Modal logic should say more than it does

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 [*]

February 18, 1991

Abstract

First-order modal logics, as traditionally formulated, are not expressive enough. It is this that is behind the difficulties in formulating a good analog of Herbrand's Theorem, as well as the well-known problems with equality, non-rigid designators, definite descriptions, and non-designating terms. We show how all these problems disappear when modal language is made more expressive in a simple, natural way. We present a semantic tableaux system for the enhanced logic, and (very) briefly discuss implementation issues.

1 Introduction

Necessity is the mother of modality, but she has a large family. It includes *knows*, *believes*, *legally requires* and *morally requires*, among others. Clearly a proper understanding of the behavior of modal operators is essential to any formal treatment of natural language, as well as to any formalization of everyday reasoning. Unfortunately, once the elementary level has been passed, virtually every feature that has made classical logic a powerful tool has led to confusion and argument in the modal framework. In this paper we propose a simple way out of these difficulties. The constructions considered here increase the expressive powers of modal language, make it possible to avoid many of the difficulties usually associated with first-order modal logics, address some interesting philosophical problems, and have a fundamental bearing on non-classical automated theorem proving. Believe it or not (a modal sentence if I ever heard one.)

In order to give a feeling for some of the problems present in first-order modal logics as usually formulated, it is convenient to make use of Kripke's possible world semantics for modal logic. We give a formal treatment below, but only an informal understanding is required now. A *model* has a

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

number of *possible worlds*; some are *accessible* from others; at each world the classical connectives and quantifiers behave in the expected classical ways; $\Box X$ (necessarily X) is true at a possible world if X is true at every world accessible from it. This is enough model detail for now.

In automated theorem proving in classical logic, a formula like $(\exists x)\Phi(x)$ is often Skolemized. One introduces a constant symbol, say c, intended to name something having the property $\Phi(x)$ if anything does. Then one replaces $(\exists x)\Phi(x)$ with $\Phi(c)$, thus eliminating a quantifier. Now, in a modal setting, how should we Skolemize $\Box(\exists x)\Phi(x)$? If it is true at a possible world, $(\exists x)\Phi(x)$ is true at each accessible world, and hence at each accessible world $\Phi(x)$ will be true of some object. But that object might be different at different worlds. If we Skolemize in the expected way, producing $\Box\Phi(c)$, the constant symbol c must be allowed to name different objects in different worlds. Philosophers call such things *non-rigid designators*. The notion of Skolemization appropriate for modal logic, then, requires the use of non-rigid designators.

As it happens, almost every treatment of first-order modal logic in the literature does not permit non-rigid designators. In [12] the usual syntax of terms is modified to allow them to be non-rigid, while in [20] and [21] the syntax of formulas is modified (the approach continued here). But essentially everywhere else, constant symbols are required to name the same objects from world to world. (Such constant symbols are called *rigid designators*.) One basic reason for this is astonishingly simple: if one allows non-rigid designators, standard modal notation becomes ambiguous. Consider again the formula $\Box\Phi(c)$; what could it mean for it to be true at the possible world Γ? One reasonable reading is: $\Box\Phi(c)$ is true at Γ provided $\Phi(c)$ is true at every alternative world Δ, taking c to name whatever it did in Γ. Another equally reasonable reading is: $\Box\Phi(c)$ is true at Γ provided $\Phi(c)$ is true at every alternative world Δ, taking c to name whatever it does in Δ. But c might name different objects in Γ and Δ. In short, there are two basic actions: letting c designate, and moving to an alternative world. These two actions commute only if c is a rigid designator. Ordinary first-order modal syntax has no machinery to distinguish the two alternate readings of $\Box\Phi(c)$. Consequently when non-rigid designators have been treated at all, one of the readings has been disallowed, thus curtailling expressive power.

If we Skolemize the two modal sentences $(\exists x)\Box\Phi(x)$ and $\Box(\exists x)\Phi(x)$, we seem to get the same sentence, $\Box\Phi(c)$, in both cases. In fact, we need to be able to read $\Box\Phi(c)$ one way for it to replace $(\exists x)\Box\Phi(x)$, and another way for it to replace $\Box(\exists x)\Phi(x)$; we need both the readings considered above, and standard modal syntax does not allow us to distinguish them. The classic paper [17] has taught us the importance of Herbrand's Theorem for automated theorem proving. In the modal case a really satisfactory analog of Herbrand's Theorem is tricky (see [1] and [12] for versions). The difficulty lies in the lack of expressive power of conventional presentations of first-order modal logics. Indeed, as we have seen, even Skolemization is a problem.

The issues raised above affect representation of everyday expressions in formal terms. For example, consider the sentence "The President of the United States someday won't be the President of the United States." In interpreting this we take the designator 'The President of the United States' as naming a person in the present world, then we pass to a future alternative world in which that person is no longer president. On the other hand, consider the sentence "The President of the United States might not have been George Bush." Here we mean that there is a logically possible world alternative to this one in which 'The President of the United States' names somebody other than George Bush. In the first example, 'The President of the United States' designates before a move is made to an alternative world; in the second example the designation action occurs after the move to an alternative world is made. These two examples show that each way of reading something like $\Box\Phi(c)$ is sometimes appropriate. In informal settings we rely on context to distinguish which version we mean, but a formal language should make the distinction explicit.

The observation that non-rigid designators are needed for automated modal theorem proving has been made by others. Konolige ([9], [12]) introduces a 'bullet' operator as a device for coping with the ambiguity problems that arise. While useful, we feel it does not go far enough in addressing the fundamental problems of limited expressibility inherent in modal logic. The mechanism presented here is exactly dual to that of Konolige. Where Konolige modifies the structure of *terms*, we do not, but instead modify the general notion of *formulas*. The two approaches are compared briefly in Section 4.

The solution to the expressibility problem that is presented here involves a device of *abstraction*. It is far from new, though it seems to be generally unknown in the theorem proving and artificial intelligence communities. It was introduced by Stalnaker and Thomason in [20] and [21]. Based on their work, modal versions of Hilbert's *epsilon calculus* were investigated in [4] and [6]. Even an analog of Herbrand's Theorem was developed in [5]. Then, curiously enough, nothing further on the subject seems to have appeared in the literature, though in a general way the basic ideas were subsumed in those of Montague semantics. Probably those portions of theoretical computer science for which such things are relevant had not sufficiently matured. At any rate, we believe it is time for a revival. We present the basic ideas, as originally introduced in [20] and [21]. To these are added tableaux methods which allow for automation, though we do not persue the issue here. We then extend the basic ideas to deal with the problems of definite descriptions in modal contexts, and of non-designating terms. Semantic tableaux methods are also generalized to treat these issues as well. We hope others will take up the challange of providing efficient implementations incorporating the devices presented here, so that applications of a truly versatile non-classical logic can be developed.

2 Predicate Abstraction

In the Lambda calculus a distinction is made between a *term*, say $x+1$, and the *function* formed by abstraction applied to this term, $(\lambda x.x+1)$. Over the years this distinction has become a familiar one to most of us. What we need here is a similar distinction between a *formula* of logic, say $\Phi(x)$, and what we might call the *predicate abstraction* formed by applying lambda abstraction to it, $(\lambda x.\Phi(x))$. This notion has not arisen in classical first-order logic because it adds nothing there. For instance, we will see that if Φ contains no modal operators, $(\lambda x.\Phi)$ and Φ behave essentially alike. But there is a significant difference between $\Box(\lambda x.\Phi(x))(c)$ and $(\lambda x.\Box\Phi(x))(c)$. In the first formula, $\Box(\lambda x.\Phi(x))(c)$, we are asserting that a certain proposition is necessary, the proposition that c has property Φ. Thinking of λ as a scoping device, c is within the scope of $\Box$ in this formula, and thus its designation can be expected to change from world to world. The second formula, $(\lambda x.\Box\Phi(x))(c)$, asserts that c has a certain property, the "necessarily-Φ" property. In this, c is not within the scope of $\Box$, so its designation can be expected to remain fixed from world to world.

A formula like $\Box(\lambda x.\Phi(x))(c)$ in which the necessity operator is applied to a proposition is sometimes said to express a *de dicto* modality (from *dictum*, proposition). Likewise a formula like $(\lambda x.\Box\Phi(x))(c)$ in which the object designated by c is asserted to have a necessary property is sometimes said to express a *de re* modality (from *res*, thing). A discussion of the *de dicto/de re* distinction can be found in [10], Chapter 10. It will play no special role here.

To make things more concrete, consider the following simple example. Is it necessary that the number of planets be odd? The number of planets is (we believe) 9, and 9 is necessarily odd. There is no alternative state of affairs in which 9 could be even. Then one may reasonably say it is necessary that the number of planets is odd. On the other hand, the number of planets might not have turned out to be 9. One can imagine an alternative universe in which there are 6, or even 0 planets. Then it does not seem necessary that the number of planets be odd. Clearly we are interpreting the natural language assertion "it is necessary that the number of planets be odd" in two different ways here. Predicate abstraction was introduced by Stalnaker and Thomason in order to make such distinctions explicit. Suppose p is a constant symbol, designating the number of the planets. Then p is non-rigid; given the universe as we know it, p designates the number 9, but one can imagine other situations in which the number of planets is a different number. Let $O(x)$ be a formula expressing that x is an odd number. Then the formula $(\lambda x.\Box O(x))(p)$ asserts of the number designated by p, namely 9, that it is necessarily odd. This is correct. On the other hand, $\Box(\lambda x.O(x))(p)$ asserts the necessity of the proposition that the number of planets be odd. That is, it asserts that in every alternative universe the number of planets will be odd. This is not correct. We see that the same natural language sentence has two possible formalizations, one true, one false.

3 Syntax

We take $\wedge$, $\neg$, $\forall$ and $\Box$ as primitive and define the other connectives, quantifiers and modalities as usual. Also *terms* are built up from an alphabet of variables and constant symbols using function symbols in the usual way; we omit details.

An *atomic formula* is an expression of the form $P(x_1, \ldots, x_n)$ where P is an n-place relation symbol and $x_1,\ldots,x_n$ are *variables.* The definition of formula, and free variable, is the usual one of first-order modal logic, with the addition of the following:

- if Φ is a formula, x is a variable and t is a term, then $(\lambda x.\Phi)(t)$ is a formula. The free variable occurrences of $(\lambda x.\Phi)(t)$ are those of Φ except for occurrences of x, together with the free variable occurrences of t.

Notice we did not allow any terms except variables to occur in atomic formulas. Where in a standard formulation of logic we might write $P(c)$, here we will write $(\lambda x.P(x))(c)$. The idea is that predicate abstractions are applied to terms to create meaningful formulas. At the atomic level, the variable x in $P(x)$ is essentially a place holder only.

We will abbreviate $(\lambda x.(\lambda y.\Phi)(u))(t)$ by $(\lambda x, y.\Phi)(t, u)$, and similarly for more complicated cases. Likewise we will occasionally write $(\lambda \mathbf{x}.\Phi)(\mathbf{t})$ for $(\lambda x_1, \ldots, x_n.\Phi)(t_1, \ldots, t_n)$ when no confusion is likely.

4 Semantics

There are many different propositional modal logics. Semantically, Kripke models for these are distinguished by putting various special restrictions on the notion of alternativeness between possible worlds; restrictions like transitivity or reflexivity [13]. Such issues are not important for the points we are concerned with, so to keep things simple we have chosen a single underlying propositional modal logic as a foundation throughout. The logic we use is generally called K; it is the weakest normal modal logic. In Kripke models suitable for this logic no special restrictions are placed on the accessibility relation at all. What we present for K carries over with no change to other modal logics as well.

When first-order machinery is added to a propositional modal logic [14], several choices are available that have no classical counterparts. In a Kripke model each possible world has a domain associated with it, for quantifiers to range over at that world. We could require that all worlds have the same associated domains. Or we could weaken this by allowing different worlds to have different associated domains, but still requiring that all the objects in the domain associated with a possible world are still available in the domains of worlds accessible from that one. Or we could simply allow different worlds to have different domains, without restriction. Each choice gives rise to a different first-order modal logic. There is no 'right' choice; the guiding principle must be appropriateness for an intended application. Again to keep things simple, we assume all possible worlds have the same associated domains.

When predicate abstraction is not considered, it is well-known that the semantic condition that all worlds have the same associated domains corresponds to the proof-theoretic condition that the *Barcan formula* be a theorem. The Barcan formula is $(\forall x)\Box\Phi(x) \supset \Box(\forall x)\Phi(x)$ (or rather, any formula of this form is a Barcan formula). This connection between the Barcan formula and constant domain models continues when predicate abstraction is allowed. Still, it is important to repeat that the restriction to constant domain models is only for convenience. There are other versions of our work that apply more generally but they are more complicated, so for reasons of clarity we confine ourselves to the constant domain case here.

Finally, there is the issue of non-designating terms. Such things are common in natural language; 'The King of France' is a well-known example. At the start we will assume we do not have such terms; all terms designate. Later on we will discuss modifications to allow non-designating terms. Now for the formal machinery.

Definition 4.1 A *first-order Kripke frame* is a triple $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ where $\mathcal{G}$ is a non-empty set, $\mathcal{R}$ is a binary relation on $\mathcal{G}$, and $\mathcal{D}$ is a non-empty set.

If $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ is a first-order Kripke frame, the members of $\mathcal{G}$ are generally called *possible worlds*. The relation $\mathcal{R}$ is the *accessibility* relation: if $\Gamma\mathcal{R}\Delta$ then the world Δ is accessible from Γ, or possible relative to Γ. $\mathcal{D}$ is the domain over which quantifiers range; it is the same for all worlds, as frames are defined here. If we wanted something more general, we could take it to be a function from worlds to non-empty sets.

Definition 4.2 A *non-rigid Kripke model* is a quadruple $\langle \mathcal{G}, \mathcal{R}, \mathcal{D}, v \rangle$ where $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ is a first-order Kripke frame and v is a mapping that:

1. assigns to each n-place relation symbol P and each world Γ in $\mathcal{G}$ an n-place relation $v(P,\Gamma)$ on $\mathcal{D}$;
2. assigns to each constant symbol c and each world Γ in $\mathcal{G}$ a member $v(c,\Gamma) \in \mathcal{D}$;
3. assigns to each n-place function symbol f and each world Γ in $\mathcal{G}$ a function $v(f,\Gamma) : \mathcal{D}^n \to \mathcal{D}$.

Informally we may think of a constant symbol c as a name, and $v(c,\Gamma)$ as the object that c names in the world Γ. Condition 2. above assumes that names always designate. Similar remarks apply to function symbols and condition 3. This is not always so in the real world, as we remarked earlier. In more conventional treatments of first-order modal logic, $v(c,\Gamma)$ is not allowed to depend on Γ; constant symbols designate the same objects in all worlds, and so are rigid. Also, function symbols are not commonly considered. We hope to show our more general version is both more expressive, and natural.

Definition 4.3 An *interpretation* s in a model $\langle \mathcal{G},\mathcal{R},\mathcal{D},v\rangle$ is a mapping from the set of free variables of the language to the domain $\mathcal{D}$ of the model. We write $s\left[\frac{a}{x}\right]$ to denote the interpretation that is like s except that it maps the variable x to a.

Interpretations are defined on variables. We extend the notion to terms and worlds as follows.

Definition 4.4 Let s be an interpretation in the non-rigid Kripke model $\langle \mathcal{G},\mathcal{R},\mathcal{D},v\rangle$, and let $\Gamma \in \mathcal{G}$. Then we set:

1. $s(x,\Gamma) = s(x)$, for a variable x;
2. $s(c,\Gamma) = v(c,\Gamma)$, for a constant symbol c;
3. $s(f(t_1,\ldots,t_n),\Gamma) = v(f,\Gamma)(s(t_1,\Gamma),\ldots,s(t_n,\Gamma))$ for a function symbol f.

Now the central notion. We write $\mathcal{M},\Gamma \models \Phi[s]$ to symbolise that the formula Φ is true, under the interpretation s, at the world Γ, in the model $\mathcal{M}$.

Definition 4.5 Let $\mathcal{M} = \langle \mathcal{G},\mathcal{R},\mathcal{D},v\rangle$ be a non-rigid Kripke model, and let s be an interpretation in $\mathcal{M}$. Then:

1. if $P(x_1,\ldots,x_n)$ is atomic, $\mathcal{M},\Gamma \models P(x_1,\ldots,x_n)[s]$ provided $\langle s(x_1),\ldots,s(x_n)\rangle$ is in the relation $v(P,\Gamma)$;
2. $\mathcal{M},\Gamma \models \neg\Phi[s]$ if it is not the case that $\mathcal{M},\Gamma \models \Phi[s]$;
3. $\mathcal{M},\Gamma \models (\Phi \wedge \Psi)[s]$ if $\mathcal{M},\Gamma \models \Phi[s]$ and $\mathcal{M},\Gamma \models \Psi[s]$;
4. $\mathcal{M},\Gamma \models (\forall x)\Phi[s]$ if $\mathcal{M},\Gamma \models \Phi[s\left[\frac{a}{x}\right]]$ for every $a \in \mathcal{D}$;
5. $\mathcal{M},\Gamma \models \Box\Phi[s]$ if $\mathcal{M},\Delta \models \Phi[s]$ for every $\Delta \in \mathcal{G}$ such that $\Gamma\mathcal{R}\Delta$;
6. $\mathcal{M},\Gamma \models (\lambda x.\Phi)(t)[s]$ if $\mathcal{M},\Gamma \models \Phi[s\left[\frac{a}{x}\right]]$ where $a = s(t,\Gamma)$.

We can think of the members of $\mathcal{D}$ as the *objects* of the model. Item 4. says quantifiers range over objects, not over names for them. Item 5. is characteristic of Kripke models. It says necessary truth at a world is equivalent to truth at all worlds that are possible relative to it, a technical version of an idea traceable to Leibnitz. Item 6. is the only one that is not standard in treatments of first-order modal logic. Essentially it says the predicate abstraction $(\lambda x.\Phi)$ is true of a name t at a world provided Φ is true of the object that t names.

If Φ is a closed formula, or *sentence*, it is easy to see that $\mathcal{M}, \Gamma \models \Phi[s]$ for some interpretation s if and only if $\mathcal{M}, \Gamma \models \Phi[s]$ for every interpretation s. For a sentence Φ we will write $\mathcal{M}, \Gamma \models \Phi$ for $\mathcal{M}, \Gamma \models \Phi[s]$ for some (any) interpretation s.

In the introduction we noted there were two possible readings of $\Box\Phi(c)$, depending on when c was allowed to designate. $\Box\Phi(c)$ is not a formula of our language as we have defined it. But corresponding to $\Box\Phi(c)$ we now have two syntactically distinct sentences, $(\lambda x.\Box\Phi(x))(c)$ and $\Box(\lambda x.\Phi(x))(c)$. A straightforward application of the definition above shows that, in a non-rigid Kripke model $\mathcal{M}$,

$\mathcal{M}, \Gamma \models (\lambda x.\Box\Phi(x))(c)[s]$ if and only if for every Δ such that $\Gamma\mathcal{R}\Delta$,
$\mathcal{M}, \Delta \models \Phi(x)[s\left[\begin{smallmatrix}a\\x\end{smallmatrix}\right]]$ where $a = s(c, \Gamma)$;

$\mathcal{M}, \Gamma \models \Box(\lambda x.\Phi(x))(c)[s]$ if and only if for every Δ such that $\Gamma\mathcal{R}\Delta$,
$\mathcal{M}, \Delta \models \Phi(x)[s\left[\begin{smallmatrix}a\\x\end{smallmatrix}\right]]$ where $a = s(c, \Delta)$.

These are exactly the two alternative readings of $\Box\Phi(c)$ we considered informally earlier.

Definition 4.6 A sentence Φ is *valid* provided, for every non-rigid model $\mathcal{M}$ and for every possible world Γ of $\mathcal{M}$ we have $\mathcal{M}, \Gamma \models \Phi$. Φ is *satisfiable* if there is some non-rigid model $\mathcal{M}$ and some possible world Γ of $\mathcal{M}$ such that $\mathcal{M}, \Gamma \models \Phi$.

In [9] an example is given to show that a naively formulated Herbrand's Theorem fails for modal logics. This example is useful here as a device to contrast the approach taken by Konolige in [12] and [9] with ours, and so we spend some time discussing it now. 1. Currently, the mayor of New York City is not Italian (it's Ed Koch). 2. Suppose that if someone isn't Italian, Sam knows it. 3. Still, it may be that Sam doesn't know the mayor of New York isn't Italian (since he may believe that Fiorello LaGuardia is still the mayor). Suppose we let m be a constant symbol intended to name the mayor of New York, and we use $N(x)$ with the intended meaning 'x is not Italian.' Then the numbered sentences can be formalized as follows, using $\Box$ to represent *Sam_knows*.

1. $N(m)$
2. $(\forall x)(N(x) \supset \Box N(x))$
3. $\neg\Box N(m)$

The argument is made in [9] that this set of sentences is satisfiable (it is essential that m be non-rigid for this), but if we add the following substitution instance of item 2., $N(m) \supset \Box N(m)$, the resulting set is not satisfiable. In order to get around this, a 'bullet' operator is introduced, so that for a term t, "$\bullet t$ always refers to whatever t denotes in the actual world, no matter what the context of interpretation." Then the 'correct' substitution instance of item 2. should be written as $N(m) \supset \Box N(\bullet m)$, and now we get a set that turns out to be satisfiable.

Our own solution to the problem is exactly complementary to that of [12]: we see nothing wrong with item 2., but both of the other two items pose problems. Item 1. is not syntactically correct in our system, but this is simple to adjust. We replace it with $(\lambda x.N(x))(m)$. Item 3., however, is the real problem. We could replace it with either 4. $\neg(\lambda x.\Box N(x))(m)$ or 5. $\neg\Box(\lambda x.N(x))(m)$. Distinctions that were blurred together before now must be carefully distinguished. $(\lambda x.\Box N(x))(m)$ can be read, "Sam knows, of the mayor of New York, that he is not Italian." Since the mayor of New York is Ed Koch (currently), and he is not Italian, presumably Sam knows he is not Italian. He can know this without knowing that he is the mayor. On the other hand, $\Box(\lambda x.N(x))(m)$ can be read as, "Sam knows that the mayor of New York is not Italian," which is quite a different assertion.

It is easy to see that at any possible world at which item 2. is true, we also have that $(\lambda x.N(x))(m) \supset (\lambda x.\Box N(x))(m)$ is true. Then it is trivial that sentences 1. (revised as above), 2. and 4. together are inconsistent. On the other hand, it is not hard to show that 1., 2. and 5. are consistent. We believe that the failure to distinguish between the two possible readings of item 3. was at the heart of the problem all along.

5 Elementary Results

We begin with a result that says the addition of predicate abstraction machinery is *conservative*. If a sentence Φ does not contain any occurrences of the lambda operator, then Φ will also be a meaningful sentence in more conventional formulations of modal logic. But, the machinery of our version of Kripke model only differs from more conventional versions in its treatment of predicate abstractions. If Φ does not involve the lambda operator, its truth value at a possible world in one of our models will be calculated the same way it would be in other treatments. Ignoring the special machinery, our models are simply those of the so-called constant domain version of K. Such models are sound and complete with respect to $K+$ the Barcan Formula [10], [11]. Consequently we have the following.

Proposition 5.1 *If Φ is a sentence that contains no predicate abstractions, then Φ is valid in our sense if and only if Φ is a theorem of the first-order modal logic $K+$ the Barcan Formula.*

We may think of $(\lambda x.\Phi(x))$ as the predicate abstracted from $\Phi(x)$. Such a notion does not seem to have arisen in classical logic; there one essentially thinks of $\Phi(x)$ as already being a predicate. The following easily established result explains why the notion of predicate abstraction has not arisen classically. It says abstraction is transparent with respect to the classical machinery.

Proposition 5.2 *The following sentences are valid (where* $\mathbf{c}$ *and* d *are constant symbols):*

1. $(\lambda \mathbf{v}.\Phi \wedge \Psi)(\mathbf{c}) \equiv (\lambda \mathbf{v}.\Phi)(\mathbf{c}) \wedge (\lambda \mathbf{v}.\Psi)(\mathbf{c})$
2. $(\lambda \mathbf{v}.\neg\Phi)(\mathbf{c}) \equiv \neg(\lambda \mathbf{v}.\Phi)(\mathbf{c})$
3. $(\lambda \mathbf{v}.(\forall x)\Phi)(\mathbf{c}) \equiv (\forall x)(\lambda \mathbf{v}.\Phi)(\mathbf{c})$ *provided x is not in the list* $\mathbf{v}$

4. $(\lambda \mathbf{v}.(\lambda x.\Phi)(d))(\mathbf{c}) \equiv (\lambda x.(\lambda \mathbf{v}.\Phi)(\mathbf{c}))(d)$ *provided* x *is not in the list* $\mathbf{v}$*.*

Even in a classical setting, however, the use of lambda abstraction can be valuable. We will see in Section 10 that if terms are allowed to be non-designating, it is natural to assign meanings in a way that makes item 2. above no longer valid.

In automated theorem proving for classical logic it is most common to convert a formula to clause form. In modal logic, this is not a tool that is available, since quantifiers can't generally be moved outside modal operators. Indeed, without predicate abstraction even Skolemization is not possible, since both $\Box(\exists x)P(x)$ and $(\exists x)\Box P(x)$ would convert to the same thing. But with predicate abstraction available we are able to Skolemize 'in place' so to speak, thus eliminating existential quantifiers. In the following we assume the notion of a positive or a negative occurrence of a subformula is known.

Proposition 5.3 (Skolemization) *Suppose* Φ *is a sentence in which* $(\forall x)\Psi$ *occurs as a negative subformula. Say this subformula occurrence is within the scope of only positive occurrences of universal quantifiers, and these involve the variables* $\mathbf{y}$*. Let* f *be a function symbol that does not occur in* Φ*, and let* Φ^* *be the result of replacing the negative occurrence of* $(\forall x)\Psi$ *in* Φ *with* $(\lambda x.\Psi)(f(\mathbf{y}))$*. Then* Φ *is satisfiable if and only if* Φ^* *is satisfiable.*

The semantic proof of this is much like the classical one, and is omitted. It was stated entirely in terms of negative occurrences of the universal quantifier. It is understood that positive occurrences of the existential quantifier are treated similarly, since they translate into negative universal occurrences. Repeated application of this Proposition allows us to fully Skolemize any sentence.

As an example, consider the sentence $(\forall w)\Box(\exists z)(\forall x)(\exists y)[R(x,y) \supset R(z,w)]$. If we negate this and perform a few elementary transformations we get $(\exists w)\Diamond(\forall z)(\exists x)(\forall y)[R(x,y) \wedge \neg R(z,w)]$. Skolemizing this produces $(\lambda w.\Diamond(\forall z)(\lambda x.(\forall y)[R(x,y) \wedge \neg R(z,w)])(f(z)))(c)$. It is not hard to show that this implies the sentence $(\lambda w.\Diamond[(\lambda z,x,y.R(x,y) \wedge \neg R(z,w))(c,f(z),w) \wedge (\lambda z,x,y.R(x,y) \wedge \neg R(z,w))(f(c),f(z),c)])(c)$. It is straightforward to check that this is not satisfiable, and so neither is $(\exists w)\Diamond(\forall z)(\exists x)(\forall y)[R(x,y) \wedge \neg R(z,w)]$. Then
$(\forall w)\Box(\exists z)(\forall x)(\exists y)[R(x,y) \supset R(z,w)]$ must be valid.

In the preceeding paragraph we produced a kind of Herbrand expansion, suggesting that there is a Herbrand Theorem in the background. Herbrand's Theorem can be looked at as a way of reducing a problem about first-order provability to problems about propositional provability. As such, it is one of a family of closely related theorems. In [5] we used a more limited version of lambda abstraction (without function symbols) to state and prove a modal analog of a theorem of Smullyan. Smullyan's Theorem, like Herbrand's, is a reduction from first-order to propositional provability, though the mechanics are different. In [12] a Herbrand Theorem using the bullet operator is given. The presence of function symbols and predicate abstraction makes it possible now to give a full modal analog of Herbrand's theorem, but its statement is of considerable complexity, and we do not give it.

6 A Tableau System

We give a tableau proof procedure that is sound and complete with respect to the semantics considered above. This tableau version is suitable for hand calculation; we will briefly discuss automation issues later. The underlying idea is an old one, and was anticipated in [2], and explicitly given in [3] and [7] for systems without non-rigid designators. The idea is to introduce a convenient syntactic device for 'naming' possible worlds, and turn semantic notions into syntax manipulations. We use finite sequences of positive integers for this purpose. The underlying idea is very elementary. If we think of the sequence $1, 2, 1$, say, as naming a possible world, then the simple extensions $1, 2, 1, 1$ and $1, 2, 1, 2$ and $1, 2, 1, 3$, and so on, all name worlds accessible from it. In general, if σ is a finite sequence and n is a positive integer, we write $\sigma\, n$ for the sequence resulting when n is added to the end of σ. Then $\sigma\, n$ is intended to name a world accessible from the world σ names.

Definition 6.1 A *prefixed formula* is an expression of the form $\sigma\ \Phi$ where σ is a finite sequence of positive integers (called a *prefix*) and Φ is a formula.

A tableau is a tree of a certain sort, with nodes labeled by prefixed formulas. We write these trees with the root node at the top and the leaves at the bottom. We give rules for 'growing' a tableau, and we begin with the propositional cases, and consider examples, before we go on to the complications brought by quantifiers.

Definition 6.2 We say a prefix σ is *available* on a tableau branch θ if there is a prefixed formula $\sigma\ \Phi$ on θ. Likewise we say σ is *unrestricted* on θ provided, for every prefixed formula $\tau\ \Phi$ on θ, σ is not an initial segment of τ (proper or not).

The intuition here is straightforward. An available prefix, informally, names a world whose existence has already been established. An unrestricted prefix has no previously determined meaning.

Now, the *propositional branch extension rules* are as follows. Let $\mathcal{T}$ be a tableau, and let θ be a branch of $\mathcal{T}$. Then:

1. if $\sigma\ \neg\neg X$ occurs on θ, $\sigma\ X$ may be added to the end of θ;
2. if $\sigma\ (X \wedge Y)$ occurs on θ, both $\sigma\ X$ and $\sigma\ Y$ may be added to the end of θ;
3. if $\sigma\ \neg(X \wedge Y)$ occurs on θ, a left child and a right child may be created for the last node of θ, one labeled $\sigma\ \neg X$, the other labeled $\sigma\ \neg Y$;
4. if $\sigma\ \Box X$ occurs on θ, $\sigma\, n\ X$ may be added to the end of θ for any $\sigma\, n$ that is available on θ;
5. if $\sigma\ \neg\Box X$ occurs on θ, $\sigma\, n\ \neg X$ may be added to the end of θ, where n is the smallest integer such that $\sigma\, n$ is unrestricted on θ.

Applying a branch extension rule to a branch of a tableau $\mathcal{T}$ yields another tableau.

Definition 6.3 A branch of a tableau is called *closed* if it contains both $\sigma\ X$ and $\sigma\ \neg X$, for some sentence X. A tableau is closed if each branch is closed.

We still must say how to start. The tableau system is designed only to prove *sentences.* In an attempt to prove the sentence X, we begin with a one-branch, one-node tree, with that node labeled $1\ \neg X$. Then we apply the branch extension rules. If we ever produce a closed tableau, we have a *proof* of X.

The ideas are rather straightforward. By beginning with $1\ \neg X$ we are informally supposing there is a world, named by 1, in which $\neg X$ is true. A closed tableau tells us this is impossible, and so X must be valid. We will make this more precise later on, but now we present an example of a proof using the rules so far. The proof is of the formula $(\Box P \wedge \Box Q) \supset \Box(P \wedge Q)$. Actually, we must translate away defined connectives, and so we really prove the formula $\neg((\Box P \wedge \Box Q) \wedge \neg\Box(P \wedge Q))$. Equally well, we could give derived branch extension rules to cover the defined connectives. The proof is contained in Figure 1. Note that the line $1, 1\ \neg(P \wedge Q)$ comes from $1\ \neg\Box(P \wedge Q)$, and at the point when the line is added, $1, 1$ is unrestricted on the branch. Likewise $1, 1\ P$ comes from $1\ \Box P$, and at the point when it is added, $1, 1$ is available on the branch.

Next we move on to the branch extension rules for quantifiers. In order to do this we need to introduce a little more machinery. First, in tableau systems for classical logic, as in [19], proofs of formulas from a first-order language L involve formulas from a larger language, say L^*. The larger language is like L except that an extra, countable set of new constant symbols has been added. These are called *parameters*; their purpose is to act like Skolem constants. Well, we must do the same thing here. So from now on, a *parameter* is a constant symbol not part of the formal language with which we began. We are going to treat these as rigid designators, and so their role will be somewhat different from constant symbols of the original language, in several respects.

Next, non-rigid constant symbols designate different things in different worlds. If c is a non-rigid constant symbol and σ is a prefix, we will write c_σ and think of it informally as the object that c designates in the world that σ names. Similarly for function symbols. We introduce a technical term for all this.

Definition 6.4 The notion of an *object expression* is given as follows:

1. if c is a (non-rigid) 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 $o_1, \ldots,\ o_n$ are object expressions, $f_\sigma(o_1, \ldots, o_n)$ is an object expression.

The notion of availability must be extended to object expressions. We say an object expression o is *available* on a branch if every subscript in o is available on the branch.

We will need things that are like formulas, except that object expressions are allowed to occur in them. We call such things *generalized formulas.* They occur in proofs, but are not part of the underlying language.

The *quantifier branch extension rules* are as follows. Let $\mathcal{T}$ be a tableau, and let θ be a branch of $\mathcal{T}$. Then:

1. if $\sigma\ (\forall x)\Phi(x)$ occurs on θ then $\sigma\ \Phi(o)$ may be added to the end of θ for any object expression o that is available on θ;

$$1\ \neg\neg((\Box P \wedge \Box Q)) \wedge \neg\Box(P \wedge Q))$$

$$1\ (\Box P \wedge \Box Q) \wedge \neg\Box(P \wedge Q)$$

$$1\ \Box P \wedge \Box Q$$

$$1\ \neg\Box(P \wedge Q)$$

$$1\ \Box P$$

$$1\ \Box Q$$

$$1,1\ \neg(P \wedge Q)$$

$$1,1\ P$$

$$\begin{tikzcd}
 & 1,1\ Q \arrow[dl, dash] \arrow[dr, dash] & \\
1,1\ \neg P & & 1,1\ \neg Q
\end{tikzcd}$$

Figure 1: A Propositional Tableau Proof

2. if $\sigma\ \neg(\forall x)\Phi(x)$ occurs on θ then $\sigma\ \neg\Phi(p)$ may be added to the end of θ, where p is a parameter that is new to θ.

In the rules above, we used $\Phi(o)$ as an informal way of designating the result of substituting occurrences of o for all free occurrences of x in $\Phi(x)$. It is expressions like $\Phi(o)$ that we will call *generalized formulas.*

As a very simple example, we give a tableau proof of $(\forall x)\Box P(x) \supset \Box(\forall x)P(x)$, or rather, of $\neg((\forall x)\Box P(x) \wedge \neg\Box(\forall x)P(x))$. This is known to be characteristic for Kripke models in which all possible worlds have the same associated quantification domains. In the proof, given in Figure 2, the line $1, 1\ \neg P(p)$ comes from $1, 1\ \neg(\forall x)P(x)$; p is a parameter, and all parameters are new to the branch at this point. Likewise, line $1\ \Box P(p)$ is from line $1\ (\forall x)\Box P(x)$, using the fact that p is an available object expression.

Finally we give the rules for the abstraction operator. And here we need one more piece of notation. Suppose σ is a prefix, and t is like a closed term, except that some of the constant and function symbols may have prefixes as subscripts. By $t@\sigma$ we mean the expression that is like t except that all unsubscripted function symbols and constant symbols (other than parameters) have had σ attached as a subscript. For example, say p is a parameter, while c and d are non-rigid constant symbols and f and g are function symbols. If $t = f_{1,2}(g(c_1), d, p)$ then $t@1,1 = f_{1,2}(g_{1,1}(c_1), d_{1,1}, p)$.

Now, the *abstraction branch extension rules* are as follows. Let $\mathcal{T}$ be a tableau, and let θ be a branch of $\mathcal{T}$. Then:

1. if $\sigma\ (\lambda x.\Phi(x))(t)$ occurs on θ then $\sigma\ \Phi(t@\sigma)$ may be added to the end of θ;

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

Once again we give a simple example, a proof of
$(\forall x)\Box(\lambda y.R(x, y))(f(x)) \supset (\forall x)\Box(\exists y)R(x, y)$, or rather of
$\neg[(\forall x)\Box(\lambda y.R(x, y))(f(x)) \wedge \neg(\forall x)\Box\neg(\forall y)\neg R(x, y)]$. The proof is contained in Figure 3; we leave the justification of the steps to you.

7 Soundness and Completeness

Soundness of the tableau system is easy to establish. As usual with both tableau and resolution systems, we want to show the rules preserve satisfiability. But since entries on tableau branches can involve object expressions and prefixes, as well as more conventional formula constructs, we must say how we are interpreting them. In fact, we simply do the obvious thing.

Suppose we have a non-rigid Kripke model $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, v\rangle$, which we hold fixed for now.

If S is a set of prefixes, a *prefix map*, $\mathcal{I}$, from S to $\mathcal{M}$ is a function that assigns possible worlds to prefixes in a way that respects the syntactical machinery of the prefixes. Specifically, we require that if $\sigma, \sigma\, n \in S$ then $\mathcal{I}(\sigma)\mathcal{R}\mathcal{I}(\sigma\, n)$. Informally, we can think of $\mathcal{I}(\sigma)$ as the world that σ names.

Next, a *parameter map* $\mathcal{P}$ to $\mathcal{M}$ is simply a function from the set of parameters to $\mathcal{D}$.

$1\ \neg\neg((\forall x)\Box P(x) \wedge \neg\Box(\forall x)P(x))$

$1\ (\forall x)\Box P(x) \wedge \neg\Box(\forall x)P(x)$

$1\ (\forall x)\Box P(x)$

$1\ \neg\Box(\forall x)P(x)$

$1,1\ \neg(\forall x)P(x)$

$1,1\ \neg P(p)$

$1\ \Box P(p)$

$1,1\ P(p)$

Figure 2: A First-Order Tableau Proof

$1\ \neg\neg[(\forall x)\Box(\lambda y\,.\,R(x,y))(f(x)) \wedge \neg(\forall x)\Box\neg(\forall y)\neg R(x,y)]$

$1\ (\forall x)\Box(\lambda y\,.\,R(x,y))(f(x)) \wedge \neg(\forall x)\Box\neg(\forall y)\neg R(x,y)$

$1\ (\forall x)\Box(\lambda y\,.\,R(x,y))(f(x))$

$1\ \neg(\forall x)\Box\neg(\forall y)\neg R(x,y)$

$1\ \neg\Box\neg(\forall y)\neg R(p,y)$

$1\ \Box(\lambda y\,.\,R(p,y))(f(p))$

$1,1\ \neg\neg(\forall y)\neg R(p,y)$

$1,1\ (\forall y)\neg R(p,y)$

$1,1\ (\lambda y\,.\,R(p,y))(f(p))$

$1,1\ R(p, f_{1,1}(p))$

$1,1\ \neg R(p, f_{1,1}(p))$

Figure 3: A Tableau Proof Involving Abstraction

Finally, if we have both a prefix map $\mathcal{I}$ and a parameter map $\mathcal{P}$, we can assign a member of $\mathcal{D}$ to each object expression in a straightforward way. We denote the member assigned to the object expression o by $o^{\mathcal{I},\mathcal{P}}$. The definition is as follows:

1. for a parameter p, $p^{\mathcal{I},\mathcal{P}} = \mathcal{P}(p)$;

2. for a non-rigid constant symbol c and a prefix $\sigma \in S$, $(c_\sigma)^{\mathcal{I},\mathcal{P}} = v(c, \mathcal{I}(\sigma))$;

3. for a function symbol f and a prefix $\sigma \in S$,
   $[f_\sigma(o_1, \ldots, o_n)]^{\mathcal{I},\mathcal{P}} = v(f, \mathcal{I}(\sigma))(o_1^{\mathcal{I},\mathcal{P}}, \ldots, o_n^{\mathcal{I},\mathcal{P}})$.

Next, the notion of *truth* for a generalized sentence $\Phi(o_1, \ldots, o_n)$, in a model $\mathcal{M}$, at a world Γ, under a prefix map $\mathcal{I}$ and a parameter map $\mathcal{P}$ is defined in the obvious way. We omit details.

The following connection with earlier definitions is not hard to establish. Let $\Phi(x_1, \ldots, x_n)$ be a formula (not a generalized formula), and let o_1,..., o_n be object expressions. Let $\mathcal{M}$ be a model, and let $\mathcal{I}$ and $\mathcal{P}$ be a prefix map and a parameter map to $\mathcal{M}$. Finally, let s be an interpretation such that, for each i, $s(x_i) = o_i^{\mathcal{I},\mathcal{P}}$. Then for each possible world Γ we have: $\mathcal{M}, \Gamma \models \Phi(x_1, \ldots, x_n)[s]$ if and only if $\Phi(o_1, \ldots, o_n)$ is true at Γ under $\mathcal{I}$ and $\mathcal{P}$.

Finally, suppose $\sigma\ \Phi(o_1, \ldots, o_n)$ is a *prefixed* generalized sentence. Let $\mathcal{I}$ be a prefix map to a model $\mathcal{M}$ and $\mathcal{P}$ be a parameter map to $\mathcal{M}$. The *truth value of* $\sigma\ \Phi(o_1, \ldots, o_n)$ *under* $\mathcal{I}$ *and* $\mathcal{P}$ is *true* if $\Phi(o_1, \ldots, o_n)$ is *true* in $\mathcal{M}$, under $\mathcal{I}$ and $\mathcal{P}$, at the world $\mathcal{I}(\sigma)$, and is *false* otherwise.

A set S of prefixed generalized sentences is *satisfiable* if there is some model $\mathcal{M}$, some parameter map $\mathcal{P}$, and some prefix map $\mathcal{I}$, defined on all prefixes occurring in S such that every member of S is true under $\mathcal{I}$ and $\mathcal{P}$.

A tableau branch is satisfiable if the set of prefixed generalized sentences on it is satisfiable. A tableau is satisfiable if some branch is satisfiable.

Lemma 7.1 *The application of a tableau rule to a satisfiable tableau yields another satisfiable tableau.*

The proof of this is straightforward, but tedious, and so we omit it. But from it soundness follows directly, by the following argument.

Suppose Φ is not valid. Then $\{1\ \neg\Phi\}$ is easily seen to be satisfiable, and so the initial tableau in a proof attempt for Φ is satisfiable. Then continued application of the tableau rules can only produce satisfiable tableaux. A satisfiable tableau can not be closed, hence no proof of Φ is possible. Stating this in a more positive form, we have the following.

Theorem 7.2 (Soundness) *If Φ has a tableau proof then Φ is valid.*

Completeness also follows using standard tableau style arguments. Suppose, in constructing tableaux, we follow some *systematic* construction procedure. There are many such. What is essential is that, unless we manage to produce a closed tableau, we eventually apply each applicable rule. Thus, if $\sigma\ (X \wedge Y)$ is on a branch, eventually we add $\sigma\ X$ and $\sigma\ Y$. If $\sigma\ (\forall x)\Phi(x)$ is on a branch, and o is any available object expression, we eventually add $\sigma\ \Phi(o)$. And so on. When we refer to a *systematic* tableau construction, we assume some particular one has been specified; we omit details.

Suppose we try to prove Φ, and so we begin a tableau with $1\ \neg\Phi$. Now, continue using a systematic tableau construction procedure. If we do not generate a closed tableau, there are two possibilities. Either the procedure terminates, leaving an unclosed branch, or the procedure never terminates, in which case König's Lemma says there is an infinite branch generated, which must be unclosed. Either way, there is an unclosed branch θ on which no rules remain unapplied.

Using θ, we construct a model in a straightforward way. We take for $\mathcal{G}$ the set of prefixes occurring on θ. We define $\mathcal{R}$ by the condition $\sigma\mathcal{R}\sigma\, n$. $\mathcal{D}$ is the set of object expressions. $v(c,\sigma) = c_\sigma$ for constant symbols. $v(f,\sigma)(o_1,\ldots,o_n) = f_\sigma(o_1,\ldots,o_n)$ for function symbols. And for a relation symbol P, $v(P,\sigma) = \{\langle o_1,\ldots,o_n\rangle \mid \sigma\ P(o_1,\ldots,o_n) \text{ occurs on } \theta\}$. This gives us a model $\mathcal{M} = \langle\mathcal{G},\mathcal{R},\mathcal{D},v\rangle$.

Let $\mathcal{I}$ be the identity prefix map, and let $\mathcal{P}$ be the identity parameter map. Now, an induction on complexity will show that each prefixed generalized sentence on θ is true under $\mathcal{I}$ and $\mathcal{P}$ in $\mathcal{M}$. Since $1\ \neg\Phi$ is on θ, it follows that Φ is false at world 1 of the model $\mathcal{M}$, and so Φ is not valid. Thus we have shown the following.

Theorem 7.3 (Completeness) *If Φ is valid then Φ has a tableau proof.*

8 Equality

The proper role of equality in modal logic has been controversial for a long time. See, for instance, [10]. Generally one thinks of substitutivity as the distinguishing characteristic of the equality relation. But then we immediately run into the well-known morning star/evening star problem. Since the morning star = the evening star, we should be able to substitute an occurrence of 'evening star' in any sentence containing an occurrence of 'morning star' without affecting its truth value. The sentence, "The Greeks knew that the morning star is the morning star," is true. Substituting for the second occurrence of 'morning star,' we get, "The Greeks knew that the morning star is the evening star," and this is false.

Put more baldly, the issue comes down to whether one wants to accept as a correct principle all instances of $(a = b) \supset \Box(a = b)$. Much argument has been generated by this question. Incidentally, it is no coincidence that the sentence displayed is not syntactically correct in our present system. The subformula that reads $\Box(a = b)$ should be replaced by one of $(\lambda x, y.\Box(x = y))(a, b)$ or by $\Box(\lambda x, y.(x = y))(a, b)$. Thus in fact, there are really two distinct 'official' versions of the problem sentence, and these turn out to represent the two sides the argument has taken.

There is no problem with substitutivity at the level of objects. If we say two objects are equal, we mean we don't have two objects after all. The difficulties arise because we don't use objects when we talk; we use names for them, and names that designate the same object in one possible world need not do so in a different one. But the observation that there is no problem at the object level is the key to the problem, and leads us to an extremely simple solution.

Definition 8.1 A non-rigid Kripke model $\mathcal{M} = \langle\mathcal{G},\mathcal{R},\mathcal{D},v\rangle$ is *normal* if, at each possible world, the interpretation of the two-place predicate $=$ is by the equality relation. That is, for each $\Gamma \in \mathcal{G}$,

$v(=,\Gamma)$ is the equality relation on $\mathcal{D}$.

We noted above that the sentence giving us trouble actually had two official versions in our syntax. These are:

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

Now, item 1. turns out to be valid in all normal models, but it is easy to produce examples that show item 2. is not.

Returning to the morning star/evening star problem, a resolution is now straightforward. The sentence, "The Greeks knew that the morning star is the evening star," is false. The natural formalization of it is:

$$\mathit{Greeks\_knew}(\lambda x, y.(x = y))(\mathit{morning\_star},\ \mathit{evening\_star})$$

This formalization corresponds to the version of substitution in item 2. above, which is not generally valid. Item 1. is the correct substitutivity principle, but it is not relevant to the morning star/evening star issue.

We conclude this section with tableau rules for equality. They are simple to state and use, though we omit proofs of soundness and completeness. There are only two rules we need.

Reflexive Rule For any available object expression o and any available prefix σ, the generalized sentence $\sigma\ o = o$ may be added to the end of any branch.

Substitution Rule If $\Phi(x)$ is a formula with only x free, then to any tableau branch containing $\sigma\ \Phi(o_1)$ and $\sigma\ o_1 = o_2$ we may add $\sigma\ \Phi(o_2)$.

Note that in the Substitution Rule o_1 and o_2 are object expressions, not terms. Substitutivity applies to objects, not to names for them. It is an interesting exercise to use these rules to give a proof of item 1. above, and to attempt to give a proof of item 2.

9 Definite Descriptions

Expressions of the form "the so-and-so such that . . . " arise constantly in natural language. Bertrand Russell [18] gave a famous formal treatment of them in classical logic, essentially showing how to translate such espressions away. Under his theory the sentence, "The King of France is bald," translates into "there is one and only one object having the property of being the King of France, and that object is bald." (In this example, the translated version is false, since no object has the property of being the King of France.) Russell's theory has been fundamental ever since its introduction.

On the other hand, Kripke [15] has argued forcefully that the Russell treatment does not extend to modal contexts. After all, if it did, how would we deal with something like "The President of the United States someday won't be the President of the United States." It would wind up as something like, "in some possible future, there is one and only one person who is the President of the United States, and that person is not the President of the United States." Clearly this is silly.

The papers [20] and [21] present a treatment of definite descriptions using their abstraction device in which, loosely speaking, they are treated as primitives. But surprisingly enough, it turns out that the Russell technique of translation works very well in modal contexts provided it is combined with the predicate abstraction device. We treat 'the object such that ... ' syntactically as a constant symbol, following the rules outlined earlier. But semanticaly we translate it away, exactly as Russell did. No other changes need to be made. In the following we use Russell's notation, reading $(\iota y)\Psi(y)$ as 'the y such that $\Psi(y)$. (Actually, Russell used an inverted ι, which we don't happen to have available.) Thus, from now on, $(\lambda x.\Phi)((\iota y)\Psi(y))$ will be counted as a formula.

Definition 9.1 If Ψ is a formula and y is a variable then $(\iota y)\Psi$ is a term.

Definition 9.2 $(\lambda x.\Phi(x))((\iota y)\Psi(y))$ is taken semantically to abbreviate $(\exists x)[\Psi(x) \wedge (\forall w)(\Psi(w) \supset (w = x)) \wedge \Phi(x)]$. (Here it is assumed that x is a variable that does not occur in $\Psi(y)$ and w does not occur in either $\Phi(x)$ or $\Psi(y)$.)

Now, we return to the example of the President of the United States, mentioned earlier. Suppose $P(x)$ is a formula that legally defines the presidency. Then what is being asserted when we say "The President of the United States someday won't be the President of the United States," is that the property of possibly not being President applies to the person who currently is the President. Symbolically this is just $(\lambda x.\Diamond\neg P(x))((\iota y)P(y))$. This translates into $(\exists x)[P(x) \wedge (\forall w)(P(w) \supset (w = x)) \wedge \Diamond\neg P(x)]$. To say this is true at a possible world Γ of some Kripke model is to say there is some object o that, at Γ, is the only object for which $P(x)$ is true, and there is a possible world Δ, alternative to Γ, such that at Δ, o does not make $P(x)$ true. This is a perfectly reasonable formal version of what we meant informally. Since it is easy to construct a model that behaves this way, the sentence $(\lambda x.\Diamond\neg P(x))((\iota y)P(y))$ is satisfiable.

We might point out that the silly reading we gave to "The President of the United States someday won't be the President of the United States" earlier formalizes as $\Diamond(\lambda x.\neg P(x))((\iota y)P(y))$. This sentence, in fact, is not satisfiable.

10 Existence

Much virtual bloodshed has occurred in battles over the correct use of non-designating names. If it is true that "Pegasus does not exist," of what is non-existence being asserted? We wish to avoid controversy, but we do recognize that we have here an issue whose formal treatment will have significant consequences for natural language processing. We propose a technical solution that has the advantage of also having nice intuitive features. We don't assert that this technical device solves the philosophical problems, but it does allow us to get on with the formal work.

In more conventional treatments of modal logic, with rigid designators, it is common to allow domains to vary from world to world. Then one has the problem of interpreting formulas containing a constant symbol in worlds where the object designated by that symbol does not happen to be in the domain. A common device, used by Kripke [14], [10], is to take any *atomic* formula containing a non-designating constant as being false. We propose adopting a similar solution here, but replacing atomic by *predicate abstraction.*

Recall the intuitive motivation behind predicate abstraction. A formula like $\Phi(x)$ is turned into a predicate abstraction using lambda: $(\lambda x.\Phi(x))$. Loosely, $(\lambda x.\Phi(x))$ is a *property*, while $\Phi(x)$ is a *formula*, in the same sense that $(\lambda x.x+1)$ is a *function*, while $x+1$ is a *term.* Now the intuitive idea is simple: *only things that exist have properties.* The rest of the section is devoted to making this precise, and looking at its consequences.

From now on we broaden the notion of Kripke frame and model by allowing the interpretation of constant and function symbols to be partial. More precisely, we consider $\langle \mathcal{G}, \mathcal{R}, \mathcal{D}, v\rangle$ where all is as before, except that:

1. v is a *partial function* on constant symbols and worlds, that is, $v(c,\Gamma)$ is defined only for some c and Γ, not necessarily for all;

2. v assigns to function symbols and worlds *partial functions*, that is, $v(f,\Gamma)$ is a mapping from a subset of $\mathcal{D}^n$ to $\mathcal{D}$, possibly a proper subset.

The result of all this is that the extension of an interpretation s to terms will also be partial. That is, for a term t, $s(t,\Gamma)$ may turn out to be undefined.

Definition 10.1 Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, v\rangle$ be a model (in the sense above, allowing non-designating terms). The definition of $\models$ is exactly as in Definition 4.5, except that item 6. is replaced by the following:

$$\mathcal{M},\Gamma \models (\lambda x.\Phi)(t)[s] \text{ provided } s(t,\Gamma) \text{ exists, and } \mathcal{M},\Gamma \models \Phi[s\left[\tfrac{a}{x}\right]], \text{ where } a = s(t,\Gamma).$$

Note that, as a consequence, $(\lambda x.\Phi)(t)$ will be false at any world where t doesn't designate. It is enlightening to introduce an *existence* predicate. This is easily defined in terms of equality.

Definition 10.2 $E(x)$ stands for $(x = x)$.

The sentence $(\forall x)E(x)$ is valid, as it should be. After all, everything exists (show me something that doesn't). In a world in which the constant symbol a designates, $(\lambda x.E(x))(a)$ will be true (because the object that a names equals itself), and in a world in which a does not designate, $(\lambda x.E(x))(a)$ will be false (because predicate abstractions are taken to be false of non-designating terms).

What is a little more surprising is that, at a world in which a does not designate, $(\lambda x.\neg E(x))(a)$ will be false (again because predicate abstractions are false of non-designating terms). But the sentence $\neg(\lambda x.E(x))(a)$ will be true, since it is the negation of something we determined was false above. Thus, if a does not designate, then a does not have the property of existence: $\neg(\lambda x.E(x))(a)$ is true. But we can not say it does have the property of non-existence: $(\lambda x.\neg E(x))(a)$ is false. In fact, our device has made it possible to talk of a non-existence property, distinct from the negation of an existence property. The solution to the Pegasus problem, then, is that it is correct to say it is not the case that Pegasus exists, but it is false to say that Pegasus has the property of non-existence. In fact, nothing has the non-existence property, $(\forall x)\neg(\lambda x.\neg E(x))(x)$ is a valid sentence.

We noted in Section 5 that the lambda abstraction device was transparent to the operations of classical logic. In fact, this is no longer true if we allow non-designating terms to occur. We just showed that it was possible to have $\neg(\lambda x.E(x))(a)$ true but $(\lambda x.\neg E(x))(a)$ false. Consequently $\neg(\lambda x.E(x))(a) \supset (\lambda x.\neg E(x))(a)$ is not valid.

We have dealt with the problem of talking about the non-existence of something that, in fact, does not exist. As we said above, it amounts to a formalization of the intuition that non-existing things don't have properties. Now what can we do with a sentence like, "Pegasus has wings?" Given the predicate abstraction machinery, how might we formalize this? We take the sentence to mean that, even though Pegasus does not exist in this world, in every world in which it does exist, it has wings. We do not wish to defend this reading philosophically. We merely observe that it is technically convenient, and does not seem to violate any intuitions we have about non-existent things. Now, suppose $F(x)$ is a formula that characterizes the flying horse usually called Pegasus, and suppose $W(x)$ is a formula asserting that x has wings. Then the formalization we propose for "Pegasus has wings" is:

$$\Box(\lambda x.E(x) \supset W(x))((\iota y)F(y))$$

Presumably the formula $F(x)$ characterizing Pegasus has a clause covering the presence of wings, and so $(\forall x)(F(x) \supset W(x))$ is valid. It is easy to see that this is enough to ensure the validity of our formalization of "Pegasus has wings." Incidentally, since contradictions imply everything, a formalization of "The round square is green," will also be valid. But this ought not be surprising; there are no possible worlds in which a round square exists that is not green.

We consider one more example. It is required that the President of the United States be (born) a citizen. How might we formalize this? Suppose we let $P(x)$ be a formula defining the presidency, as above, and let $C(x)$ be a formula defining citizenship. Then a first attempt at a formalization is:

$$\Box(\lambda x.C(x))((\iota y)P(y))$$

This attempt fails, though. For it to be true at a possible world it must be the case that $(\lambda x.C(x))((\iota y)P(y))$ holds at every accessible world, and for this to be so, there must be a President of the United States in every such world. But there are unfortunate times when there is no President. The requirement that a President be a citizen surely is not also a requirement that there always be a President. Rather what is meant is, it is legally necessary that, if there is a President, then that person must be a citizen. Then a more reasonable formalization is:

$$\Box\,[(\lambda x.E(x))((\iota y)P(y)) \supset (\lambda x.C(x))((\iota y)P(y))]$$

It is easy to check that this formula is satisfiable, and captures the intended legal requirement.

A sound and complete tableau system allowing non-designating terms is easy to produce. We need to change three rules. The first is the one for the universal quantifier. It should be changed to: if $\sigma\ (\forall x)\Phi(x)$ occurs on a branch θ then $\sigma\ \Phi(o)$ may be added to the end of θ for any object expression o *that already occurs on* θ. The second rule change involves the Reflexive Rule for equality. It should be restricted to: $\sigma\ o = o$ can be added to θ for any available σ and any available object expression o *that already occurs on* θ. The final rule change involves negated predicate abstractions. It becomes: if $\sigma\ \neg(\lambda x.\Phi(x))(t)$ occurs on θ then $\sigma\ \neg\Phi(t@\sigma)$ may be added to the end of θ *provided* $t@\sigma$ *already occurs on* θ. With these changes, there are only two rules that can introduce new object expressions to a branch. One is the rule for a negated universal quantifier, which introduces a new parameter. The other is the rule for an unnegated predicate abstraction. In each of these cases, if the premise of the rule is true at a possible world, we are guaranteed the existence of a suitable object. Thus, in the modified system, we are only allowed to add object expressions when existence is a sure thing, and we are only allowed to use object expressions that we know are safe.

11 Notes on implementation

It would make this paper too long to get seriously into issues of implementation. Instead we present a brief outline of an approach that seems quite promising. The specifically modal problems involved in automation all come from the branch extension rule involving un-negated occurrences of $\Box$. If $\sigma\ \Box X$ occurs on a branch, we are allowed to add $\sigma n\ X$ for *any* available prefix, and how are we to know which is best to add. Although based on resolution rather than tableaux, [16] presents an interesting family of theorem provers for first-order modal logics using a device that is, in essence, the same as that of prefixes above. The solution to the comparable problem there is to introduce a variable, and later on to use unification to decide what an appropriate value of it should be. Of course this leads to problems with negated occurrences of $\Box$, since we won't know what is unrestricted unless we already know what is present. There are ways around this, essentially using Skolem functions in prefixes. What is more interesting, though, is the mechanism used for getting the various modal logics. In [3] and [7] our solution for hand calculation consisted of modifying rules to syntactically build in reflexivity or transitivity or whatever. The tableau implentation of [8] was built on very different principles, and so its methods are not directly applicable here. But in [16], modified forms of unification are used at this point, building reflexivity or transitivity or whatever into the unification algorithm. This elegant solution can be applied to a tableaux formulation as well as to a resolution based one. Or equivalently, a resolution formulation of predicate abstraction can be developed by building on [16] directly.

But in fact, the difficulties to be faced all come from the underlying modal logic. The additional problems posed by predicate abstraction have essentially straightforward solutions, as a little thought and experimentation with pencil and paper will quickly show. Indeed, we even have a small advantage now, since Skolemization is possible, and was not when predicate abstraction was absent. Consequently the introduction of parameters during the course of a proof can be avoided, in favor of Skolemizing ahead of time.

Finally, we note that predicate abstraction does not have a utility that is confined to modal logics alone. Such a device should be considered for any first-order non-classical logic. After all, it has happened in other subjects that abstraction has brought insight.

References

[1] M. Cialdea, L. Fariñas del Cerro. A Modal Herbrand's property, *Zeitschr. f. math. Logik und Grundlagen d. Math.*, vol 32, pp 523–530 (1986).

[2] F. Fitch. Tree proofs in modal logic (abstract), *Journal of Symbolic Logic*, vol 31, p 152 (1966).

[3] M. Fitting. Tableau methods of proof for modal logics, *Notre Dame Journal of Formal Logic*, vol 13, pp 237–247 (1972).

[4] 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).

[5] 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).

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

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

[8] M. Fitting. First-Order modal tableaux, *Journal of Automated Reasoning*, pp 191–213, vol 4 (1988).

[9] C. Geissler, K. Konolige. A Resolution method for quantified modal logics of knowledge and belief, *Reasoning About Knowledge, Proc. of the 1986 conf.*, J. Y. Halpern, editor, pp 309–324, Morgan Kaufman (1986).

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

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

[12] K. Konolige. *A deduction model of belief*, Morgan Kaufman, Los Altos, CA (1986).

[13] 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).

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

[15] S. Kripke. *Naming and Necessity*, Harvard University Press, Cambridge (1980).

[16] 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).

[17] J. A. Robinson. A Machine-oriented logic based on the resolution principle, *Journal of the ACM*, vol 12, pp 23–41 (1965).

[18] B. Russell. On denoting, *Mind*, new series, vol 14, pp 479–493 (1905).

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

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

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