Higher-Order Modal Logic—A Sketch

Melvin Fitting

Dept. Mathematics and Computer Science
Lehman College (CUNY), Bronx, NY 10468, USA,
fitting@alpha.lehman.cuny.edu,
WWW home page: http://math240.lehman.cuny.edu/fitting

Abstract. First-order modal logic, in the usual formulations, is not sufficiently expressive, and as a consequence problems like Frege's morning star/evening star puzzle arise. The introduction of predicate abstraction machinery provides a natural extension in which such difficulties can be addressed. But this machinery can also be thought of as part of a move to a full higher-order modal logic. In this paper we present a sketch of just such a higher-order modal logic: its formal semantics, and a proof procedure using tableaus. Naturally the tableau rules are not complete, but they are with respect to a Henkinization of the "true" semantics. We demonstrate the use of the tableau rules by proving one of the theorems involved in Gödel's ontological argument, one of the rare instances in the literature where higher-order modal constructs have appeared. A fuller treatment of the material presented here is in preparation.

1 Introduction

Standard first-order classical logic is so well behaved that concentration on it lulls the mind. The behavior of terms provides an instructive example. For one thing, classical terms are always defined—in every classical model all terms have values. But it is well-known that this convention leads to difficulties when definite descriptions are involved since, considered as terms, they don't always denote. As Bertrand Russell noted, "The King of France is not bald" has two quite different, but equally plausible readings. First, it could mean that the King of France has the non-baldness property. This is false since non-existents don't have properties—they don't even have the non-existence property. Second, it could deny the assertion that the King of France has the baldness property. This is true because no bald King of France can be produced. The single string of English words has two possible logical formulations, and conventional first-order syntax cannot distinguish them.

Russell's solution to the problem was to introduce a scoping mechanism—it appears fully developed in *Principia Mathematica*. While he thought of it only in the context of definite descriptions, it is more generally applicable. Using more modern notation, we distinguish between a *formula* Φ and a *predicate abstract* $\langle \lambda x.\Phi \rangle$ drawn from it. Thinking of $B(x)$ as "x is bald," and k as "King of France," we can symbolize the two possible readings mentioned in the previous paragraph

as $\langle\lambda x.\neg B(x)\rangle(k)$ and $\neg\langle\lambda x.B(x)\rangle(k)$. It can be shown that, with a reasonable semantics, these two are equivalent exactly when k denotes, so it is non-denoting terms that force us to use such machinery classically.

Frege noted an analogous problem with intentional contexts, and introduced the notions of "sense" and "denotation" to deal with it. Roughly, this gives terms two kinds of values, what they denote, and what they mean. Of course this is loose. But the introduction of a scoping mechanism also turns out to be of considerable use here. This was done first in [7, 9]. My colleague Richard Mendelsohn and I developed the idea quite fully in [3], and a highly condensed version is available in [2]. But suffice it to say that the notion of predicate abstraction supplies an essential missing ingredient for formal treatments of intentional logics, modal in particular, as well as for cases where terms can lack designations.

Thinking further on the matter, I came to realize that even with predicate abstraction machinery added as outlined above, first-order modal logic is still not as expressive as one would like. And an informal illustration is easy to present.

Assume the word "tall" has a definite meaning—say everybody gets together and votes on which people are tall. The key point is that the meaning of "tall," even though precise, drifts with time. Someone who once was considered tall might not be considered so today.

Now suppose I say, "Someday everybody will be tall." There is more than one ambiguity here. On the one hand I might mean that at some point in the future, everybody alive will be a tall person. On the other hand I might mean that everybody now alive will grow, and so at some point everybody now alive will be a tall person. Let us read modal operators temporally, so that $\Box X$ informally means that X is true and will remain true, and $\Diamond X$ means that X either is true or will be true at some point in the future. Also, let us use $T(x)$ as a tallness predicate symbol. Then the two readings of our sentence are easily expressed in conventional notation as follows.

$$(\forall x)\Diamond T(x) \tag{1}$$

$$\Diamond(\forall x)T(x) \tag{2}$$

Formula (1) refers to those alive now, and says at some point they will all be tall. Formula (2) refers to those alive at some point in the future, and asserts of them that they will be tall. All this is standard, and is not the ambiguity that matters here. The problem is with the adjective "tall." Do we mean that at some point in the future everybody (read either way) will be tall as *they* use the word in the future, or as *we* use the word now? Standard possible world semantics for first-order modal logic is constrained to interpret formulas involving T at a world according to that world's meaning of T. In fact, there is no way of formalizing, using standard first-order modal machinery, the assertion that, at some point in the future, everybody will be tall as *we* understand the term. But this is what is most likely meant if someone says, "Someday everybody will be tall."

The missing piece of machinery to disambiguate the sentence "Someday everybody will be tall," is abstraction, applied at the level of relation symbols, rather than at the level of terms. We get the following *six* versions.

$$(\forall x)\langle \lambda X.\Diamond X(x)\rangle(T) \tag{3}$$
$$(\forall x)\Diamond\langle \lambda X.X(x)\rangle(T) \tag{4}$$
$$\langle \lambda X.\Diamond(\forall x)X(x)\rangle(T) \tag{5}$$
$$\Diamond(\forall x)\langle \lambda X.X(x)\rangle(T) \tag{6}$$
$$\langle \lambda X.(\forall x)\Diamond X(x)\rangle(T) \tag{7}$$
$$\Diamond\langle \lambda X.(\forall x)X(x)\rangle(T) \tag{8}$$

We will introduce semantics for interpreting these shortly, but for the time being we can provide informal readings. Once semantics have been introduced, it can be shown that item (7) is equivalent to (3), and item (8) is equivalent to (6), so we omit readings for them.

> It is true of everybody currently alive that they will be tall, as we understand the word. (3)

> It is true of everybody currently alive that they will be tall, as the word is understood in the future. (4)

> At some point in the future everybody then alive will be tall, as we understand the word. (5)

> At some point in the future everybody will be tall, as the word is understood at that time. (6)

Essentially, in first-order modal logic as it has usually been formulated, all relation symbols are read as if they had narrow scope, and all constants as if they had broad scope. Thus it is as if (4) and (6) were meant by (1) and (2) respectively. There is no way of representing (3) or (5). The machinery for this representation makes for complicated looking formulas. But we point out, in everyday discourse all this machinery is hidden—we infer it from our knowledge of what must have been meant.

Now, why not go the whole way? If we are going to introduce abstraction syntax for terms and for relation symbols, why not treat relation symbols as terms of a higher order. And then why not introduce the whole mechanism of higher-order logic, and do things uniformally all the way up. In fact, this is what we do. The following is a very brief sketch—a much fuller development is in preparation.

2 Syntax

In first-order logic, relation symbols have an *arity*. In higher-order logic this gets replaced by a *typing* mechanism. There are several ways this can be done: logical connectives can be considered primitive, or as constants of the language; a boolean type can be introduced, or not. We adopt a straightforward approach similar to the usual treatments of first-order logic.

Definition 1 (Type). *0 is a type. If $t_1, \ldots, t_n$ are types, $\langle t_1, \ldots, t_n \rangle$ is a type. We systematically use t, t_1, t_2, t', etc. to represent types.*

For each type t we assume we have infinitely many constant and variable symbols of that type. We generally use letters from the beginning of the Greek alphabet to represent variables, with the type written as a superscript: α^t, β^t, γ^t, ... Likewise we generally use letters from the beginning of the Latin alphabet as constant symbols, again with the type written as a superscript: A^t, B^t, C^t, ... We take equality as primitive, so for each type t we assume we have a constant symbol $=^{\langle t,t \rangle}$ of type $\langle t, t \rangle$. Generally types can be inferred from context, and so superscripts will be omitted where possible, in the interests of uncluttered notation.

Sometimes it is helpful to refer to the *order* of a term or formula—first-order, second-order, and so on. Types will play the fundamental role, but order provides a convenient way of referring to the maximum complexity of some construct.

Definition 2 (Order). *The type 0 is of* order *0. And if each of t_1, ..., t_n is of order $\leq k$, with at least one of them being of order k itself, we say $\langle t_1, \ldots, t_n \rangle$ is of* order $k + 1$.

When we talk about the order of a constant or variable, we mean the order of its type. Likewise, once formulas are defined, we may refer to the order of the formula, by which we mean the highest order of a typed part of it.

Next we define the class of formulas, and their free variables. Unlike in the first-order version, the notion of term cannot be defined first; both term and formula must be defined together. And to define both, we need the auxiliary notion of predicate abstract which is, itself, part of the mutual recursion.

Definition 3 (Predicate Abstract). *Suppose Φ is a formula and α_1, ..., α_n is a sequence of distinct variables of types t_1, ..., t_n respectively. We call $\langle \lambda \alpha_1, \ldots, \alpha_n . \Phi \rangle$ a* predicate abstract. *Its type is $\langle t_1, \ldots, t_n \rangle$, and its free variable occurrences are the free variable occurrences in the formula Φ, except for occurrences of the variables α_1, ..., α_n.*

Definition 4 (Term). *Terms of each type are characterized as follows.*

1. *A constant symbol or variable is a term. If it is a constant symbol, it has no free variable occurrences. If it is a variable, it has one free variable occurrence, itself.*

2. *A predicate abstract is a term. Its free variable occurrences were defined above.*

We use τ, with and without subscripts, to stand for terms.

Definition 5 (Formula). *The notion of* formula *is given as follows.*

1. *If τ is a term of type $t = \langle t_1, \ldots, t_n \rangle$, and τ_1, ..., τ_n is a sequence of terms of types t_1, ..., t_n respectively, then $\tau(\tau_1, \ldots, \tau_n)$ is a formula. The free variable occurrences in it are the free variable occurrences of τ, τ_1, ..., τ_n.*
2. *If Φ is a formula so is $\neg\Phi$. The free variable occurrences of $\neg\Phi$ are those of Φ.*
3. *If Φ and Ψ are formulas so is $(\Phi \wedge \Psi)$. The free variable occurrences of $(\Phi \wedge \Psi)$ are those of Φ together with those of Ψ.*
4. *If Φ is a formula and α is a variable then $(\forall\alpha)\Phi$ is a formula. The free variable occurrences of $(\forall\alpha)\Phi$ are those of Φ, except for occurrences of α.*
5. *If Φ is a formula so is $\Box\Phi$. The free variable occurrences of $\Box\Phi$ are those of Φ.*

We use $\vee$, $\supset$, $\Diamond$, $\exists$ as defined symbols, with their usual definitions. Also we use square and curly parentheses, in addition to the official round ones, to aid readability. In addition, since equality plays a fundamental role, we introduce a standard abbreviation for it.

Definition 6 (Equality). *Suppose τ_1 and τ_2 are variables of type t, and $=$ is the equality constant symbol of type $\langle t, t \rangle$. We write $(\tau_1 = \tau_2)$ as an abbreviation for $=(\tau_1, \tau_2)$.*

Example 1. For this example we give explicit type information (in superscripts), until the end of the example. In the future we will generally omit the superscripts, and say in English what is needed to fill them in.

Suppose x^0, $X^{\langle 0 \rangle}$, and $\mathcal{X}^{\langle\langle 0 \rangle\rangle}$ are variables (the first is of order 0, the second is of order 1, and the third is of order 2). Also suppose $\mathcal{P}^{\langle\langle 0 \rangle\rangle}$ and g^0 are constant symbols (the first is of order 2 and the second is of order 0).

1. Both $\mathcal{X}^{\langle\langle 0 \rangle\rangle}(X^{\langle 0 \rangle})$ and $X^{\langle 0 \rangle}(x^0)$ are atomic formulas. All variables present have free occurrences.
2. $\langle \lambda\mathcal{X}^{\langle\langle 0 \rangle\rangle}.\mathcal{X}^{\langle\langle 0 \rangle\rangle}(X^{\langle 0 \rangle}) \rangle$ is a predicate abstract, of type $\langle\langle\langle 0 \rangle\rangle\rangle$. Only the occurrence of $X^{\langle 0 \rangle}$ is free.
3. Since $\mathcal{P}^{\langle\langle 0 \rangle\rangle}$ is of type $\langle\langle 0 \rangle\rangle$, $\langle \lambda\mathcal{X}^{\langle\langle 0 \rangle\rangle}.\mathcal{X}^{\langle\langle 0 \rangle\rangle}(X^{\langle 0 \rangle}) \rangle(\mathcal{P}^{\langle\langle 0 \rangle\rangle})$ is a formula. Only $X^{\langle 0 \rangle}$ is free.
4. $[\langle \lambda\mathcal{X}^{\langle\langle 0 \rangle\rangle}.\mathcal{X}^{\langle\langle 0 \rangle\rangle}(X^{\langle 0 \rangle}) \rangle(\mathcal{P}^{\langle\langle 0 \rangle\rangle}) \supset X^{\langle 0 \rangle}(x^0)]$ is a formula. The only free variable occurrences are those of $X^{\langle 0 \rangle}$ and x^0.
5. $(\forall X^{\langle 0 \rangle})[\langle \lambda\mathcal{X}^{\langle\langle 0 \rangle\rangle}.\mathcal{X}^{\langle\langle 0 \rangle\rangle}(X^{\langle 0 \rangle}) \rangle(\mathcal{P}^{\langle\langle 0 \rangle\rangle}) \supset X^{\langle 0 \rangle}(x^0)]$ is a formula. The only free variable occurrence is that of x^0.
6. $\langle \lambda x^0.(\forall X^{\langle 0 \rangle})[\langle \lambda\mathcal{X}^{\langle\langle 0 \rangle\rangle}.\mathcal{X}^{\langle\langle 0 \rangle\rangle}(X^{\langle 0 \rangle}) \rangle(\mathcal{P}^{\langle\langle 0 \rangle\rangle}) \supset X^{\langle 0 \rangle}(x^0)] \rangle$ is a predicate abstract. It has no free variable occurrences, and is of type $\langle 0 \rangle$.

We need the type machinery to guarantee that what we write is well-formed. Now that we have gone through the exercise above, we can display the predicate abstract without superscripts, as

$$\langle \lambda x.(\forall X)[\langle \lambda \mathcal{X}.\mathcal{X}(X)\rangle(\mathcal{P}) \supset X(x)]\rangle,$$

leaving types to be inferred, or explained in words, as necessary.

3 Models

Just as in the classical setting there are standard higher-order modal models and non-standard ones. Because of space limitations I'll only sketch the standard version, and say a few words later on about the non-standard one.

A higher-order modal model is a structure $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I}\rangle$, and we spend much of the rest of the section saying what each component is.

The pair $\langle \mathcal{G}, \mathcal{R}\rangle$ is a *frame*. In it, $\mathcal{G}$ is a non-empty set of possible worlds, and $\mathcal{R}$ is an accessibility relation on $\mathcal{G}$. This much is familiar from propositional modal logic treatments, and we do not elaborate on it. As usual, different restrictions on $\mathcal{R}$ give rise to different modal logics.

Domains of (ground level) objects are introduced into a modal model, just as in a classical one. There are two different ways of doing this. Each possible world in $\mathcal{G}$ can have its own domain, in which case we take $\mathcal{D}$ to be a *domain function*, mapping worlds to non-empty sets. Or, all possible worlds can have the same domain, in which case we take $\mathcal{D}$ to be just a set, the common domain for all worlds. In [5] and [3] reasons are presented as to why either version can be taken as basic in the first-order case—essentially each can simulate the other. In the interests of simplicity we adopt the constant domain version in the higher-order setting. Philosophically, this amounts to a possibilist approach to quantification, rather than an actualist one.

Formally, we take $\mathcal{D}$ to be a single non-empty set, called the *domain* of the model $\mathcal{M}$.

Definition 7 (Relation Types). *Let S be a non-empty set. For each type t we define the collection $[\![t, S]\!]$ of* relations of type t over S*.*

1. $[\![0, S]\!] = S$.
2. $[\![\langle t_1, \ldots, t_n\rangle, S]\!]$ *is the collection of all subsets of* $[\![t_1, S]\!] \times \cdots \times [\![t_n, S]\!]$.

We say O is an object of type t over S *if* $O \in [\![t, S]\!]$.

At last we can characterize $\mathcal{I}$, the *interpretation* of the model. Note that it is world-dependent.

Definition 8 (Interpretation). *$\mathcal{I}$ is a mapping from constant symbols and worlds meeting the following conditions. For each world $\Gamma \in \mathcal{G}$:*

1. *If A^t is a constant symbol of type t, $\mathcal{I}(A^t, \Gamma) \in [\![t, \mathcal{D}]\!]$.*
2. *If $=^{\langle t,t\rangle}$ is an equality constant symbol, $\mathcal{I}(=^{\langle t,t\rangle}, \Gamma)$ is the equality relation on $[\![t, \mathcal{D}]\!]$.*

This completes the specification for each component of $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I}\rangle$. If all the conditions given above are met, we say $\mathcal{M}$ is a *higher-order modal model*.

4 Truth

Assume $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I} \rangle$ is a higher-order modal model. We give meaning to $\mathcal{M}, \Gamma \Vdash_v \Phi$, which is read: the formula Φ is true at the world Γ of the model $\mathcal{M}$, with respect to the valuation v which assigns meanings to free variables. To do this we have to assign denotations to terms in general—the denotation of a term of type t will be an object of type t over $\mathcal{D}$. And this can not be done independently. The assignment of denotations to terms, and the determination of formula truth at worlds constitutes a mutually recursive pair of definitions, as was the case for the syntactic notions of term and formula in Section 2.

Definition 9 (Valuation). *We say v is a* valuation *in model $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I} \rangle$ if v assigns to each variable α^t of type t some member of $[\![t, \mathcal{D}]\!]$, that is, $v(\alpha^t) \in [\![t, \mathcal{D}]\!]$.*

Note that, unlike interpretations, valuations are not world dependent.

Definition 10 (Variant). *We say a valuation w is an* α-variant *of a valuation v if v and w agree on all variables except possibly α. More generally, we say w is an* $\alpha_1, \ldots, \alpha_n$-variant *if v and w agree on all variables except possibly $\alpha_1, \ldots, \alpha_n$.*

Definition 11 (Denotation of a Term). *Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I} \rangle$ be a higher-order modal model, and let v be a valuation in it. We define a mapping $(v * \mathcal{I})$, assigning to each term and each world a denotation for that term, at that world.*

1. *If A is a constant symbol then $(v * \mathcal{I})(A, \Gamma) = \mathcal{I}(A, \Gamma)$.*
2. *If α is a variable then $(v * \mathcal{I})(\alpha, \Gamma) = v(\alpha)$.*
3. *If $\langle \lambda \alpha_1, \ldots, \alpha_n . \Phi \rangle$ is a predicate abstract of type t, then $(v * \mathcal{I})(\langle \lambda \alpha_1, \ldots, \alpha_n . \Phi \rangle, \Gamma)$ is the following member of $[\![t, \mathcal{D}]\!]$:*

$$\{\langle w(\alpha_1), \ldots, w(\alpha_n) \rangle \mid w \text{ is an } \alpha_1, \ldots, \alpha_n \text{ variant of } v \text{ and } \mathcal{M}, \Gamma \Vdash_w \Phi\}$$

Definition 12 (Truth of a Formula). *Again let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I} \rangle$ be a higher-order modal model, and let v be a valuation in it. The notion $\mathcal{M}, \Gamma \Vdash_v \Phi$, is characterized as follows.*

1. *For an atomic formula, $\mathcal{M}, \Gamma \Vdash_v \tau(\tau_1, \ldots, \tau_n)$ provided $\langle (v * \mathcal{I})(\tau_1, \Gamma), \ldots, (v * \mathcal{I})(\tau_n, \Gamma) \rangle \in (v * \mathcal{I})(\tau, \Gamma)$.*
2. *$\mathcal{M}, \Gamma \Vdash_v \neg \Phi$ if it is not the case that $\mathcal{M}, \Gamma \Vdash_v \Phi$.*
3. *$\mathcal{M}, \Gamma \Vdash_v \Phi \wedge \Psi$ if $\mathcal{M}, \Gamma \Vdash_v \Phi$ and $\mathcal{M}, \Gamma \Vdash_v \Psi$.*
4. *$\mathcal{M}, \Gamma \Vdash_v \Box \Phi$ if $\mathcal{M}, \Delta \Vdash_v \Phi$ for all $\Delta \in \mathcal{G}$ such that $\Gamma \mathcal{R} \Delta$.*
5. *$\mathcal{M}, \Gamma \Vdash_v (\forall \alpha) \Phi$ if $\mathcal{M}, \Gamma \Vdash_{v'} \Phi$ for every α-variant v' of v.*

Here are a few examples on which you can test your understanding of the definitions above. We are assuming our models are constant domain, so not surprisingly, the Barcan formula is valid. But one must be careful. If Φ is a formula, the following is certainly valid.

$$\Diamond (\exists x) \Phi \supset (\exists x) \Diamond \Phi.$$

But, the following formula is *not* valid, even though it has a Barcan-like quantifier/modality permutation.

$$\Diamond(\exists x)\langle\lambda X.X(x)\rangle(P) \supset (\exists x)\langle\lambda X.\Diamond X(x)\rangle(P).$$

The shift of variable binding for X changes things; in the antecedent it is narrow scope, but in the consequent it is not.

As another slightly surprising example, the following formula is valid.

$$\langle\lambda X.\Diamond(\exists x)X(x)\rangle(P) \supset \langle\lambda X.(\exists x)X(x)\rangle(P) \tag{9}$$

In this example the symbol P is given broad scope in both the antecedent and the consequent of the implication. This essentially says its meaning in alternative worlds will be the same as in the present world. Under these circumstances, existence of something falling under P in an alternate world is equivalent to existence of something falling under P in the present world. (Don't forget, we are assuming constant domains.) This is just a formal variation on the old observation that, in conventional first-order Kripke models, if relation symbols could not vary their interpretation from world to world, modal operators would have no effect. It is also something that can't be said without the use of abstraction notation.

5 Non-standard Models

Just as in the classical case, there can be no proof procedure that is complete with respect to the semantics presented in the previous two sections. And just as in the classical case, one can introduce a modal version of *Henkin models*. Essentially, at each type level of a model we take *some* of the relations available in principle, but not necessarily all of them. We do not have the space here to give details, but they are direct analogs of the classical version.

The important thing to note, for our purposes, is that the most natural higher-order modal tableau rules do *not* give completeness with respect to modal Henkin models. Instead we need a broader notion of model yet—*non-extensional* Henkin models. These can be characterized, and are natural things to study, though knowledge of them is not widespread even though Henkin himself mentioned them. After all, it seems reasonable to have a notion of model in which the properties of being the morning star and being the evening star are different even though they have the same extension.

Space does not permit a formulation of modal higher-order non-extensional Henkin models here. But when formulated, tableau rules given below turn out to be complete with respect to them. Then extensionality can be imposed by adopting extensionality axioms, in the usual way. The completeness proof has considerable complexity, but ultimately is based on constructions of [6, 8].

6 Tableaus

We present a version of *prefixed* tableaus, which incorporate a kind of naming mechanism for possible worlds in such a way that syntactic features of prefixes—world names—reflect semantic features of models, or of candidates for them. Prefixed tableau systems exist for most standard modal logics. Here we only give a version for **S5**, *without equality and without extensionality*. We refer you to the literature for modifications appropriate for other modal logics—the same modifications that work at the propositional level work in our setting too.

Definition 13 (Prefix). *An* **S5** prefix *is a single positive integer.*

Prefixes have two uses in tableau proofs. The first gives them their name.

Definition 14 (Prefixed Formula). *A* prefixed formula *is an expression of the form $\sigma\,\Phi$, where σ is a prefix and Φ is a formula.*

Think of a prefix as a name for a possible world of some model. And think of $\sigma\,\Phi$ as saying that formula Φ is true at the world that σ names.

All tableau proofs are proofs of *sentences*—closed formulas. A tableau proof of Φ is a tree that has $1\,\neg\Phi$ at its root, is constructed according to certain *branch extension* rules to be given below, and is *closed*, which essentially means it embodies an obvious syntactic contradiction. This intuitively says $\neg\Phi$ cannot happen at an arbitrary world, and so Φ is valid.

The branch extension rules for the propositional connectives are all straightforward. We give them here, including rules for various defined connectives, for convenience. In these, and throughout, we use σ, σ', σ_1, and the like as standing for prefixes.

Definition 15 (Conjunctive Rules). *For any prefix σ,*

$$\frac{\sigma\,X \wedge Y}{\begin{array}{c}\sigma\,X\\ \sigma\,Y\end{array}} \qquad \frac{\sigma\,\neg(X \vee Y)}{\begin{array}{c}\sigma\,\neg X\\ \sigma\,\neg Y\end{array}} \qquad \frac{\sigma\,\neg(X \supset Y)}{\begin{array}{c}\sigma\,X\\ \sigma\,\neg Y\end{array}} \qquad \frac{\sigma\,X \equiv Y}{\begin{array}{c}\sigma\,X \supset Y\\ \sigma\,Y \supset X\end{array}}$$

For the conjunctive rules, if the prefixed formula above the line appears on a branch of a tableau, the items below the line may be added to the end of the branch. The rule for double negation is of the same nature, except that only a single added item is involved.

Definition 16 (Double Negation Rule). *For any prefix σ,*

$$\frac{\sigma\,\neg\neg X}{\sigma\,X}$$

Next we have the disjunctive rules. For these, if the prefixed formula above the line appears on a tableau branch, the end node can have two children added, labeled with the two items shown below the line in the rule. In this case we say there is tableau branching.

Definition 17 (Disjunctive Rules). *For any prefix σ,*

$$\frac{\sigma\, X \vee Y}{\sigma\, X \mid \sigma\, Y} \qquad \frac{\sigma\, \neg(X \wedge Y)}{\sigma\, \neg X \mid \sigma\, \neg Y}$$

$$\frac{\sigma\, X \supset Y}{\sigma\, \neg X \mid \sigma\, Y} \qquad \frac{\sigma\, \neg(X \equiv Y)}{\sigma\, \neg(X \supset Y) \mid \sigma\, \neg(Y \supset X)}$$

Next we give the modal rules. It is here that the structure of prefixes plays a role. For **S5**, each world is accessible from each world.

Definition 18 (Possibility Rules). *If the positive integer n is new to the branch,*

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

This implicitly treats $\Diamond$ as a kind of existential quantifier. Correspondingly, the following rules treat $\Box$ as a version of the universal quantifier.

Definition 19 (Necessity Rules). *For any positive integer n,*

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

Many examples of the application of these propositional rules can be found in [3]. We do not give any here.

Next, for quantifiers. For the existential quantifier we do the usual thing: if an existentially quantified formula is true (at some world), we introduce a new name into the language and say in effect, let that be the thing of which the formula is true. For this it is convenient to enhance the collection of free variables available. We add a second kind, called parameters.

Definition 20 (Parameters). *We have assumed that for each type t we had an infinite collection of free variables of that type. We now assume we also have a second, disjoint, list of free variables of type t, called* parameters*. They may appear in formulas in the same way as the original list of free variables* but we never quantify them. *Also* we never λ bind them. *We use letters like p, q, P, Q, ... to represent parameters.*

Technically, parameters are free variables. When interpreting a formula with parameters in a model, a valuation must provide values for parameters as well as for the standard free variables. But since parameters are never quantified or used in λ bindings, any occurrence of a parameter must be a free occurrence. (Consequently they cannot appear in sentences.) We will never need to substitute a term for a parameter, though we will need to substitute terms for free occurrences of variables that are not parameters. For this, and other reasons, we adopt the following convention.

Definition 21 (Variable Convention). *Occurrences of parameters in a formula are not counted as free occurrences. Further, if we refer to a* variable*, it is assumed it is not a parameter. If we need to speak about a parameter, we will explicitly say so.*

To state the existential tableau rules, we use the following convention. Suppose $\Phi(\alpha^t)$ is a formula in which the variable α^t, of type t, may have free occurrences. And suppose p^t is a parameter of type t. Then $\Phi(p^t)$ is the result of replacing all free occurrences of α^t with occurrences of p^t. Since our convention is that parameters are never bound, we don't have to worry about accidental variable capture. Now, here are the existential quantifier rules.

Definition 22 (Existential Rules). *In the following, p^t is a parameter of type t that is* new to the tableau branch*.*

$$\frac{\sigma\,(\exists\alpha^t)\Phi(\alpha^t)}{\sigma\,\Phi(p^t)} \qquad \frac{\sigma\,\neg(\forall\alpha^t)\Phi(\alpha^t)}{\sigma\,\neg\Phi(p^t)}$$

The rules above embody the familiar notion of existential instantiation. As noted, the use of parameters instead of conventional variables avoids complications due to conflicts between free and bound occurrences.

We said prefixes had two roles. We have seen one: formulas are prefixed. The other use of prefixes is to qualify *terms*. Loosely, think of a term τ with σ as prefix as representing the value taken on by the term τ at the world designated by σ. However, writing prefixes in front of terms makes formulas even more unreadable than they already are. Instead, in an abuse of language, we have chosen to write them as subscripts, τ_σ though, of course, the intention is the same, and we still refer to them as prefixes.

Formally, we broaden the notion of term (and consequently of formula) to allow for prefixes/subscripts. Constant symbols may have prefixes—they are non-rigid and can have different values at different worlds, so a prefix plays a significant role, fixing the world at which its value is determined. Similarly for predicate abstracts. But variables and parameters are thought of as ranging over objects directly, and are not world-dependent. Consequently they are not given prefixes.

Definition 23 (Extended Term). *An* extended term *is like a term except that some subterms have prefixes attached (as subscripts). Prefixes may appear as subscripts on constant symbols and predicate abstracts; they may not appear on variables or parameters. It is allowed that no prefixes occur, in which case we have a term in the conventional sense. The* type *of an extended term is the same as the type of the underlying term, that is, of the expression resulting from dropping all prefixes.*

Extended terms are allowed to occur in the formulas appearing in tableaus. Next we need an analog of the notion of closed term, as used in classical first-order tableaus.

Definition 24 (Grounded). *A parameter is a grounded term. A prefixed constant symbol is a grounded term. A prefixed predicate abstract containing no free variables (parameters are allowed) is a grounded term. Also, if $\tau_0(\tau_1, \ldots, \tau_n)$ is an atomic formula and τ_0, τ_1, $\ldots$, τ_n are grounded terms, we refer to the formula as grounded.*

Example 2. $\langle \lambda x.(\forall X)[\langle \lambda \mathcal{X}.\mathcal{X}(X)\rangle(\mathcal{P}) \supset X(x)]\rangle$ is a predicate abstract, hence a term. Then, $\langle \lambda x.(\forall X)[\langle \lambda \mathcal{X}.\mathcal{X}(X)\rangle_2(\mathcal{P}_1) \supset X(x)]\rangle$ is an extended term. It is not grounded, but $\langle \lambda x.(\forall X)[\langle \lambda \mathcal{X}.\mathcal{X}(X)\rangle_2(\mathcal{P}_1) \supset X(x)]\rangle_3$ is.

The presence of a prefix σ on a subterm is intended to indicate that we are thinking about the object the subterm denotes at the world that σ denotes. Since not all subterms may have been intuitively evaluated at a particular stage of a proof, there might be subterms that have not been prefixed.

Definition 25 (Universal Rules). *In the following, τ^t is any grounded term of type t.*

$$\frac{\sigma\, (\forall \alpha^t)\Phi(\alpha^t)}{\sigma\, \Phi(\tau^t)} \qquad \frac{\sigma\, \neg(\exists \alpha^t)\Phi(\alpha^t)}{\sigma\, \neg\Phi(\tau^t)}$$

Now we give the rules for predicate abstracts and atomic formulas. And to do this, we first define an auxiliary notion. The intuition is that $\tau@\sigma$ plays the role of the object the extended term τ designates at world σ. Note that $\tau@\sigma$ must be grounded.

Definition 26 (Evaluation At a Prefix). *Let σ be a prefix. If τ is an extended term without free variables, $\tau@\sigma$ is defined as follows.*

1. *If τ is a parameter, $\tau@\sigma = \tau$.*
2. *If τ is an unsubscripted constant symbol or predicate abstract, $\tau@\sigma = \tau_\sigma$.*
3. *If τ is a subscripted constant symbol or predicate abstract, $\tau@\sigma = \tau$.*

Also, if $\tau_0(\tau_1, \ldots, \tau_n)$ is atomic, where each τ_i is an extended term without free variables, we set

$$[\tau_0(\tau_1, \ldots, \tau_n)]@\sigma = [\tau_0@\sigma(\tau_1@\sigma, \ldots, \tau_n@\sigma)]$$

The next rule says that determining the truth of an atomic formula at a world requires we evaluate its constituents at that world.

Definition 27 (Atomic Evaluation Rules). *Let X be an atomic formula.*

$$\frac{\sigma\, X}{\sigma\, X@\sigma} \qquad \frac{\sigma\, \neg X}{\sigma\, \neg X@\sigma}$$

If a term is grounded, its meaning is fixed across worlds. If I say "the President of the United States," it means different people at different times, but if I say "the President of the United States in 1812," it designates the same person at all times. This motivates the following rule.

Definition 28 (World Shift Rules). *Let X be a* grounded *atomic formula.*

$$\frac{\sigma\, X}{\sigma'\, X} \qquad \frac{\sigma\, \neg X}{\sigma'\, \neg X}$$

Finally, a rule intended to capture the meaning of predicate abstracts. Note the respective roles of σ and σ'. Also, we extend earlier notation so that, if $\Phi(\alpha_1, \ldots, \alpha_n)$ is a formula, α_1, ..., α_n are free variables, and τ_1, ..., τ_n are extended terms of the same respective types as α_1, ..., α_n, then $\Phi(\tau_1, \ldots, \tau_n)$ is the result of simultaneously substituting each τ_i for all free occurrences of α_i in Φ.

Definition 29 (Predicate Abstract Rules). *In the following, $\tau_1, \ldots, \tau_n$ are all* grounded *terms*

$$\frac{\sigma'\, \langle \lambda\alpha_1, \ldots, \alpha_n.\Phi(\alpha_1, \ldots, \alpha_n)\rangle_\sigma(\tau_1, \ldots, \tau_n)}{\sigma\, \Phi(\tau_1, \ldots, \tau_n)}$$

$$\frac{\sigma'\, \neg\langle \lambda\alpha_1, \ldots, \alpha_n.\Phi(\alpha_1, \ldots, \alpha_n)\rangle_\sigma(\tau_1, \ldots, \tau_n)}{\sigma\, \neg\Phi(\tau_1, \ldots, \tau_n)}$$

Finally what, exactly, constitutes a proof.

Definition 30 (Closure). *A tableau branch is* closed *if it contains $\sigma\, \Psi$ and $\sigma\, \neg\Psi$, for some formula Ψ.*

Definition 31 (Tableau Proof). *For a sentence Φ, a closed tableau beginning with $1\, \neg\Phi$ is a* proof *of Φ.*

This concludes the presentation of the basic tableau rules. We have not given rules for equality or extensionality. In fact, extensionality is not provable without further rules. In the next section we give a few examples of tableau proofs using the rules above.

7 Tableau Examples

Tableaus for classical logic are well-known, and even for propositional modal logics they are rather familiar. The abstraction rules of the previous section are new, and we give two examples illustrating their uses, one easy, one harder.

Example 3. Here is a proof of (9), $\langle\lambda X.\Diamond(\exists x)X(x)\rangle(P) \supset \langle\lambda X.(\exists x)X(x)\rangle(P)$, which we earlier noted was valid.

$$
\begin{array}{lll}
1 & \neg[\langle\lambda X.\Diamond(\exists x)X(x)\rangle(P) \supset \langle\lambda X.(\exists x)X(x)\rangle(P)] & 1. \\
1 & \langle\lambda X.\Diamond(\exists x)X(x)\rangle(P) & 2. \\
1 & \neg\langle\lambda X.(\exists x)X(x)\rangle(P) & 3. \\
1 & \langle\lambda X.\Diamond(\exists x)X(x)\rangle_1(P_1) & 4. \\
1 & \neg\langle\lambda X.(\exists x)X(x)\rangle_1(P_1) & 5. \\
1 & \Diamond(\exists x)P_1(x) & 6. \\
1 & \neg(\exists x)P_1(x) & 7. \\
2 & (\exists x)P_1(x) & 8. \\
2 & P_1(p) & 9. \\
1 & \neg P_1(p) & 10. \\
1 & P_1(p) & 11.
\end{array}
$$

In this, 2 and 3 are from 1 by a conjunctive rule; 4 is from 2 and 5 is from 3 by atomic evaluation; 6 and 7 are from 4 and 5 respectively by predicate abstract rules; 8 is from 6 by a possibility rule; 9 is from 8 by an existential rule (p is a new parameter); 10 is from 7 by a universal rule; and 11 is from 9 by a world shift rule.

Example 4. Our next example is from Gödel's ontological argument for the existence of God [4]. We don't need all the details—think of it simply as a technical issue. Essentially, Gödel modified an earlier argument due to Leibniz. Part of what Gödel did was replace a somewhat intuitive notion of *perfection* with the notion of a *positive* property. This notion was not analyzed, but certain features were assumed for it—axioms, in effect.

Positive Property We have a constant symbol $\mathcal{P}$ of type $\langle\langle 0\rangle\rangle$ (think of it as "positiveness"). We assume

$$(\forall X)[\neg\mathcal{P}(X) \supset \mathcal{P}(\neg X)]$$

where, for X of type $\langle 0\rangle$, $\neg X$ abbreviates $\langle\lambda x.\neg X(x)\rangle$.

Next, Gödel takes *being God* to mean having all positive properties.

Being God We use G to abbreviate the type $\langle 0\rangle$ term

$$\langle\lambda x.(\forall X)[\mathcal{P}(X) \supset X(x)]\rangle.$$

Finally as far as our example is concerned, Gödel characterizes a notion of *essence*. Roughly speaking, a property X is the essence of an object x if every property of x is a necessary consequence of X.

Essence We use $\mathcal{E}$ to abbreviate the type $\langle\langle 0\rangle, 0\rangle$ term

$$\langle\lambda X, x.(\forall Y)[Y(x) \supset \Box(\forall y)[X(y) \supset Y(y)]]\rangle$$

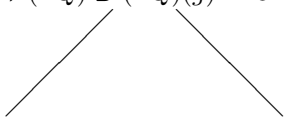
Now we show one step of Gödel's argument: $(\forall x)[G(x) \supset \mathcal{E}(G,x)]$. That is, being God is the essence of anything that is, in fact, God. We give a tableau derivation of it from the assumption about positive properties.

$1\ \neg(\forall x)[G(x) \supset \mathcal{E}(G,x)]$ 1.
$1\ \neg[G(g) \supset \mathcal{E}(G,g)]$ 2.
$1\ \ G(g)$ 3.
$1\ \neg\mathcal{E}(G,g)$ 4.
$1\ \ G_1(g)$ 5.
$1\ \neg\langle\lambda X,x.(\forall Y)[Y(x) \supset \Box(\forall y)[X(y) \supset Y(y)]]\rangle(G,g)$ 6.
$1\ \neg\langle\lambda X,x.(\forall Y)[Y(x) \supset \Box(\forall y)[X(y) \supset Y(y)]]\rangle_1(G_1,g)$ 7.
$1\ \neg(\forall Y)[Y(g) \supset \Box(\forall y)[G_1(y) \supset Y(y)]]$ 8.
$1\ \neg[Q(g) \supset \Box(\forall y)[G_1(y) \supset Q(y)]]$ 9.
$1\ \ Q(g)$ 10.
$1\ \neg\Box(\forall y)[G_1(y) \supset Q(y)]$ 11.
$1\ \ (\forall X)[\neg\mathcal{P}(X) \supset \mathcal{P}(\neg X)]$ 12.
$1\ \neg\mathcal{P}(Q) \supset \mathcal{P}(\neg Q)$ 13.

In this, 2 is from 1 by an existential rule (g is a new parameter); 3 and 4 are from 2 by a conjunctive rule; 5 is from 3 by an atomic evaluation rule; 6 is 4 unabbreviated; 7 is from 6 by an atomic evaluation rule; 8 is from 7 by a predicate abstract rule; 9 is from 8 by an existential rule (Q is a new parameter); 10 and 11 are from 9 by a conjunctive rule; 12 is our assumption about positiveness; 13 is from 12 by a universal rule.

At this point the tableau branches, using item 13. We first present the *right* branch, then the left.

$1\ \mathcal{P}(\neg Q)$ 14.
$1\ \langle\lambda x.(\forall X)[\mathcal{P}(X) \supset X(x)]\rangle(g)$ 15.
$1\ \langle\lambda x.(\forall X)[\mathcal{P}(X) \supset X(x)]\rangle_1(g)$ 16.
$1\ (\forall X)[\mathcal{P}(X) \supset X(g)]$ 17.
$1\ \mathcal{P}(\neg Q) \supset (\neg Q)(g)$ 18.



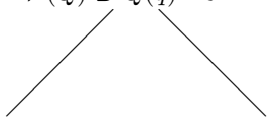
$1\ \neg\mathcal{P}(\neg Q)$ 19.

$1\ \ (\neg Q)(g)$ 20.
$1\ \ \langle\lambda x.\neg Q(x)\rangle(g)$ 21.
$1\ \ \langle\lambda x.\neg Q(x)\rangle_1(g)$ 22.
$1\ \neg Q(g)$ 23.

Item 14 is from 13 by a disjunctive rule; 15 is 3 unabbreviated; 16 is from 15 by an atomic evaluation rule; 17 is from 16 by a predicate abstract rule; 18 is from 17 using a universal rule; 19 and 20 are from 18 by a disjunctive rule; 21 is 20 unabbreviated; 22 is from 21 an atomic evaluation rule; 23 is from 22 by a predicate abstract rule. Closure is by 14 and 19, and 10 and 23.

Now we display the *left* branch.

$1\,\neg\neg\mathcal{P}(Q)$ 24.
$1\;\; \mathcal{P}(Q)$ 25.
$2\,\neg(\forall y)[G_1(y) \supset Q(y)]$ 26.
$2\,\neg[G_1(q) \supset Q(q)]$ 27.
$2\;\; G_1(q)$ 28.
$2\,\neg Q(q)$ 29.
$2\;\; \langle\lambda x.(\forall X)[\mathcal{P}(X) \supset X(x)]\rangle_1(q)$ 30.
$1\;\; (\forall X)[\mathcal{P}(X) \supset X(q)]$ 31.
$1\;\; \mathcal{P}(Q) \supset Q(q)$ 32.



$1\,\neg\mathcal{P}(Q)$ 33. $1\,Q(q)$ 34.

Item 24 is from 13 by a disjunctive rule; 25 is from 24 by double negation; 26 is from 11 by a possibility rule; 27 is from 26 by an existential rule (q is a new parameter); 28 and 29 are from 27 by a conjunctive rule; 30 is 28 unabbreviated; 31 is from 30 by a predicate abstract rule; 32 is from 31 by a universal rule; 33 and 34 are from 32 by a disjunctive rule. Closure is by 25 and 33, and 29 and 34.

8 Conclusion

Higher-order modal logic is inherently complex. Just a sketch was possible here. There was no room to present Henkin-modal models, let alone non-extensional versions, though they are extremely natural to work with. Tableau completeness arguments are especially elaborate. A much longer treatment is in preparation. We hope this brief sketch is enough to raise interest in an issue that has rarely been looked at in modal logic ([1] is a rare but noteworthy instance).

References

1. A. Bressan. *A General Interpreted Modal Calculus*. Yale University Press, 1972.
2. M. C. Fitting. Bertrand Russell, Herbrand's theorem, and the assignment statement. In J. Calmet and J. Plaza, editors, *Artificial Intelligence and Symbolic Computation*, pages 14–28. Springer Lecture Notes in Artificial Intelligence, 1476, 1998.
3. M. C. Fitting and R. Mendelsohn. *First-Order Modal Logic*. Kluwer, 1998.
4. K. Gödel. Ontological proof. In S. Feferman, J. W. Dawson, Jr., W. Goldfarb, C. Parsons, and R. M. Solovay, editors, *Kurt Gödel Collected Works*, volume III, pages 403–404. Oxford, Oxford, 1995.
5. G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, London, 1996.
6. D. Prawitz. Hauptsatz for higher order logic. *Journal of Symbolic Logic*, 33:452–457, 1968.

7. R. Stalnaker and R. Thomason. Abstraction in first-order modal logic. *Theoria*, 34:203–207, 1968.
8. M. Takahashi. A proof of cut-elimination theorem in simple type theory. *J. Math. Soc. Japan*, 19:399–410, 1967.
9. R. Thomason and R. Stalnaker. Modality and reference. *Nous*, 2:359–372, 1968.