f c1
41        (1),(2),(3), ⊢ (c4 *2 *1) f *1, ⊢ c3 f c2, ⊢ c2 f c1 ⇒ ⊢ (c4 *2 *1) *2, ⊢ c3 f c1   :unify
              T *1 := c1, *3 := c2, *2 := f c1, *4 := c3
```

4.4 Results

THEOREM	OUTCOME	TIME	GC	CONSES	SEQUENTS	THETAKT
FT0	T	296	0	758	14	0
FT1	T	807	0	2237	41	0
FT2	F	-	-	-	-	-
FT3	T	780	0	2197	44	0
FT4	T	539	0	1192	24	0
FT5A	T	2960	1	5172	77	1
FT5B	T	603	0	1580	35	0
FT5*A	T	3014	1	4584	57	1
FT5*B	T	536	0	1387	30	0
FT6A	T	1037	0	2480	48	0
FT6B	T	916	1	2443	48	0
FT6*A	T	4249	1	6769	72	1
FT6*B	F	-	-	-	-	-
FT7A	T	415	0	1092	20	0
FT7B	T	1641	0	3521	54	1
FT8A	T	855	0	1983	37	0
FT8B	F	-	-	-	-	-
FT9A	T	1391	0	3040	52	0
FT9B	T	11608	3	14246	171	1
FT9*A	T	1276	0	2817	46	0
FT9*B	T	13736	4	13988	168	1
FT10A	T	3346	1	3047	54	0
FT10B	F	-	-	-	-	-
FT10*A	T	2693	1	2834	48	0
FT10*B	F	-	-	-	-	-
FT11A	T	954	0	2409	44	0
FT11B	T	2516	1	3834	56	1
FT11*A	T	1848	1	2159	38	0
FT11*B	T	2623	1	3710	53	1

THEOREM	OUTCOME	TIME	GC	CONSES	SEQUENTS	THETAKT
SFT0	T	248	0	598	15	0
SFT1	T	903	0	2230	41	0
SFT2	F	-	-	-	-	-
SFT3	T	858	0	26273	44	0
SFT4	T	446	0	2176	24	0
SFT5A	T	640	0	1522	34	0
SFT5B	T	775	0	1858	39	0
SFT6A	T	958	0	2335	49	0
SFT6B	F	-	-	-	-	-
SFT7A	T	386	0	1122	21	0
SFT7B	T	506	0	1122	21	0
SFT8A	T	970	0	2240	45	0
SFT8B	F	-	-	-	-	-
SFT9	T	612	0	1404	29	0
SFT10	T	560	0	1414	31	0
SFT11	T	528	0	1121	25	0

5. REFERENCES

1. Brown, F. M., "A Deductive System for Elementary Arithmetic," *2nd AISB Conference Proceedings*, Edinburgh, July 1976.

2. Brown, F. M., "The Role of Extensible Deductive Systems in Mathematical Reasoning," *2nd AISB Conference Proceedings*, Edinburgh, July 1976.

3. Brown, F. M., "Doing Arithmetic without Diagrams," *Artificial Intelligence*, Vol. 8, Spring 1977.

4. Brown, F. M., "A Theorem Prover for Elementary Set Theory," *5th International Joint Conference on Artificial Intelligence*, MIT, August 1977. Also the abstract is in the *Workshop on Automatic Deduction Collected Abstracts*, MIT, August 1977.

5. Brown, F. M., "Towards the Automation of Set Theory and its Logic," *Artificial Intelligence*, Vol. 10, 1978.

6. Brown, F. M., "A Theory of Meaning," Department of Artificial Intelligence Working Paper 16, University of Edinburgh, November 1976.

7. Brown, F. M., "The Theory of Meaning," Department of Artificial Intelligence Research Report 35, University of Edinburgh, June 1977.

8. Brown, F. M., *Towards the Automation of Mathematical Reasoning*, Ph.D. Thesis, University of Edinburgh, 1977.

9. Brown, F. M., "A Sequent Calculus for Modal Quantificational Logic," *3rd AISB/GI Conference Proceedings*, Hamburg, July 1978.

10. Brown, F. M., "A Semantic Theory for Logic Programming," *Colloquia Mathematica Societatis Janos Bolyai, 26 Mathematical Logic in Computer Science*, Salgotarjan, Hungary 1978.

11. Brown, F. M., "An Automatic Proof of the Completeness of Quantificational Logic," Department of Artificial Intelligence Research Report 52, 1978.

12. Brown, F. M., "A Theorem Prover for Metatheory," *4th Conference on Automatic Theorem Proving*, Austin Texas, 1979.

13. Brown, F. M., "Intensional Logic for a Robot, Part 1: Semantical Systems for Intensional Logics Based on the Modal Logic S5+Leib," UT Tech. Report 97, 1979. Invited paper for the Electrotechnical Laboratory Seminar IJCAI 6, Tokyo 1979.