Modal Logics
A Summary of the Well-Behaved

Melvin Fitting
mlflc@cunyvm.cuny.edu
Dept. Mathematics and Computer Science
Lehman College (CUNY), Bronx, NY 10468
Depts. Computer Science, Philosophy, Mathematics
Graduate Center (CUNY), 33 West 42nd Street, NYC, NY 10036 *

September 8, 1989

Modal logic is an enormous subject, and so any discussion of it must limit itself according to some set of principles. Modal logic is of interest to mathematicians, philosophers, linguists and computer scientists, for somewhat different reasons. Typically a philosopher may be interested in capturing some aspect of necessary truth, while a mathematician may be interested in characterizing a class of models having special structural features. For a computer scientist there is another criterion that is not as relevant for the other disciplines: a logic should be 'well-behaved.' This is, admittedly, a vague notion, but some things are clear enough. A logic that can be axiomatized is better than one that can't be; a logic with a simple axiomatization is better yet; and a logic with a reasonably implementable proof procedure is best of all. My current interests are largely centered in computer science, and so I will only discuss well-behaved modal logics. My talk is organized into three sections, depending on the expressiveness of the modal logics considered: propositional; first-order with rigid designators; first-order with non-rigid designators. Even limited as I have said, the subject is a big one, and this talk can be no more than an outline. For a general discussion of propositional modal logic see [2]; for this and first-order modal logic see [9], [10] and [8]; for tableau systems in modal logic see [6].

1 Propositional Modal Logics

1.1 Syntax

We take *propositional formulas* as being built up from *propositional letters* P, P_1, P_2, Q, etc., using *propositional connectives* $\neg$, $\supset$, and the *modal operator* $\Box$ (necessity). The definition is just as in classical logic, with the additional condition: if X is a formula, so is $\Box X$. We sometimes use other

*Research partly supported by NSF Grant CCR-8702307 and PSC-CUNY Grant 668283.

connectives such as $\wedge$ and $\vee$, with the usual definitions, and we also use $\Diamond$ as a defined modal operator, writing $\Diamond X$ for $\neg\Box\neg X$.

1.2 Frames and Models

Relational semantics, due to Kripke [11] and others, plays a role for modal logics comparable to that of truth tables for classical propositional logic. It should be mentioned that there are other kinds of semantics, algebraic and neighborhood. We do not have space to discuss these here.

Definition 1.2.1 A *frame* is a pair $\langle \mathcal{G}, \mathcal{R} \rangle$ where $\mathcal{G}$ is a non-empty set (of *possible worlds*) and $\mathcal{R}$ is a binary relation on $\mathcal{G}$ (of *accessibility*). If $\Gamma\mathcal{R}\Delta$ we say Δ is *accessible from* Γ or *possible relative to* Γ.

Definition 1.2.2 A *propositional model* is a triple $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, v \rangle$ where $\langle \mathcal{G}, \mathcal{R} \rangle$ is a frame and v is a mapping from possible worlds and propositional letters to the set $\{\mathbf{true}, \mathbf{false}\}$ of classical truth values. The model $\langle \mathcal{G}, \mathcal{R}, v \rangle$ is *based on* the frame $\langle \mathcal{G}, \mathcal{R} \rangle$.

We symbolize by $\mathcal{M}, \Gamma \| \!\!- X$ the notion that the formula X is true at the possible world Γ in the model $\mathcal{M}$. The definition is built on Leibniz's idea that necessary truth is truth in all possible worlds.

Definition 1.2.3 Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, v \rangle$ be a model and let $\Gamma \in \mathcal{G}$.

1. for a propositional letter P,
   $\mathcal{M}, \Gamma \| \!\!- P \iff v(\Gamma, P) = \mathbf{true}$;
2. $\mathcal{M}, \Gamma \| \!\!- \neg X \iff$ not-$\mathcal{M}, \Gamma \| \!\!- X$;
3. $\mathcal{M}, \Gamma \| \!\!- X \supset Y \iff$ not-$\mathcal{M}, \Gamma \| \!\!- X$ or $\mathcal{M}, \Gamma \| \!\!- Y$;
4. $\mathcal{M}, \Gamma \| \!\!- \Box X \iff$ for every $\Delta \in \mathcal{G}$ such that $\Gamma\mathcal{R}\Delta, \mathcal{M}, \Delta \| \!\!- X$.

Definition 1.2.4

1. A formula X is *valid in a model* $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, v \rangle$ if $\mathcal{M}, \Gamma \| \!\!- X$ for every $\Gamma \in \mathcal{G}$.
2. X is *valid in a set of models* S if X is valid in each model in S.
3. X is *valid in a set of frames* $\mathbf{F}$ if X is valid in the set of models based on frames in $\mathbf{F}$.

For example, it is easy to check that $\Box(X \supset Y) \supset (\Box X \supset \Box Y)$ is valid in every set of frames; $\Box X \supset X$ is valid in the set of frames having reflexive accessibility relations; and $\Box X \supset \Box\Box X$ is valid in the set of frames having transitive accessibility relations. Such things are standard, and come from [11].

1.3 Logics

The general notion of a logic can be discussed syntactically or semantically. For propositional modal logics the notion of a *normal* logic is a central one, and its characterization is essentially syntactic. (There is a broader class of logics called *regular*; we do not discuss them here.)

Definition 1.3.1 A set N of propositional formulas is a *normal modal logic* if:

1. every tautology is in N;
2. every formula of the form $\Box(X \supset Y) \supset (\Box X \supset \Box Y)$ is in N;
3. N is closed under Modus Ponens, X, $X \supset Y \in N$ implies $Y \in N$;
4. N is closed under Substitution;
5. N is closed under Necessitation, $X \in N$ implies $\Box X \in N$.

For example, the collection of all formulas is a normal modal logic, albiet a trivial one. The intersection of any non-empty family of normal modal logics is a normal modal logic. It follows that there is a smallest normal modal logic; it is called K. Similarly there is a smallest normal modal logic containing all formulas of the form $\Box X \supset X$; it is called T.

Many things are known about normal modal logics. Each one, for instance, has a model that exactly characterizes it. But generally such characteristic models have neither intuitive content nor constructive existence. For a variety of reasons, frames are more natural things to look at when trying to understand a logic.

Proposition 1.3.1 *Let* $\mathbf{F}$ *be any non-empty set of frames, and let* N *be the set of formulas valid in* $\mathbf{F}$*. Then* N *is a normal modal logic.*

Unfortunately the converse of Proposition 1.3.1 is not true (see [10] Chapter 4). Logics that are normal but not characterized by a class of frames are not sufficiently well-behaved for our purposes, and from now on we exclude them.

Definition 1.3.2 A set L of formulas is a *frame-logic* if L is the set of formulas valid in some non-empty set $\mathbf{F}$ of frames.

Generally we will use the same notation, L, for a set of frames and for the frame logic it determines. Context will make clear which is meant. Certain well-known logics are characterized by simple sets of frames. We use the following standard names. (We follow the practice of saying a frame is, say, reflexive, if its accessibility relation is.) K is the set of all frames. T is the set of all reflexive frames. $K4$ is the set of all transitive frames. $S4$ is the set of transitive and reflexive frames. B is the set of transitive and symmetric frames. $S5$ is the set of transitive, symmetric and reflexive frames. There are other standard frame logics. More can be found in [9], [10], [6] and [2].

1.4 Consequence

The notion of logical consequence is considerably more complicated in the modal setting than in the classical because we have the level of worlds, of models, and of frames to choose from. We introduce some special notation to capture the various levels. The notation is from [6].

Definition 1.4.1 Let L be a set of frames, S and U be sets of formulas, and X be a formula. By $S \models_L U \rightarrow X$ we mean: for every frame $\langle \mathcal{G}, \mathcal{R} \rangle$ in L, and for every model $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, v \rangle$ based on this frame in which the members of S are valid, and for every world $\Gamma \in \mathcal{G}$ at which the members of U are true, $\mathcal{M}, \Gamma \Vdash X$.

In the notation $S \models_L U \rightarrow X$ we think of L as specifying the underlying frame logic. S and U are sets of assumptions, axioms. We think of S as *global* assumptions, holding at all worlds; and U as *local* assumptions, holding at particular worlds. The two play quite different roles, a difference that is illustrated by the Deduction Theorem, whose proof we omit. Ultraproducts have the standard meaning from classical model theory. The subframe of $\langle \mathcal{G}, \mathcal{R} \rangle$ generated by $\mathcal{G}_0 \subseteq \mathcal{G}$ is the frame $\langle \mathcal{G}^*, \mathcal{R}^* \rangle$ where $\mathcal{G}^*$ is the smallest subset of $\mathcal{G}$ extending $\mathcal{G}_0$ and closed under the accessibility relation, and $\mathcal{R}^*$ is $\mathcal{R}$ restricted to $\mathcal{G}^*$.

Theorem 1.4.1

1. $S \models_L U \cup \{X\} \rightarrow Y \iff S \models_L U \rightarrow (X \supset Y)$.
2. $S \cup \{X\} \models_L U \rightarrow Y \iff S \models_L U \cup \{X, \Box X, \Box\Box X, \ldots\} \rightarrow Y$, *provided* L *is closed under generated subframes and ultraproducts.*

1.5 Proof Procedures

Axiom systems, also known as Hilbert systems, are very versatile and are well-represented in the literature. There is one fundamental drawback to them, however. They do not lend themselves to proof discovery, and so not to automation either. Gentzen sequent calculi (or the variant, semantic tableaux) exist for a smaller class of logics, but these include several of considerable use and interest [6]. Systems like these lend themselves well to proof discovery, and have been automated. More recently, resolution methods have also been developed. All these come in several versions, and so to keep things relatively simple we will only consider a single frame logic, the logic K.

A Hilbert system for K has, as axioms, all tautologies, and all formulas of the form $\Box(X \supset Y) \supset (\Box X \supset \Box Y)$. There are two rules of inference: Modus Ponens, from X and $X \supset Y$ to conclude Y, and the rule of Necessitation, due to Gödel, from X conclude $\Box X$. Note the difference between the necessitation *rule* and the *formula* $X \supset \Box X$, which would trivialize the logic if taken as an axiom.

Completeness and soundness can be proved in the form: X has a proof in this Hilbert system if and only if X is valid in all frames. But a *strong* completeness proof is possible. Two premise introduction rules are needed.

Global Premise Rule for S At any point in a Hilbert proof construction any formula in S may be introduced as a line.

Local Premise rule for S Any member of S may be introduced as a line in a Hilbert proof *provided* it is before any application of the Rule of Necessitation.

Now one can prove: $S \models_K U \to X$ if and only if X has a Hilbert proof allowing members of S as global premises and members of U as local premises.

Tableau systems come in two styles, with prefixes and without. The version without prefixes gives more information, but such tableau systems exist for rather few modal logics. Prefixes give us considerable flexibility as we add quantifiers and other features. We briefly sketch both kinds of systems, for the logic K only. We begin with the version that does not use prefixes.

A tableau proof is a tree, generally displayed with the root at the top, with formulas as node labels. Tableaux are refutations, so a proof of X begins with $\neg X$. Then the tree 'grows' according to the following rules. A branch containing $\neg\neg Z$ may have Z added to the end. A branch containing $\neg(Z \supset W)$ may have Z and $\neg W$ added to the end. A branch containing $Z \supset W$ forks, with $\neg Z$ added to one side and W added to the other. This completes the classical rules, so we pause and summarize so far. Schematically:

$$\frac{\neg\neg Z}{Z} \qquad \frac{\neg(Z \supset W)}{\begin{array}{c} Z \\ \neg W \end{array}} \qquad \frac{Z \supset W}{\neg Z \mid W}$$

A tableau branch is *closed* if it contains Z and $\neg Z$ for some formula Z. A tableau is *closed* if every branch is. A closed tableau with $\neg X$ at the root is a *proof* of X. Thus far we have a system that proves exactly the classical tautologies [14].

Now for the modal rule — there is only one for K. And it is more complex than the others in that it modifies a whole branch, instead of merely extending it.

Modal Rule for K If $\neg\Box Z$ occurs on a branch then: 1) $\neg Z$ may be added; 2) for each formula of the form $\Box W$ the formula W may be added; 3) all other formulas must be deleted.

Once again, premise rules are straightforward, and we state them informally. A global premise can be added to a branch at any time. A local premise can be added to a branch provided the Modal Rule for K has not been applied on the branch. Then a strong completeness theorem is provable.

Tableau systems of this kind exist for several standard modal logics such as $K4$, T and $S4$, but not for others such as B and $S5$. For these, other devices are possible however. We present *prefixed* tableaux. The idea is to incorporate directly into the system names for possible worlds, and to do so in a way that makes syntactic manipulations of the names correspond to semantic facts of importance to us. The paper [13] has used a somewhat similar device in connection with resolution systems for modal logics. We use as prefixes finite sequences of integers, with the informal idea that the sequence 1,2,1,3 say, names a world accessible from 1,2,1.

Definition 1.5.1

1. A *prefix* is a finite sequence of positive integers.

2. σX is a prefixed formula if σ is a prefix and X is a formula.

3. Adding one positive integer to the end of a prefix produces an *immediate extension*. We use σ, n for the result of adding n to the end of σ.

This time a tableau is a tree with prefixed formulas as labels. A branch is closed if it contains σZ and $\sigma\neg Z$ for some formula Z and prefix σ. The classical branch extension rules are what one might expect.

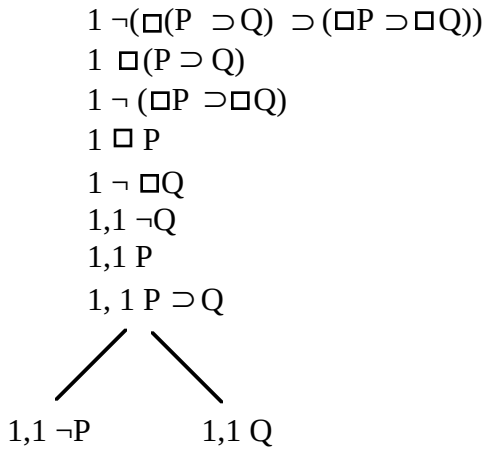
$$\frac{\sigma\ \neg\neg X}{\sigma\ X} \qquad \frac{\sigma\ \neg(X \supset Y)}{\begin{array}{c}\sigma\ X\\ \sigma\ \neg Y\end{array}} \qquad \frac{\sigma\ (X \supset Y)}{\sigma\ \neg X \mid \sigma\ Y}$$

There are two modal rules now. First, if $\sigma\Box Z$ occurs on a branch, σ, nZ may be added, provided σ, n already occurs as a prefix on the branch. Second, if $\sigma\neg\Box Z$ occurs, $\sigma, n\neg Z$ may be added provided σ, n *does not* occur on the branch. Schematically:

$$\frac{\sigma\ \Box X}{\sigma, n\ X} \qquad \frac{\sigma\ \neg\Box X}{\sigma, n\ \neg X}$$

for σ, n used — for σ, n new

A proof of X is a closed tableau with $1\ \neg X$ at the root. The following is an example of a prefixed tableau proof, of $\Box(X \supset Y) \supset (\Box X \supset \Box Y)$. We leave it to you to supply justification.



Although prefixed tableaux give less information, the device has considerable versatility. By varying it slightly, or imposing special conditions, tableaux systems can be created for a wide variety of standard logics, including B and $S5$. Recently [13] has used a similar device, in conjunction with resolution, to produce automated theorem provers for several modal logics.

2 First-Order Modal Logics, Rigid Designators

2.1 Syntax and Semantics

Most treatments of first-order modal logics in the literature assume constant symbols have the same meanings from world to world: they are *rigid designators*. Such logics are not as expressive as one would like, but the proof theory and semantics appropriate for them is quite simple. And it turns out that such logics are natural in the sense that they are what one gets if one simply combines, in a naive way, proof techniques for propositional modal logic with proof techniques for classical first-order logic.

The syntax of propositional logic is extended by adding quantifiers exactly as one does in classical logic. We will take $\forall$ as basic, and think of $\exists$ as defined. It will be simplest not to have function symbols present for now, though we do allow constant symbols.

Semantically the idea is also simple. One allows each world of a Kripke model to have its own domain of quantification. It turns out that some relationships between these domains is often desirable; the most common being to assume that anything existing in a world also exists in any accessible world.

Definition 2.1.1 A *first-order frame* is a triple, $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ where $\langle \mathcal{G}, \mathcal{R} \rangle$ is a frame and $\mathcal{D}$ is a mapping that assigns to each $\Gamma \in \mathcal{G}$ some non-empty set $\mathcal{D}(\Gamma)$, the *domain* of Γ.

Definition 2.1.2 A frame $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ is *monotonic* if, for each $\Gamma, \Delta \in \mathcal{G}$, $\Gamma \mathcal{R} \Delta$ implies $\mathcal{D}(\Gamma) \subseteq \mathcal{D}(\Delta)$. The frame is *constant domain* if $\mathcal{D}(\Gamma) = \mathcal{D}(\Delta)$ for all $\Gamma, \Delta \in \mathcal{G}$.

Next we define the notion of a rigid interpretation in a frame. The notion is a straightforward modification of interpretation from classical logic.

Definition 2.1.3 $\langle \mathcal{F}_C, \mathcal{F}_R \rangle$ is a *rigid interpretation* in the frame $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ provided:

1. $\mathcal{F}_C$ assigns to each constant symbol c an object $\mathcal{F}_C(c)$ that is in the domain of every possible world in $\mathcal{G}$;
2. $\mathcal{F}_R$ assigns to every n-place relation symbol R and every world $\Gamma \in \mathcal{G}$ an n-place relation $\mathcal{F}_R(\Gamma, R)$ on $\mathcal{D}(\Gamma)$.

A *rigid model* is a 5-tuple $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_R \rangle$ where $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ is a first-order frame and $\langle \mathcal{F}_C, \mathcal{F}_R \rangle$ is a rigid interpretation in it.

What makes a model rigid is that the interpretation of a constant symbol does not depend on the possible world. Next, we must deal with free variables, and we do this by a direct adaptation of classical techniques.

Definition 2.1.4 An *assignment* in a rigid model $\mathcal{M}$ is a mapping s that associates with each variable x an object $s(x)$. (It is not required that $s(x)$ be in the domain of every possible world.) We write $s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right]$ for the assignment that is like s except that the variable x is mapped to a.

Since we have no function symbols now, terms are just constant symbols and variables. For an assignment s in a model $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_R \rangle$ we write: $x^{\mathcal{M},s} = s(x)$ for a variable x, and $c^{\mathcal{M},s} = \mathcal{F}_C(c)$ for a constant symbol c. Now truth at a world, under an assignment, is defined in a direct way.

Definition 2.1.5 Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_R \rangle$ be a rigid model, and let s be an assignment. The relation $\mathcal{M}, \Gamma \Vdash X[s]$ is defined as follows:

1. $\mathcal{M}, \Gamma \Vdash R(t_1, \ldots, t_n)[s]$ for an atomic formula $R(t_1, \ldots, t_n) \Leftrightarrow \langle t_1^{\mathcal{M},s}, \ldots, t_n^{\mathcal{M},s} \rangle$ is in the relation $\mathcal{F}_R(\Gamma, R)$;
2. $\mathcal{M}, \Gamma \Vdash \neg X[s] \Leftrightarrow$ not-$\mathcal{M}, \Gamma \Vdash X[s]$;
3. $\mathcal{M}, \Gamma \Vdash (X \supset Y)[s] \Leftrightarrow$ not-$\mathcal{M}, \Gamma \Vdash X[s]$ or $\mathcal{M}, \Gamma \Vdash Y[s]$;
4. $\mathcal{M}, \Gamma \Vdash \Box X[s] \Leftrightarrow \mathcal{M}, \Delta \Vdash X[s]$ for every Δ such that $\Gamma \mathcal{R} \Delta$;
5. $\mathcal{M}, \Gamma \Vdash (\forall x)\Phi[s] \Leftrightarrow \mathcal{M}, \Gamma \Vdash \Phi[s\,[{}^{x}_{a}]]$ for every $a \in \mathcal{D}(\Gamma)$.

Note, by the way, that if the value $s(x)$ of a free variable is not a member of the domain of a particular world, Γ, the definition above makes any *atomic* formula containing it evaluate to false at Γ. Non-atomic formulas are then evaluated according to the other rules.

The notation of logical consequence that we introduced for propositional modal logics can be carried over to the first-order setting, but since things are relatively complicated now and other concerns are primary, we will not do so.

Definition 2.1.6 For a *closed formula* Φ we write $\mathcal{M} \Vdash \Phi[s]$ to mean $\mathcal{M}, \Gamma \Vdash \Phi[s]$ for all worlds Γ of the model $\mathcal{M}$. And we write $\mathcal{M} \Vdash \Phi$ for $\mathcal{M} \Vdash \Phi[s]$ for all assignments s. We say Φ is *valid* in a class $\mathbf{C}$ of rigid models if $\mathcal{M} \Vdash \Phi$ for every $\mathcal{M} \in \mathbf{C}$. Likewise Φ is *valid* in a class of frames provided Φ is valid in the class of models based on those frames.

We continue using K, T, etc. though now K is the class of all *first-order* frames, T is the class of all reflexive first-order frames, and so on.

2.2 Proof Procedures

A device we find useful for first-order proof procedures is *parameters*. These are extra constant symbols, not part of the original language, not used in stating what is to be proved, but used in proofs. Typically if we have established in the course of a proof that something exists having a certain property, we can introduce a parameter as a kind of name for such a thing. Free variables are often used for this purpose as well, but we find the use of parameters simpler, as they will allow us to have only closed formulas appearing in proofs. Parameters will have a place in all the proof procedures we consider.

Hilbert style systems are relatively straightforward. For classical logic only, we can add to the propositional base of the previous section two more pieces of machinery.

$\forall$ **axiom schema** $(\forall x)\Phi(x) \supset \Phi(t)$ is an axiom, for any formula Φ with only x free, and any closed term t.

$\forall$ **rule** Conclude $\Psi \supset (\forall x)\Phi(x)$ from $\Psi \supset \Phi(c)$ provided c is a parameter that does not occur in the closed formula $\Psi \supset (\forall x)\Phi(x)$.

If we add this machinery to the Hilbert system for propositional K we get a first-order system in which the theorems are exactly the closed formulas valid in all first-order frames that are monotonic. Likewise adding the machinery to propositional T axiomatizes the validities of all monotonic, reflexive first-order frames. And so on.

If we want to impose a constant domain requirement on frames, the so-called Barcan formula must be added as an axiom schema. (The converse is, in fact, provable in the systems just described.)

Barcan Formula $(\forall x)\Box\Phi \supset \Box(\forall x)\Phi$

If neither monotonicity nor constant domain conditions are desired, axiomatizations are still possible, but the underlying formulation of classical logic must be modified. We do not give details here. Completeness proofs can be found in [12] (where the ideas originated) and in [9] and [6].

Tableaux systems also adapt readily to quantifiers, but with some surprises. The classical propositional system (without prefixes) is turned into a sound and complete first-order version by the addition of two quantifier rules. In them, t is any closed term, while c is a parameter that has not previously been used on the branch.

$$\frac{(\forall x)\Phi(x)}{\Phi(t)} \quad (\text{any } t) \qquad \frac{\neg(\forall x)\Phi(x)}{\neg\Phi(c)} \quad (c \text{ new parameter})$$

Now, if these rules are added to the tableau system for K, a first-order version results that is sound and complete with respect to the family of *monotonic* frames. Similarly for other logics. In fact, a constant domain version can not be developed using unprefixed tableaux.

We can add similar quantifier rules to the prefixed tableau systems.

$$\frac{\sigma\ (\forall x)\Phi(x)}{\sigma\ \Phi(t)} \quad (\text{any } t) \qquad \frac{\sigma\ \neg(\forall x)\Phi(x)}{\sigma\ \neg\Phi(c)} \quad (c \text{ new parameter})$$

If these rules are added to the prefixed K system we get a first-order version that is sound and complete with respect to the family of *constant domain* frames!

The difference between the two tableau systems can be made intuitively plausible. In the unprefixed version, applications of the Modal rule informally involve a move from one world to another. Parameters introduced by the $\neg\forall$ rule can be thought of as names for things that exist in a world. Such names can be introduced after a move to a different world has been made and, since we can't go backwards to a former world, no use can be made of them in earlier worlds. Monotonicity, not constant domains, is implied.

In the prefixed version however, all worlds are present at the same time, in the form of prefixes naming them. A parameter introduced by a rule applied to a formula with a prefix of σ can subsequently be used in a formula with any other prefix. Constant domains are implied now.

Alternatives to constant domain logics can be handled using prefixed tableaux, by slightly complicating the machinery. We could, for instance, introduce a separate set of parameters for each prefix. In this way monotonic, and no-restrictions logics can be handled. We do not give details here. See [6].

We conclude with an example of a prefixed K tableau proof in the constant domain first-order system, of the Barcan formula $(\forall x)\Box P(x) \supset \Box(\forall x)P(x)$.

$$
\begin{array}{ll}
1 & \neg((\forall x)\Box P(x) \supset \Box(\forall x)P(x)) \\
1 & (\forall x)\Box P(x) \\
1 & \neg\Box(\forall x)P(x) \\
1,1 & \neg(\forall x)P(x) \\
1,1 & \neg P(c) \\
1 & \Box P(c) \\
1,1 & P(c)
\end{array}
$$

3 First-Order Modal Logics, Non-Rigid Designators

3.1 Background

If modal logic is to be a truly expressive tool it must be able to work with non-rigid designators. Such things occur in everyday speech routinely. After all, we easily understand something like "From now on, the world's tallest building will be taller than the world's tallest tree." There are two non-rigid designators involved.

As it happens, allowing non-rigid designators introduces parsing problems. If c is non-rigid, how should we read $\Diamond P(c)$? One way is: there is an alternative world in which the property P holds of the thing that c names in that world. But another quite reasonable reading is: consider the object that c names in this world; in some alternative world the property P holds of it. If c is non-rigid, these can lead to different results. But there are simple natural language sentences that embody both versions.

The problem is that ordinary modal syntax is not expressive enough to make the distinctions we want. Fortunately a solution is easily come by.

In the lambda calculus one makes a conceptual distinction between a *term* like $x + 1$ and the *function* abstracted from it $(\lambda x.x + 1)$. We will make a similar distinction between a *formula* $\Phi(x)$ and the *predicate* abstracted from it $(\lambda x.\Phi(x))$. It is predicates that we will apply to terms. Thus $\Phi(c)$ will not be a formula from now on, but $(\lambda x.\Phi(x))(c)$ will be. Now the two readings of $\Diamond P(c)$ become expressible, as $\Diamond(\lambda x.P(x))(c)$ and $(\lambda x.\Diamond P(x))(c)$. We refer to this use of λ notation as *predicate abstraction*.

These ideas originated in [15] and [16], were further explored in [3], [4] and [5], and have recently been extended in [7].

3.2 Syntax

In the interests of having as rich a language as possible, we now allow function symbols. Thus, from now on, *terms* are built up from constant symbols and variables using function symbols.

For convenience we restrict the notion of *atomic formula*. From now on an atomic formula is something of the form $R(x_1, \ldots, x_n)$ where R is an n-place relation symbol and $x_1, \ldots, x_n$ are *variables*. What we used to write as $P(c)$, say, we will now write as $(\lambda x.P(x))(c)$.

Finally, the definition of formula is as before, with one more clause: if Φ is a formula, x is a variable, and t is a term, $(\lambda x.\Phi)(t)$ is a formula. The free variable occurrences in $(\lambda x.\Phi)(t)$ are those of Φ except for occurrences of x, together with those of t.

3.3 Semantics

In order to keep the discussion from getting too cluttered, from now on we assume all frames are constant domain. This is not essential — the methods work in more general settings, but they are more complicated to explain.

We use the same notion of constant domain first-order frame as before. Since all possible worlds have the same domain now, we can refer to the domain of a *frame*, meaning the domain common to all possible worlds.

Definition 3.3.1 The triple $\langle \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R \rangle$ is a *non-rigid interpretation* in the frame $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ provided:

1. $\mathcal{F}_C$ assigns to each constant symbol c and world $\Gamma \in \mathcal{G}$ a member $\mathcal{F}_C(\Gamma, c)$ that is in the domain of the frame;
2. $\mathcal{F}_F$ assigns to each n-place function symbol f and world Γ an n-place function $\mathcal{F}_F(\Gamma, f)$ on the domain of the frame.
3. $\mathcal{F}_R$ assigns to every n-place relation symbol R and every world $\Gamma \in \mathcal{G}$ an n-place relation $\mathcal{F}_R(\Gamma, R)$ on the domain of the frame.

A *non-rigid model* is a 6-tuple $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R \rangle$ where $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ is a first-order frame and $\langle \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R \rangle$ is a non-rigid interpretation in it.

Notice that the meaning associated with constant symbols and with function symbols is now world dependent. On the other hand, we carry over the definition of *assignment* from before. In effect, this means we are going to think of quantifiers as quantifying over *things*, not over *names* for them.

Now we give meanings to arbitrary terms. We use the notation $t^{\Gamma,\mathcal{M},s}$ to denote the object named by the term t in the world Γ of the model $\mathcal{M}$ under the assignment s.

Definition 3.3.2 Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{F}_C, \mathcal{F}_F, \mathcal{F}_R \rangle$ be a non-rigid model, and let s be an assignment in it.

1. for a constant symbol c, $c^{\Gamma,\mathcal{M},s} = \mathcal{F}_C(\Gamma, c)$;

2. for a variable x, $x^{\Gamma,\mathcal{M},s} = s(x)$;
3. for a function symbol f, $[f(t_1,\ldots,t_n)]^{\Gamma,\mathcal{M},s} = \mathcal{F}_F(\Gamma,f)(t_1^{\Gamma,\mathcal{M},s},\ldots,t_n^{\Gamma,\mathcal{M},s})$.

Finally, the definition of truth, in a model, at a possible world, under an assignment, is the same as before with one additional clause.

$$\mathcal{M},\Gamma\Vdash(\lambda x.\Phi)(t)[s] \text{ if } \mathcal{M},\Gamma\Vdash\Phi[s\,{}^{[x]}_{[a]}] \text{ where } a = t^{\Gamma,\mathcal{M},s}.$$

It is easy to check that this does make the required distinctions. In particular, for a constant symbol c,

$\mathcal{M},\Gamma\Vdash(\lambda x.\Diamond P(x))(c)[s]$ if and only if there is some Δ such that $\Gamma\mathcal{R}\Delta$ and $\mathcal{M},\Delta\Vdash P(x)[s\,{}^{[x]}_{[a]}]$ where $a = \mathcal{F}_C(\Gamma,c)$;

$\mathcal{M},\Gamma\Vdash\Diamond(\lambda x.P(x))(c)[s]$ if and only if there is some Δ such that $\Gamma\mathcal{R}\Delta$ and $\mathcal{M},\Delta\Vdash P(x)[s\,{}^{[x]}_{[a]}]$ where $a = \mathcal{F}_C(\Delta,c)$.

3.4 Proof Procedures

Hilbert systems are much less satisfactory now. A Hilbert system for a more restricted version of modal logic with predicate abstraction was given in [4], but it is complicated and difficult to use. Tableau systems without prefixes seem inapplicable as well. But prefixed tableaux extend readily to encompass predicate abstractions. As usual, we give rules only for the logic K, and recall we are assuming models have constant domains.

We will use the device of attaching subscripts to constant and function symbols in terms. If c is a constant symbol, think of c_σ as the object that c names in the world that the prefix σ names.

Definition 3.4.1 *Object expressions* are defined as follows.

1. If c is a constant symbol and σ is a prefix, c_σ is an object expression.
2. If p is a parameter, p is an object expression.
3. If f is an n-place function symbol, σ is a prefix, and o_1, ..., o_n are object expressions, $f_\sigma(o_1,\ldots,o_n)$ is an object expression.

Now, the propositional tableau rules, both classical and modal, are exactly as before. One of the quantifier rules is modified slightly. Let us say an object expression o is *available* on a tableau branch provided every subscript in it occurs as a prefix on the branch. The $\forall$ rule becomes: if $\sigma\,(\forall x)\Phi(x)$ occurs on a branch then $\sigma\,\Phi(o)$ may be added, for any available object expression o.

Finally we need rules for predicate abstraction. One more piece of notation is useful. For a term t, possibly with subscripts on symbols, we write $t@\sigma$ to denote the result of attaching σ as a subscript to all unsubscripted function symbols in t and to all unsubscripted constant symbols other than parameters.

Predicate Abstraction Rules

1. if $\sigma\,(\lambda x.\Phi(x))(t)$ occurs on a branch then $\sigma\,\Phi(t@\sigma)$ may be added to the end.

2. if $\sigma\, \neg(\lambda x.\Phi(x))(t)$ occurs on a branch then $\sigma\, \neg\Phi(t@\sigma)$ may be added to the end.

We conclude with a proof of $(\forall x)\Box((\lambda y.R(x,y))(f(x)) \supset (\forall x)\Box\neg(\forall y)\neg R(x,y)$.

$$
\begin{array}{ll}
1 & \neg[(\forall x)\Box(\lambda y.R(x,y))(f(x)) \supset (\forall x)\Box\neg(\forall y)\neg R(x,y)] \\
1 & (\forall x)\Box(\lambda y.R(x,y))(f(x)) \\
1 & \neg(\forall x)\Box\neg(\forall y)\neg R(x,y) \\
1 & \neg\Box\neg(\forall y)\neg R(p,y) \\
1 & \Box(\lambda y.R(p,y))(f(p)) \\
1,1 & \neg\neg(\forall y)\neg R(p,y) \\
1,1 & (\forall y)\neg R(p,y) \\
1,1 & (\lambda y.R(p,y))(f(p)) \\
1,1 & R(p, f_{1,1}(p)) \\
1,1 & \neg R(p, f_{1,1}(p))
\end{array}
$$

Soundness and completeness proofs for this tableau system are not difficult. A sketch can be found in [7].

3.5 Extensions

Of course one would like to give modal logic analogs of all the machinery that has been developed for classical logic. In particular, equality should be added, and definite descriptions. Also, something that is less of an issue classically but is more central in a modal setting: we would like to be able to deal with terms that do not designate. As it happens, all these are relatively simple to deal with now that predicate abstraction is available. Equality, for instance, can be treated by simply adding an equality symbol, $=$, to the language and interpreting it as the equality relation at every possible world. There are certain well-known problems with substitutivity of equality in modal contexts. These come down to whether or not one wants to accept as valid the formula $a = b \supset \Box(a = b)$. As it happens, this corresponds to two distinct formulas in notation using predicate abstraction. These are:

1. $(\lambda x, y.(x = y))(a, b) \supset (\lambda x, y.\Box(x = y))(a, b)$

2. $(\lambda x, y.(x = y))(a, b) \supset \Box(\lambda x, y.(x = y))(a, b)$

Interpreting the equality symbol as we suggested above, item 1) is valid while item 2) is not. Further investigation shows it is item 2) that is involved in the morning star/evening star paradox.

Likewise the Russell treatment of definite descriptions can be extended to the modal setting with little difficulty, once predicate abstraction is available. And machinery for non-designating terms is relatively straightforward. A full discussion of this material may be found in [7]. We should also mention [1] which deals with the same problems in a rather different way, using a modal logic of higher types.

References

[1] A. Bressan. *A General Interpreted Modal Calculus*, Yale University Press (1972).

[2] B. F. Chellas. *Modal Logic, an Introduction*, Cambridge University Press (1980).

[3] M. Fitting. An Epsilon-calculus system for first-order S4, *Conference in Mathematical Logic, London '70*, W. Hodges editor, pp 103–110, Springer Lecture Notes in Mathematics, No. 255 (1972).

[4] M. Fitting. A Modal logic analog of Smullyan's fundamental theorem, *Zeitschrift für mathematische Logik und Gründlagen der Mathematik*, vol 19, pp 1–16 (1973).

[5] M. Fitting. A Modal logic epsilon-calculus, *Notre Dame Journal of Formal Logic*, vol 16, pp 1–16 (1975).

[6] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel Publishing Co., Dordrecht (1983).

[7] M. Fitting. Modal logic should say more than it does, submitted for publication.

[8] D. Gabbay and F. Guenthner editors, *Handbook of Philosophical Logic, vol II*, D. Reidel (1984).

[9] G. E. Hughes, M. J. Cresswell. *An Introduction to Modal Logic*, Methuen, London (1968).

[10] G. E. Hughes, M. J. Cresswell. *A Companion to Modal Logic*, Methuen, London (1984).

[11] S. Kripke. Semantical analysis of modal logic I, normal propositional calculi, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, vol 9, pp 67–96 (1963).

[12] S. Kripke. Semantical considerations on modal logics, *Acta Philosophica Fennica, Modal and Many-valued Logics*, pp 83–94 (1963).

[13] H. J. Ohlbach. A Resolution calculus for modal logics, *Ninth International Conference on Automated Deduction (CADE-9)*, E. Lusk and R. Overbeek editors, pp 500–516 (1988).

[14] R. M. Smullyan. *First Order Logic*, Springer-Verlag (1968).

[15] R. Stalnaker, R. Thomason. Abstraction in first-order modal logic, *Theoria*, vol 34, pp 203–207 (1968).

[16] R. Thomason, R. Stalnaker. Modality and reference, *Nous*, vol 2, pp 359–372 (1968).