Modality and Databases

Melvin Fitting

Dept. Mathematics and Computer Science
Lehman College (CUNY), Bronx, NY 10468
e-mail: fitting@alpha.lehman.cuny.edu
web page: comet.lehman.cuny.edu/fitting

Abstract. Two things are done in this paper. First, a modal logic in which one can quantify over both objects and concepts is presented; a semantics and a tableau system are given. It is a natural modal logic, extending standard versions, and capable of addressing several well-known philosophical difficulties successfully. Second, this modal logic is used to introduce a rather different way of looking at relational databases. The idea is to treat records as possible worlds, record entries as objects, and attributes as concepts, in the modal sense. This makes possible an intuitively satisfactory relational database theory. It can be extended, by the introduction of higher types, to deal with multiple-valued attributes and more complex things, though this is further than we take it here.

1 Introduction

A few years ago my colleague, Richard Mendelsohn, and I finished work on our book, "First-Order Modal Logic," [2]. In it, non-rigidity was given an extensive examination, and formal treatments of definite descriptions, designation, existence, and other issues were developed. I next attempted an extension to higher-order modal logic. After several false starts (or rather, unsatisfactory finishes) this was done, and a book-length manuscript is on my web page inviting comments, [1]. Carrying out this extension, in turn, led me to rethink the first-order case. There were two consequences. First, I came to realize that the approach in our book could be extended, without leaving the first-order level, to produce a quite interesting logic with a natural semantics and a tableau proof procedure. And second, I realized that this modal logic provided a natural alternative setting for relational databases, which are usually treated using first-order classical logic. In this paper I want to sketch both the modal logic and its application to databases.

In a full treatment of first-order modal logic, one must be able to discourse about two kinds of things: *individual objects* and *individual concepts*. "George Washington" and "Millard Fillmore" denote individual objects, while "the President of the United States" denotes an individual concept, which in turn denotes various individuals at different times. Or again, at the time I am writing this the year is 2000. This particular year is an individual object. "The current year" is an individual concept, and will not always denote 2000. In [2] we had quantifiers ranging over individual objects, and constant symbols with values that

were individual concepts. That was a good combination to elucidate a number of well-known philosophical problems, but it is not a full picture. In this paper the formal system presented will have quantifiers over individual objects, and also a second kind of quantifier ranging over individual concepts. Likewise there will be two kinds of constant symbols. The system of [2] can be embedded in the present one. (Of course this is only approximate. In our book we had function symbols, and we do not have them here. There are other differences as well, but the embedability claim is essentially correct.) I'll begin with a presentation of the logic, and then consider its applications to databases.

In a sense, using the modal logic of this paper to supply a semantics for relational databases does not give us anything new. We are able to treat things that, previously, had been treated using classical first-order logic. The modal point of view is substantially different, and hence interesting, but does not expand our concept of relational database. The real significance lies in what comes next, just after the conclusion of this paper. The modal logic presented here is the first-order fragment of a higher-order modal logic, with both extensional and intensional objects at each level. When such a logic is applied to database theory, we get a natural setting within which to model multiple-valued relations, relations having a field whose values are sets of attributes, and more complex things yet. Think of the present paper, then, as providing a different point of view on what is generally understood, and as signaling the approach of an extension, which can be glimpsed down the road.

2 Syntax

The syntax of this version of first-order modal logic is a little more complex than usual, and so some care must be taken in its presentation.

There are infinitely many variables and constants, in each of two categories: individual objects and individual concepts. I'll use lowercase Latin letters x, y, z as object variables, and lowercase Greek letters α, β, γ as concept variables. (Based on the notion that the ancient Greeks were the theoreticians, while the Romans were the engineers.) The Greek letter ϱ, with or without subscripts, represents a variable of either kind. For constants, I'll use lowercase Latin letters such as a, b, c for both kinds, and leave it to context to sort things out.

Definition 1 (Term). A constant symbol or a variable is a *term*. It is an *object term* if it is an individual object variable or constant symbol. Similarly for *concept terms*.

If t is a concept term, $\downarrow t$ is an object term. It is called a *relativized* term.

The idea is, if t is a concept term, $\downarrow t$ is intended to designate the object denoted by t, in a particular context. Sometimes I'll refer to $\downarrow$ as the *evaluate at* operator.

Since there are two kinds of variables and constants, assigning an arity to relation symbols is not sufficient. By a *type* I mean a finite sequence of o's and c's, such as $\langle c, o, c\rangle$. Think of an o as marking an object position and a c as marking

a concept position. There are infinitely many relation symbols of each type. In particular there is an equality symbol, $=$, of type $\langle o, o\rangle$. That is, equality is a relation on individual objects. One could also introduce a notion of equality for individual concepts, but it will not be needed here. I allow the empty sequence $\langle\rangle$ as a type. It corresponds to what are sometimes called propositional letters, taking no arguments.

Definition 2 (Formula). The set of formulas, and their free variables, is defined as follows.

1. If P is a relation symbol of type $\langle\rangle$, it is an atomic formula, and has no free variables.
2. Suppose R is a relation symbol of type $\langle n_1, n_2, \ldots, n_k\rangle$ and t_1, t_2, $\ldots$, t_k is a sequence of terms such that t_i is an individual object term if $n_i = o$ and is an individual concept term if $n_i = c$. Then $R(t_1, t_2, \ldots, t_k)$ is an atomic formula. Its free variables are the variable occurrences that appear in it.
3. if X is a formula, so are $\neg X$, $\Box X$, and $\Diamond X$. Free variable occurrences are those of X.
4. If X and Y are formulas, so are $(X \wedge Y)$, $(X \vee Y)$, and $(X \supset Y)$. Free variable occurrences are those of X together with those of Y.
5. If X is a formula and ϱ is a variable (of either kind), $(\forall\varrho)X$ and $(\exists\varrho)X$ are formulas. Free variable occurrences are those of X, except for occurrences of ϱ.
6. If X is a formula, ϱ is a variable (again of either kind), and t is a term of the same kind as ϱ, $\langle\lambda\varrho.X\rangle(t)$ is a formula. Free variable occurrences are those of X, except for occurrences of ϱ, together with those of t.

As usual, parentheses will be omitted from formulas to improve readability. Also $=(x, y)$ will be written as $x = y$. And finally, a formula like

$$\langle\lambda\varrho_1.\langle\lambda\varrho_2.\langle\lambda\varrho_3.X\rangle(t_3)\rangle(t_2)\rangle(t_1)$$

will be abbreviated by the simpler expression

$$\langle\lambda\varrho_1, \varrho_2, \varrho_3.X\rangle(t_1, t_2, t_3)$$

and similarly for other formulas involving iterated abstractions.

3 Semantics

I will only formulate an **S5** logic—the ideas carry over directly to other logics, but **S5** is simplest, the ideas are clearest when stated for it, and it is all that is actually needed for databases.

Frames essentially disappear, since we are dealing with **S5**. A model has a set of possible worlds, but we can take every world to be accessible from every other, and so no explicit accessibility relation is needed.

The usual constant/varying domain dichotomy is easily ignored. For first-order modal logics generally, a constant domain semantics can simulate a varying domain version, through the use of an explicit existence predicate and the relativization of quantifiers to it. Here I'll take object domains to be constant—not world dependent—which makes things much simpler.

Since the language has two kinds of terms, we can expect models to have two domains—two sorts, in other words. There will be a domain of individual objects, and a domain of individual concepts. Concepts will be functions, from possible worlds to individual objects. It is not reasonable, or desirable, to insist that *all* such functions be present. After all, if there are countably many possible worlds, there would be a continuum of such concept functions even if the set of individual objects is finite, and this probably cannot be captured by a proof procedure. But anyway, the notion of an individual concept presupposes a kind of coherency for that individual concept—not all functions would be acceptable intuitively. I simply take the notion of individual concept as basic; I do not try to analyze any coherency condition. It is allowed that some, not necessarily all, functions can serve as individual concepts.

Definition 3 (Model). A *model* is a structure $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I} \rangle$, where:

1. $\mathcal{G}$ is a non-empty set, of *possible worlds*;
2. $\mathcal{D}_o$ is a non-empty set, of *individual objects*;
3. $\mathcal{D}_c$ is a non-empty set of functions from $\mathcal{G}$ to $\mathcal{D}_o$, called *individual concepts*;
4. $\mathcal{I}$ is a mapping that assigns:
   (a) to each individual object constant symbol some member of $\mathcal{D}_o$;
   (b) to each individual concept constant symbol some member of $\mathcal{D}_c$;
   (c) to each relation symbol of type $\langle\rangle$ a mapping from $\mathcal{G}$ to $\{\mathit{false}, \mathit{true}\}$;
   (d) to each relation symbol of type $\langle n_1, n_2, \ldots, n_k \rangle$ a mapping from $\mathcal{G}$ to the power set of $\mathcal{D}_{n_1} \times \mathcal{D}_{n_2} \times \cdots \times \mathcal{D}_{n_k}$. It is required that $\mathcal{I}(=)$ be the constant mapping that is identically the equality relation on $\mathcal{D}_o$.

Some preliminary machinery is needed before truth in a model can be characterized.

Definition 4 (Valuation). A *valuation* v in a model $\mathcal{M}$ is a mapping that assigns to each individual object variable some member of $\mathcal{D}_o$, and to each individual concept variable some member of $\mathcal{D}_c$.

Definition 5 (Value At). Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I} \rangle$ be a model, and v be a valuation in it. A mapping $(v * \mathcal{I})$ is defined, assigning a meaning to each term, at each possible world. Let $\Gamma \in \mathcal{G}$.

1. If ϱ is a variable, $(v * \mathcal{I})(\varrho, \Gamma) = v(\varrho)$.
2. If c is a constant symbol, $(v * \mathcal{I})(c, \Gamma) = \mathcal{I}(c)$.
3. If $\downarrow t$ is a relativized term, $(v * \mathcal{I})(\downarrow t, \Gamma) = (v * \mathcal{I})(t)(\Gamma)$.

Item 3 is especially significant. If $\downarrow t$ is a relativized term, t must be a constant or variable of concept type, and so $(v * \mathcal{I})(t)$ has been defined for it in parts 1 and 2, and is a function from worlds to objects. Thus $(v * \mathcal{I})(t)(\Gamma)$ is a member of $\mathcal{D}_o$.

Now the main notion, which is symbolized by $\mathcal{M}, \Gamma \Vdash_v \Phi$, and is read: formula Φ is true in model $\mathcal{M}$, at possible world Γ, with respect to valuation v. To make reading easier, the following special notation is used. Let $\varrho_1, \ldots, \varrho_k$ be variables of any type, and let $d_1, \ldots, d_k$ be members of $\mathcal{D}_o \cup \mathcal{D}_c$, with $d_i \in \mathcal{D}_o$ if the variable ϱ_i is of object type, and $d_i \in \mathcal{D}_c$ if ϱ_i is of concept type. Then

$$\mathcal{M}, \Gamma \Vdash_v \Phi[\varrho_1/d_1, \ldots, \varrho_k/d_k]$$

abbreviates: $\mathcal{M}, \Gamma \Vdash_{v'} \Phi$ where v' is the valuation that is like v on all variables except $\varrho_1, \ldots, \varrho_k$, and $v'(\varrho_1) = d_1, \ldots, v'(\varrho_k) = d_k$.

Here is the central definition. For simplicity, take $\vee$, $\supset$, $\exists$, and $\Diamond$ as defined symbols, in the usual way.

Definition 6 (Truth in a Model). Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I} \rangle$ be a model, and v be a valuation in it.

1. If P is of type $\langle\rangle$, $\mathcal{M}, \Gamma \Vdash_v P$ iff $\mathcal{I}(P)(\Gamma) = \mathit{true}$.
2. If $R(t_1, \ldots, t_k)$ is atomic, $\mathcal{M}, \Gamma \Vdash_v R(t_1, \ldots, t_k)$ iff $\langle (v * \mathcal{I})(t_1, \Gamma), \ldots, (v * \mathcal{I})(t_k, \Gamma) \rangle \in \mathcal{I}(R)(\Gamma)$.
3. $\mathcal{M}, \Gamma \Vdash_v \neg\Phi$ iff $\mathcal{M}, \Gamma \nVdash_v \Phi$.
4. $\mathcal{M}, \Gamma \Vdash_v \Phi \wedge \Psi$ iff $\mathcal{M}, \Gamma \Vdash_v \Phi$ and $\mathcal{M}, \Gamma \Vdash_v \Psi$.
5. $\mathcal{M}, \Gamma \Vdash_v (\forall x)\Phi$ iff $\mathcal{M}, \Gamma \Vdash_v \Phi[x/d]$ for all $d \in \mathcal{D}_o$.
6. $\mathcal{M}, \Gamma \Vdash_v (\forall \alpha)\Phi$ iff $\mathcal{M}, \Gamma \Vdash_v \Phi[\alpha/d]$ for all $d \in \mathcal{D}_c$.
7. $\mathcal{M}, \Gamma \Vdash_v \Box\Phi$ iff $\mathcal{M}, \Delta \Vdash_v \Phi$ for all $\Delta \in \mathcal{G}$.
8. $\mathcal{M}, \Gamma \Vdash_v \langle \lambda\varrho.\Phi \rangle(t)$ if $\mathcal{M}, \Gamma \Vdash_v \Phi[\varrho/d]$, where $d = (v * \mathcal{I})(t, \Gamma)$.

Definition 7 (Validity). A closed formula X is *valid* in a model if it is true at every world of it.

A notion of consequence is a little more complicated because, in modal settings, it essentially breaks in two. These are sometimes called *local* and *global* consequence notions. For a set S of formulas, do we want X to be true at every world at which members of S are true (local consequence), or do we want X to be valid in every model in which members of S are valid (global consequence). These have quite different flavors. Fortunately, for **S5**, the situation is somewhat simpler than it is for other modal logics since, to say X is valid in a model is just to say $\Box X$ is true at some world of it. So, if we have a notion of local consequence, we can define a corresponding global consequence notion simply by introducing necessity symbols throughout. So, local consequence is all we need here.

Definition 8 (Consequence). A closed formula X is a *consequence* of a set S of closed formulas if, in every model, X is true at each world at which all the members of S are true.

4 Rigidity

An individual concept term t can vary from world to world in what it designates. Call t *rigid* in a model if it is constant in that model, designating the same object at each world. This is a notion that plays an important role in philosophy. For instance Kripke [3], among others, asserts that names are rigid designators. The notion of rigidity can be captured by a formula. Assume c is an individual concept constant symbol in the following.

$$\langle \lambda x.\Box(x =\!\downarrow c)\rangle(\downarrow c) \tag{1}$$

It is quite straightforward to show that (1) is valid in a model if and only if the interpretation of c is rigid in that model. In [2] this, in turn, was shown to be equivalent to the vanishing of the *de re/de dicto* distinction, though this will not be needed here.

One can also speak of *relativized* notions of rigidity. Let us say the interpretation of c is rigid on a particular subset $\mathcal{G}_0$ of the collection $\mathcal{G}$ of possible worlds of a model provided it designates the same object throughout $\mathcal{G}_0$. And let us say c is *rigid relative to* d in a model provided the interpretation of c is rigid on any subset of worlds on which the interpretation of d is rigid. (Of course, this notion applies to individual concept terms that are variables as well. I'm using constant symbols just as a matter of convenience.) The notion of c being rigid relative to d is captured by formula (2).

$$\langle \lambda x, y.\Box[x =\!\downarrow d \supset y =\!\downarrow c]\rangle(\downarrow d, \downarrow c) \tag{2}$$

One can also consider more complicated situations. Formula (3) asserts that c is rigid relative to the rigidity of d and e jointly.

$$\langle \lambda x, y, z.\Box[(x =\!\downarrow d \wedge y =\!\downarrow e) \supset z =\!\downarrow c]\rangle(\downarrow d, \downarrow e, \downarrow c) \tag{3}$$

Finally, one can even say that *all* individual concepts are rigid relative to c. This is done in formula (4). Individual concept quantification is obviously essential here.

$$(\forall \alpha)\langle \lambda x, y.\Box[x =\!\downarrow c \supset y =\!\downarrow \alpha]\rangle(\downarrow c, \downarrow \alpha) \tag{4}$$

5 Databases With a Single Relation

In this section we begin taking a look at relational databases. What we consider is quite basic, and can be found in any textbook on databases—[4] is a good source. Relational databases are commonly reasoned about using classical first-order logic. I want to show that modal logic is also a natural tool for this purpose. For now, only a single relation will be considered—this will be extended later.

The *record* is the basic unit of a relational database, yet it is not a first-class object in the sense that it is not something we can get as an answer to a query. We could get a record number, perhaps, but not a record. We will take the records of a relational database to be the *possible worlds* of a Kripke model. In any standard modal language possible worlds, in fact, cannot be directly spoken of. The accessibility relation will be the usual **S5** one, though there could be circumstances where something more complex might be appropriate.

Entries that fill fields of a relational database generally can be of several data types. To keep things simple, let's say there is just one data type used for this purpose. (In examples I'll use strings.) Looking at this in terms of modal logic, these field entries will be the *individual objects* of a Kripke model.

Next come the attributes themselves. If the database is one listing family relationships, say, and there is an attribute recording "father-of," it has a value that varies from record to record, but in every case that value is what we have taken to be an individual object. In other words, attributes are simply *individual concepts*.

The next question is, what about things like functional dependencies, keys, and so on? Let's begin with the notion of functional dependency. Say we have a relational database in which there are two attributes, call them c and d, and c is functionally dependent on d. Then, if we are at an arbitrary possible world (record) at which c and d have particular values, at any other world at which d has the value it has in this one, c must also have the same value it has in this one. This can be expressed by requiring validity of the following formula, in which we assume c and d of the Kripke model also occur as individual concept constants of the language, and designate themselves.

$$\langle \lambda x, y. \Box [x =\!\downarrow d \supset y =\!\downarrow c] \rangle (\downarrow d, \downarrow c)$$

But this is just formula (2), and says c is rigid relative to d. In this case, a functional dependency can be expressed by a relative rigidity assertion.

More complicated functional dependencies also correspond to relative rigidity formulas. For instance, if c is functionally dependent on $\{d, e\}$, this is expressed by (3).

Next, what about the notion of *keys*? As usually treated, to say an attribute c is a key is to say there cannot be two records that agree on the value of c. We cannot quite say that, since records cannot directly be spoken of. What we can say is that two possible worlds agreeing on the value of c must agree on the values of all attributes. More formally, this is expressed by the validity of the following formula.

$$(\forall \alpha) \langle \lambda x, y. \Box [x =\!\downarrow c \supset y =\!\downarrow \alpha] \rangle (\downarrow c, \downarrow \alpha)$$

Note that this is our earlier formula (4).

Now, what does the design of a relation schema look like from the present modal point of view? We must specify the domain for atomic values of the relation schema. Semantically, that amounts to specifying the set $\mathcal{D}_o$ of a modal

model. Proof-theoretically, it amounts to saying what the individual object constant symbols of the formal language are. (I'll generally assume that constant symbols of the language can also occur in models, and designate themselves.)

Next, we must specify what the attributes for the relation schema are. This amounts to specifying the set $\mathcal{D}_c$ of a model, or the set of individual concept constant symbols of a language.

Finally, we must impose some constraints, such as requiring that some attribute or set of attributes be a key, or that various functional dependencies must hold. This corresponds to taking appropriate relative rigidity formulas as axioms.

6 A Simple Example

In this section I'll show how a simple, standard, example translates into modal language both semantically and proof-theoretically. Consider the relation schema with five attributes: NAME, SSN, BIRTHDATE, JOBNUMBER, and JOBTITLE. It will be assumed that SSN is the primary key, and JOBNUMBER is functionally dependent on JOBTITLE.

We set up a formal language with "NAME," "SSN," "BIRTHDATE," "JOBNUMBER," and "JOBTITLE" as individual concept constant symbols. Then we adopt the following two *constraint axioms*.

1. $\Box(\forall\alpha)\langle\lambda x, y.\Box[x =\!\downarrow\texttt{SSN} \supset y =\!\downarrow\alpha]\rangle(\downarrow\texttt{SSN}, \downarrow\alpha)$
2. $\Box\langle\lambda x, y.\Box[x =\!\downarrow\texttt{JOBTITLE} \supset y =\!\downarrow\texttt{JOBNUMBER}]\rangle(\downarrow\texttt{JOBTITLE}, \downarrow\texttt{JOBNUMBER})$

Table 1 displays a particular relation instance of this relation schema.

NAME	SSN	BIRTHDATE	JOBNUMBER	JOBTITLE
Adam	1	01/06/-4004	1	Gardener
Eve	2	01/08/-4004	2	Explorer
Cain	3	10/21/-4004	1	Gardener
Abel	4	11/05/-4003	2	Shepherd
Seth	5	02/04/-3983	2	Explorer

Table 1. The relation PERSONS

To represent this relation instance as a modal axiomatic theory, we add the following to the constraint axioms above; we call them *instance axioms*.

3. $\Diamond[(\downarrow\texttt{NAME} = \texttt{Adam}) \wedge (\downarrow\texttt{SSN} = 1) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{01/06/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})]$
4. $\Diamond[(\downarrow\texttt{NAME} = \texttt{Eve}) \wedge (\downarrow\texttt{SSN} = 2) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{01/08/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]$
5. $\Diamond[(\downarrow\texttt{NAME} = \texttt{Cain}) \wedge (\downarrow\texttt{SSN} = 3) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{10/03/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})]$

6. $\Diamond[(\downarrow\texttt{NAME} = \texttt{Abel}) \wedge (\downarrow\texttt{SSN} = \texttt{4}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{08/05/-4003}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{2}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Shepherd})]$
7. $\Diamond[(\downarrow\texttt{NAME} = \texttt{Seth}) \wedge (\downarrow\texttt{SSN} = \texttt{5}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{02/04/-3983}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{2}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]$

This, of course, assumes individual object constant symbols, "01/06/-4004," "Adam," and so on have been added to the language. I'll also assume these symbols are intended to designate distinct objects. This gives us a (long) list of axioms.

8. $\neg(\texttt{1} = \texttt{2})$, $\neg(\texttt{Adam} = \texttt{Eve})$, etc.

Corresponding to this, semantically, we have the following **S5** model. There are five possible worlds, one for each of the five rows of Table 1; call them $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$, and Γ_5. $\mathcal{D}_o = \{\texttt{Adam}, \texttt{1}, \texttt{01/06/-4004}, \dots\}$. $\mathcal{D}_c = \{\widehat{\texttt{NAME}}, \widehat{\texttt{SSN}}, \widehat{\texttt{BIRTHDATE}}, \widehat{\texttt{JOBNUMBER}}, \widehat{\texttt{JOBTITLE}}\}$, where $\widehat{\texttt{NAME}}$ is the function that maps Γ_1 to Adam, Γ_2 to Eve, and so on. $\mathcal{I}(\texttt{Adam}) = \texttt{Adam}$, $\dots$, $\mathcal{I}(\texttt{NAME}) = \widehat{\texttt{NAME}}$, and so on.

Finally, here are some sample queries, in modal language. First, who are the explorers? This corresponds to the following, in which appropriate keys (social security numbers) are desired.

$$\langle\lambda x.\Diamond[(\downarrow\texttt{SSN} = x) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]\rangle \tag{5}$$

Suppose we abbreviate formula (5) by Q. Then, on the one hand, $Q(z)$ is true in the modal model we constructed just in case z is 2 or 5. On the other hand, $Q(z)$ is provable in the axiom system we constructed just in case z is 2 or 5.

Here is a second sample query: is there more than one gardener?

$$\begin{aligned}(\exists x)(\exists y)\{&\Diamond[(\downarrow\texttt{SSN} = x) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})] \wedge \\ &\Diamond[(\downarrow\texttt{SSN} = y) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})] \wedge \\ &\neg(x = y)\}\end{aligned} \tag{6}$$

Formula (6) is derivable from our axioms, and true in our model.

7 Connections

In the example of the previous section we saw that being a consequence of certain axioms and being true in a particular model could coincide. Now we examine to what extent this is generally the case.

Suppose we have a relation schema R and a corresponding set of constraints concerning keys and functional dependencies. Associated with R is a set of *constraint axioms*, which I'll denote $\mathsf{axiom}(R)$, consisting of formulas like (2), (3), and (4). I'll omit an exact definition—axioms 1 and 2 of the example in the previous section is a sufficient guide. Note that these axioms are either quantifier free, or else involve just universal quantifiers, and $\Box$ is the only modal operator.

Next, suppose we have a relation instance r of the relation schema R—a particular set of tuples. Associated with this is a set of *instance axioms*, all of which are quantifier free. Again I omit an exact definition, but axioms 3–7 of the previous section illustrate the notion sufficiently. Finally there are *distinctness axioms*, as in axiom 8 of the previous section. By $\mathsf{axiom}(r)$ I mean the collection of these instance axioms and distinctness axioms, together with the members of $\mathsf{axiom}(R)$. Clearly, to say r is an instance of R and satisfies the constraints, is just to say $\mathsf{axiom}(r)$ is a consistent set of model axioms.

Next, we want a designated modal model to correspond to relation instance r. Again, the example of the previous section serves as a guide. We want the model, denote it by $\mathsf{model}(r)$, meeting the following conditions. The collection of possible worlds $\mathcal{G}$ is the collection of tuples in r. The domain $\mathcal{D}_o$ of individual objects is just the collection of table entries in r. The domain $\mathcal{D}_c$ of individual concepts is the collection of attributes of relation schema R. The interpretation $\mathcal{I}$ maps each table entry (as a constant of the modal language) to itself (as a member of $\mathcal{D}_o$). And $\mathcal{I}$ maps each attribute `ATT` to the function whose value at tuple (possible world) Γ is the entry in the tuple Γ corresponding to `ATT`. The only relation symbol is $=$, which is interpreted as equality on $\mathcal{D}_o$.

Question: what are the connections between $\mathsf{axiom}(r)$ and $\mathsf{model}(r)$? I don't know the most general answer to this, but here is something that accounts for what was seen in the previous section. Note that the queries considered there, formulas (5) and (6), were either quantifier free or involved existential quantifiers of individual object type only. This is significant.

Definition 9. Call a closed modal formula *simple existential* if it is of the form

$$(\exists x_1)\cdots(\exists x_n)\Diamond\Phi$$

where Φ is quantifier and modality free, and contains only $=$ as a relation symbol.

Proposition 10. *For a relation instance r and a simple existential sentence X, X is a consequence of $\mathsf{axiom}(r)$ if and only if X is true in $\mathsf{model}(r)$.*

I'll postpone a proof of this to Section 12.3.

8 Partial Concepts

We have taken individual concepts to be *total* functions on the set of possible worlds of a modal model. But there are many circumstances where a more general notion is desirable. "The King of France," for instance, designates an individual at many points in history, but not at all of them. Fortunately, it is straightforward to allow partiality.

Definition 3, of modal model, is changed as follows. From now on $\mathcal{D}_c$ will be a non-empty set of functions, each from some subset of $\mathcal{G}$ to $\mathcal{D}_o$, where that subset can be proper. If Γ is not in the domain of some individual concept f, we will write $f(\Gamma) = \perp$, where $\perp$ is an arbitrary item not a member of $\mathcal{D}_o$. Definition 5,

value at, can be used with no change in wording, but notice that the scope of part 3 has been extended. If $\downarrow t$ is a relativized term, and Γ is not in the domain of $(v * \mathcal{I})(t)$, then $(v * \mathcal{I})(\downarrow t, \Gamma) = (v * \mathcal{I})(t)(\Gamma) = \perp$.

Now Definition 6, truth in a model, must also be modified. I'll follow the pretty obvious general principle that one cannot ascribe properties to what is designated by a non-designating term. In the Definition, item 2 is changed to read as follows.

2. If $R(t_1, \ldots, t_k)$ is atomic,
   (a) if any of $(v*\mathcal{I})(t_1, \Gamma), \ldots, (v*\mathcal{I})(t_k, \Gamma)$ is $\perp$ then $\mathcal{M}, \Gamma \nVdash_v R(t_1, \ldots, t_k)$;
   (b) otherwise $\mathcal{M}, \Gamma \Vdash_v R(t_1, \ldots, t_k)$ iff $\langle (v * \mathcal{I})(t_1, \Gamma), \ldots, (v * \mathcal{I})(t_k, \Gamma)\rangle \in \mathcal{I}(R)(\Gamma)$.

Also item 8 must be changed.

8. For an abstract $\langle \lambda\varrho.\Phi\rangle(t)$,
   (a) if $(v * \mathcal{I})(t, \Gamma) = \perp$, $\mathcal{M}, \Gamma \nVdash_v \langle \lambda\varrho.\Phi\rangle(t)$;
   (b) otherwise $\mathcal{M}, \Gamma \Vdash_v \langle \lambda\varrho.\Phi\rangle(t)$ if $\mathcal{M}, \Gamma \Vdash_v \Phi[\varrho/d]$, where $d = (v * \mathcal{I})(t, \Gamma)$.

Notice that, with the definitions modified in this way, $\downarrow t =\downarrow t$ is true at a world of a model if and only if the term t "designates" at that world. This makes possible the following piece of machinery.

Definition 11 (Designation Formula). D abbreviates the abstract $\langle \lambda\alpha.\ \downarrow \alpha =\downarrow\alpha\rangle$.

Clearly $\mathcal{M}, \Gamma \Vdash_v \mathbf{D}(t)$ iff $(v * \mathcal{I})(t, \Gamma) \neq \perp$, where $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I}\rangle$. Thus **D** really does express the notion of designation.

Now our earlier notions of rigidity and relative rigidity must be modified. We say c is rigid if it designates the same thing in all worlds in which it designates at all. This means formula (1) must be replaced with the following, if we want to express a notion of rigidity allowing for partial concepts.

$$\mathbf{D}(c) \supset \langle \lambda x.\Box[\mathbf{D}(c) \supset (x =\downarrow c)]\rangle(\downarrow c) \tag{7}$$

Likewise, c being rigid relative to d must be modified. It should now say: if d designates in two worlds, and designates the same thing, then if c also designates in those worlds, it must designate the same thing at both. This means formula (2) must be replaced with the following.

$$(\mathbf{D}(c) \wedge \mathbf{D}(d)) \supset \langle \lambda x, y.\Box[(\mathbf{D}(c) \wedge \mathbf{D}(d) \wedge x =\downarrow d) \supset (y =\downarrow c)]\rangle(\downarrow d, \downarrow c) \tag{8}$$

Similar changes must be made to the other notions from Section 4. In particular, (4), expressing that every attribute is rigid relative to c, gets expressed as follows.

$$(\forall\alpha)\{(\mathbf{D}(c) \wedge \mathbf{D}(\alpha)) \supset \langle \lambda x, y.\Box[(\mathbf{D}(c) \wedge \mathbf{D}(\alpha) \wedge x =\downarrow c) \supset (y =\downarrow\alpha)]\rangle(\downarrow c, \downarrow\alpha)\} \tag{9}$$

9 Relational Databases More Generally

A relational database generally has more than one relation involved. Now that partial individual concepts are available, this is easy to handle. Suppose we add to the database containing the relation given in Table 1 a second relation, given in Table 2.

JOBNUMBER	WHERE
1	Home
2	Away

Table 2. The relation LOCATION

We allowed, in our modal language, relation symbols of type $\langle\rangle$. Let us introduce two specific ones, PERSONS and LOCATION, intended to distinguish the two relations we now have. The idea is, we will have two kinds of possible worlds, those at which LOCATION is true and those at which PERSONS is true. The first kind of world should correspond to a row of the LOCATION table, and so JOBNUMBER and WHERE should be defined, but nothing else. Similarly for the second kind. This gives us the following new kinds of constraint axioms.

1. $\Box\{\texttt{PERSONS} \supset [\mathbf{D}(\texttt{JOBNUMBER}) \wedge \neg\mathbf{D}(\texttt{WHERE}) \wedge \mathbf{D}(\texttt{NAME}) \wedge \mathbf{D}(\texttt{SSN}) \wedge\mathbf{D}(\texttt{BIRTHDATE}) \wedge \mathbf{D}(\texttt{JOBTITLE})]\}$
2. $\Box\{\texttt{LOCATION} \supset [\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge \neg\mathbf{D}(\texttt{NAME}) \wedge \neg\mathbf{D}(\texttt{SSN}) \wedge\neg\mathbf{D}(\texttt{BIRTHDATE}) \wedge \neg\mathbf{D}(\texttt{JOBTITLE})]\}$

We still want the functional dependencies we had before, but these need to be modified to take partiality of intensional concepts into account. We also want a new dependency saying WHERE is functionally dependent on JOBNUMBER. These take the following form.

3. $\Box(\forall\alpha)\{(\mathbf{D}(\texttt{SSN}) \wedge \mathbf{D}(\alpha)) \supset \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{SSN}) \wedge \mathbf{D}(\alpha) \wedge x =\!\downarrow\texttt{SSN}) \supset (y =\!\downarrow \alpha)]\rangle(\downarrow\texttt{SSN}, \downarrow\alpha)\}$
4. $\Box\{(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{JOBTITLE})) \supset \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{JOBTITLE}) \wedge x =\!\downarrow\texttt{JOBTITLE}) \supset (y =\!\downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{JOBTITLE}, \downarrow\texttt{JOBNUMBER})\}$
5. $\Box\{(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE})) \supset \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge x =\!\downarrow\texttt{WHERE}) \supset (y =\!\downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{WHERE}, \downarrow\texttt{JOBNUMBER})\}$

Next we need the instance axioms. These are quite straightforward.

6. $\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{1}) \wedge (\downarrow\texttt{WHERE} = \texttt{Home})]$
7. $\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{2}) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$

8. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Adam}) \wedge (\downarrow\texttt{SSN} = \texttt{1}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{01/06/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{1}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})]$
9. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Eve}) \wedge (\downarrow\texttt{SSN} = \texttt{2}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{01/08/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{2}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]$
10. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Cain}) \wedge (\downarrow\texttt{SSN} = \texttt{3}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{10/03/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{1}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})]$
11. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Abel}) \wedge (\downarrow\texttt{SSN} = \texttt{4}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{08/05/-4003}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{2}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Shepherd})]$
12. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Seth}) \wedge (\downarrow\texttt{SSN} = \texttt{5}) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{02/04/-3983}) \wedge (\downarrow\texttt{JOBNUMBER} = \texttt{2}) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]$

Finally we assume that all our object constant symbols are distinct.

13. $\neg(\texttt{1} = \texttt{2})$, $\neg(\texttt{Home} = \texttt{Away})$, $\neg(\texttt{Adam} = \texttt{Eve})$, etc.

10 Tableaus

Since consequence issues are important, a sound and complete proof procedure can be useful. Fortunately, standard tableau methods using prefixed formulas adapt quite naturally.

Proofs, and derivations, will be of closed formulas. As usual, in order to handle existential quantifiers, *parameters* will be introduced. We can think of these as being additional constant symbols, added to the language for the purpose of proof construction. Since we have two kinds of quantifiers, object and concept, we will have two kinds of parameters as well. I'll use p^o, q^o, etc. as object parameters, and p^c, q^c, etc. as concept parameters.

A *prefix* for **S5** is simply a positive integer, which we can intuitively think of as the name of a possible world in some model. Unlike in more conventional treatments of modal logic, we must allow not only formulas, but also certain terms to have prefixes. For example, if c is an individual concept constant symbol, its designation in a model will vary from world to world. Think of c with prefix n as the individual object that c designates at the world named by n. To keep notation simple, I'll violate the literal meaning of the word "prefix," and display them as subscripts. Thus c_n is an example of a prefixed concept constant symbol. In our proofs, individual concept constants and individual concept parameters may have prefixes.

A little more formally, by an *extended formula* I mean one that may contain parameters, and in which individual concept constants and parameters may have prefixes. A prefixed concept constant or parameter is considered to be an individual *object* term. All proofs will be of closed formulas, but closed *extended* formulas will appear in proofs.

A *prefixed formula* is a closed extended formula, with a prefix, and here we actually write them as prefixes. Thus, if X is a closed, extended formula, and n is a positive integer, $n\,X$ is a prefixed formula.

As usual, a tableau proof of a sentence X is a tree with $1\,\neg X$ at the root, and meeting certain other conditions which we will specify. Think of the initial

prefixed formula as intuitively asserting there is a world of a model, denoted by 1, at which X is not true. The tableau is constructed using various *branch extension rules*. In them σ is an arbitrary prefix.

Conjunctive Rules

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

Double Negation Rule

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

Disjunctive Rules

$$\frac{\sigma\, X \vee Y}{\sigma\, X \,|\, \sigma\, Y} \quad \frac{\sigma\, \neg(X \wedge Y)}{\sigma\, \neg X \,|\, \sigma\, \neg Y} \quad \frac{\sigma\, X \supset Y}{\sigma\, \neg X \,|\, \sigma\, Y}$$

Necessity Rules For any positive integer n that *already occurs* on the branch,

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

Possibility Rules If the positive integer n is *new* to the branch,

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

Concept Existential Rules In the following, p^c is an individual concept parameter that is *new to the tableau branch*.

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

Object Existential Rules In the following, p^o is an individual object parameter that is *new to the tableau branch*.

$$\frac{\sigma\, (\exists x)\Phi(x)}{\sigma\, \Phi(p^o)} \quad \frac{\sigma\, \neg(\forall x)\Phi(x)}{\sigma\, \neg\Phi(p^o)}$$

Concept Universal Rules In the following, τ is any individual concept constant symbol or parameter.

$$\frac{\sigma\, (\forall\alpha)\Phi(\alpha)}{\sigma\, \Phi(\tau)} \quad \frac{\sigma\, \neg(\exists\alpha)\Phi(\alpha)}{\sigma\, \neg\Phi(\tau)}$$

Object Universal Rules In the following, τ is any individual object constant symbol or parameter, or *a prefixed individual concept constant symbol or parameter*.

$$\frac{\sigma\, (\forall x)\Phi(x)}{\sigma\, \Phi(\tau)} \quad \frac{\sigma\, \neg(\exists x)\Phi(x)}{\sigma\, \neg\Phi(\tau)}$$

Concept Abstract Rules In the following, τ is an individual concept constant symbol or parameter.

$$\frac{\sigma\, \langle \lambda\alpha.\Phi(\alpha)\rangle(\tau)}{\sigma\, \Phi(\tau)} \qquad \frac{\sigma\, \neg\langle \lambda\alpha.\Phi(\alpha)\rangle(\tau)}{\sigma\, \neg\Phi(\tau)}$$

Object Abstract Rules In the following, τ is an individual object constant symbol, parameter, or *a prefixed individual concept constant symbol or parameter.*

$$\frac{\sigma\, \langle \lambda x.\Phi(x)\rangle(\tau)}{\sigma\, \Phi(\tau)} \qquad \frac{\sigma\, \neg\langle \lambda x.\Phi(x)\rangle(\tau)}{\sigma\, \neg\Phi(\tau)}$$

Before giving the next set of abstract rules, recall that we are allowing individual concepts to be partial functions in models. If τ is an individual concept constant symbol or parameter, what is the status of $\downarrow\tau$? If we are at a world in the domain of the individual concept named by τ, $\downarrow\tau$ should be the individual object designated by that concept at that world, and otherwise it should be $\bot$. Now, if we know $\langle \lambda x.\Phi(x)\rangle(\downarrow\tau)$ is true at a world, it must be that $\downarrow\tau$ is not $\bot$ at that world, since an abstract applied to $\bot$ is false. In such a case we can introduce a name for the object designated by τ at the world; we do this by prefixing (subscripting) τ. On the other hand, if we know $\langle \lambda x.\Phi(x)\rangle(\downarrow\tau)$ is false, it could be that τ does not designate, or it could be that it does, but designates something making Φ false. In such cases we need other information to conclude whether or not $\downarrow\tau$ is $\bot$. This gives us the motivation for the following rules.

Mixed Abstract Rules In the two rules following, τ is an individual concept constant symbol or parameter.

$$\frac{\sigma\, \langle \lambda x.\Phi(x)\rangle(\downarrow\tau)}{\sigma\, \Phi(\tau_\sigma)}$$

In addition, if τ_σ already occurs on the tableau branch, the following rule may be applied.

$$\frac{\sigma\, \neg\langle \lambda x.\Phi(x)\rangle(\downarrow\tau)}{\sigma\, \neg\Phi(\tau_\sigma)}$$

Rules similar to these apply to atomic formulas as well.

Atomic Rules In the two rules following, τ is an individual concept constant symbol or parameter, R is a relation symbol, and $\ldots$ represents a sequence of terms.

$$\frac{\sigma\, R(\ldots, \downarrow\tau, \ldots)}{\sigma\, R(\ldots, \tau_\sigma, \ldots)}$$

And, if τ_σ already occurs on the tableau branch, the following rule may be applied.

$$\frac{\sigma\, \neg R(\ldots, \downarrow\tau, \ldots)}{\sigma\, \neg R(\ldots, \tau_\sigma, \ldots)}$$

Finally, we have the rules for equality. The first one corresponds to the semantic fact that equality is interpreted the same at every world; if individual objects are equal at some world, they are equal at every world.

Equality Transfer Rule If τ_1 and τ_2 are individual object constant symbols or parameters, or prefixed individual concept constant symbols or parameters, and if σ' is a prefix that already occurs on the branch

$$\frac{\sigma\,(\tau_1 = \tau_2)}{\sigma'\,(\tau_1 = \tau_2)}$$

Reflexivity Rule If τ is an individual object constant symbol, parameter, or a prefixed individual concept constant symbol or parameter, and the prefix σ already occurs on the branch,

$$\frac{}{\sigma\,(\tau = \tau)}$$

Substitutivity Rule If τ_1 and τ_2 are individual object constant symbols or parameters, or prefixed individual concept constant symbols or parameters

$$\frac{\sigma\,\Phi(\tau_1) \qquad \sigma'\,(\tau_1 = \tau_2)}{\sigma\,\Phi(\tau_2)}$$

This concludes the presentation of the branch extension rules.

Definition 12 (Closed). A tableau branch is *closed* if it contains $\sigma\,X$ and $\sigma\,\neg X$ for some X. A tableau is *closed* if every branch is closed.

Definition 13 (Proof and Derivation). A sentence Φ (without parameters) is *provable* if there is a closed tableau beginning with $1\,\neg\Phi$. Likewise, Φ has a *derivation* from a set S of sentences if there is a closed tableau beginning with $1\,\neg\Phi$, in which $1\,X$ can be added to any branch at any point, for any X that is a member of S.

This concludes the description of the tableau system.

11 A Derivation Example

The example given here is a derivation of

$$\neg\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})] \tag{10}$$

from the axioms of Section 9. It establishes that $\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$ cannot be inserted into the database, because it violates an integrity constraint. Before presenting the derivation itself, here is a derived rule that will shorten the presentation.

Derived Rule Suppose τ is an individual concept constant symbol or parameter.

$$\frac{\begin{array}{c}\sigma\ \downarrow\tau = a \\ \sigma\,\mathbf{D}(\tau) \supset \Phi\end{array}}{\sigma\,\Phi}$$

Think of this as abbreviating the following tableau construction.

$$\begin{array}{l} \sigma\ \downarrow\tau = a \quad 1. \\ \sigma\,\mathbf{D}(\tau) \supset \Phi \quad 2. \\ \sigma\,\tau_\sigma = a \quad 3. \end{array}$$

$$\begin{array}{l|l} \sigma\,\neg\mathbf{D}(\tau) \quad 4. & \sigma\,\Phi \quad 5. \\ \sigma\,\neg\langle\lambda x.\ \downarrow x = \downarrow x\rangle(\tau) \quad 6. & \\ \sigma\,\neg(\downarrow\tau = \downarrow\tau) \quad 7. & \\ \sigma\,\neg(\tau_\sigma = \tau_\sigma) \quad 8. & \\ \sigma\quad (\tau_\sigma = \tau_\sigma) \quad 9. & \end{array}$$

Explanation: 3 is from 1 by an atomic rule; 4 and 5 are from 2 by a disjunctive rule; 6 is 4 unabbreviated; 7 is from 6 by a concept abstract rule; 8 is from 7 by an atomic rule, making use of the fact that τ_σ occurs in 3; 9 is by the reflexivity rule. The left branch is closed, and the right branch gives the effect of the conclusion of the derived rule.

Now, the derivation of formula (10) is in Figure 1. The explanation of Figure 1 is as follows: 2 is from 1 by the double negation rule; 3 is from 2 by a possibility rule; 4, 5, and 6 are from 3 by repeated uses of a conjunction rule; 7 introduces axiom 5 of Section 9; 8 is from 7 by a necessity rule; 9 is from 5, 6, and 8 by repeated uses of the derived rule above; 10 is from 9 by a mixed abstract rule; 11 is from 10 by an object abstract rule; 12 is axiom 7 of Section 9; 13 is from 12 by a possibility rule; 14, 15, and 16 are from 13 by repeated conjunction rules; 17 is from 11 by a necessity rule; 18 is from 15, 16, and 17 by the derived rule above (slightly modified); 19 and 20 are from 18 by a disjunctive rule; 21 is from 16 by an atomic rule, as are 22 from 20, 23 from 19, 24 from 5, 25 from 6, and 26 from 15; 27 is from 25 by an equality transfer rule, as is 28 from 24; 29 is from 23 and 27 by a substitutivity rule, as are 30 from 22 and 28, 31 from 27 and 29, and 32 from 26 and 30; 33 is by the reflexivity rule; 34 is by axiom 13 of Section 9; and 35 is from 34 by an equality transfer rule.

As another, and simpler, example, you might try giving a derivation of the following, essentially establishing that Eve is someone who works Away.

$$\begin{array}{l}\langle\lambda x, y.(\exists z)\{\Diamond[\mathtt{PERSONS} \wedge (\downarrow\mathtt{NAME} = x) \wedge (\downarrow\mathtt{JOBNUMBER} = z)] \\ \qquad \wedge\Diamond[\mathtt{LOCATION} \wedge (\downarrow\mathtt{JOBNUMBER} = z) \wedge (\downarrow\mathtt{WHERE} = y)]\}\rangle(\mathtt{Eve}, \mathtt{Away})\end{array} \tag{11}$$

12 Completeness et. al.

In this section I'll sketch soundness and completeness arguments for the tableau system, as well as give a proof for Proposition 10. Nothing is given in much detail, because proofs are straightforward adaptations of what are, by now, fairly standard arguments.

12.1 Soundness

Soundness is by the usual tableau method. One defines a notion of *satisfiability* for prefixed formulas—a set S is satisfiable if there is a model $\mathcal{M}$, a mapping m assigning to each prefix σ a possible world $m(\sigma)$ of $\mathcal{M}$, and a formula Φ is true at world $m(\sigma)$ of $\mathcal{M}$ whenever $\sigma\,\Phi \in S$. A tableau is called satisfiable if the set of prefixed formulas on one of its branches is satisfiable. Then one shows that each tableau rule preserves tableau satisfiability. This requires a case by case check.

Now, if there is a closed tableau for $1\,\neg X$, then X must be valid. For, otherwise, there would be some model in which X was false at some world. It follows that the set $\{1\,\neg X\}$ is satisfiable, so we begin with a satisfiable tableau. Then we can only get satisfiable tableaus, and since we had a closed tableau for $1\,\neg X$ we have the impossible situation of having a closed, satisfiable tableau.

12.2 Completeness

Suppose X is a sentence that has no tableau proof—it must be shown that X is not valid. Again the methodology is standard. Also, while the proof sketch below is just for tableau provability, and not derivability, the argument extends directly. I'm giving the simpler version, for simplicity.

Begin by constructing a tableau for $1\,\neg X$, and do so *systematically*, in such a way that all tableau rules are *fairly* applied. That is, during the tableau construction, any rule that could eventually be applied is. There are many such fair tableau construction procedures—I'll leave the details to you. The result is a tableau that does not close—say it has an open branch θ (König's lemma is needed to guarantee such a branch exists, if the tableau construction is infinite).

Now, construct a model as follows.

1. The set $\mathcal{G}$ of possible worlds is the set of prefixes that occur on branch θ.
2. The domain $\mathcal{D}_o$ of objects is the set consisting of: all individual object constant symbols of the language, all individual object parameters that occur on θ, and all subscripted (prefixed) individual concept constant symbols and parameters that occur on θ.
3. if f is an individual concept constant symbol, or individual concept parameter that occurs on θ, a function $\widehat{f}$ is defined as follows. The domain of $\widehat{f}$ is the set of prefixes σ such that f_σ occurs on θ. And if σ is in the domain of $\widehat{f}$ then $\widehat{f}(\sigma) = f_\sigma$. Note that $\widehat{f}$ maps a subset of $\mathcal{G}$ to $\mathcal{D}_o$. The domain $\mathcal{D}_c$ of concepts is the set of all these $\widehat{f}$.
4. For the interpretation $\mathcal{I}$:

(a) $\mathcal{I}$ assigns to each member of $\mathcal{D}_o$ itself.
(b) $\mathcal{I}$ assigns to each f that is an individual concept constant or parameter (on θ) the function $\widehat{f}$.
(c) $\mathcal{I}$ assigns to a relation symbol P of type $\langle\rangle$ the mapping from $\mathcal{G}$ to $\{\mathit{false}, \mathit{true}\}$ such that $\mathcal{I}(P)(\sigma) = \mathit{true}$ iff $\sigma\, P$ occurs on θ.
(d) To make this clause easier to state, I'll use the following notation. If f is an object symbol, set $\widehat{f} = f$. If f is a concept constant or parameter, $\widehat{f}$ has already been defined. Now, $\mathcal{I}$ assigns to a relation symbol R of type $\langle n_1, n_2, \ldots, n_k\rangle$ a mapping on $\mathcal{G}$ such that $\langle\widehat{t_1}, \ldots, \widehat{t_k}\rangle \in \mathcal{I}(R)(\sigma)$ iff $\sigma\, R(t_1, \ldots, t_k)$ occurs on θ.

This completes the definition of a model, call it $\mathcal{M}$. Actually, the equality symbol may not be interpreted by equality, but leaving this aside for the moment, one can show by standard methods involving an induction on formula degree that, for any valuation v:

– If $\sigma\, \Phi$ occurs on θ then $\mathcal{M}, \sigma \Vdash_v \Phi$.
– If $\sigma\, \neg\Phi$ occurs on θ then $\mathcal{M}, \sigma \nVdash_v \Phi$.

The valuation v can be arbitrary because free variables do not occur in tableaus. Since $1\, \neg X$ begins the tableau, it occurs on θ, and hence X is not valid in $\mathcal{M}$ since it is false at world 1.

Finally, one "factors" the model $\mathcal{M}$ using the equivalence relation that is the interpretation of the equality symbol, turning it into a normal modal model. The Equality Transfer Rule tells us that the interpretation of the equality symbol will be the same at every world of $\mathcal{M}$. I'll leave details to you.

12.3 A Sketch of Proposition 10

Assume r is a relation instance and X a simple existential sentence X. All members $\mathsf{axiom}(r)$ are true in $\mathsf{model}(r)$, so if X is a consequence of $\mathsf{axiom}(r)$, then X will be true in $\mathsf{model}(r)$. It is the converse direction that needs work.

Suppose X is not a consequence of $\mathsf{axiom}(r)$. Then X does not have a tableau derivation from $\mathsf{axiom}(r)$. Suppose we now carry out the steps of the completeness argument, from Section 12.2. We begin a tableau for $1\, \neg X$, carry out its construction systematically and, since it is a derivation, we introduce members of $\mathsf{axiom}(r)$ onto the branch during the construction. As usual, I'll omit details.

Members of $\mathsf{axiom}(r)$ that are constraint axioms all involve $\Box$ in a positive location—they do not involve $\Diamond$. The sentence X is simple existential, and so $\neg X$ contains $\Diamond$ in a negative location—it behaves like $\Box$. None of these formulas, then, can invoke applications of a possibility rule. Only the instance axioms of $\mathsf{axiom}(r)$ can do this. So, if there are n members of $\mathsf{axiom}(r)$ that are instance axioms, an open tableau branch will have exactly $n + 1$ different prefixes on it: prefix 1 with which we started, and the n additional prefixes introduced by possibility rule applications to instance axioms.

Now, proceed with the construction of a model, using an open tableau branch, as outlined in Section 12.2. We get a model with $n+1$ worlds, with X is false at

world 1, and a world corresponding to each instance axiom. Because of the form of X (only existential quantifiers and a single possibility symbol), since it is false at world 1, it is also false at every world. If we now consider the submodel in which world 1 is dropped, it is not hard to check that truth values of members of $\mathsf{axiom}(r)$ do not change at remaining worlds, nor does the truth value of X. And the resulting model is (isomorphic to) $\mathsf{model}(r)$.

13 Conclusion

I want to finish by describing two plausible directions for future work, one having to do with the modal logic directly, the other with its applications to databases.

The tableau proof procedure given here used parameters and, as such, is meant for human application. But it should be possible to develop a free-variable version that can be automated. The **S5** modality itself is a kind of quantifier, but it is of a simple nature. Object quantification is essentially classical. Concept quantification may create some difficulty—I don't know. Equality plays a fundamental role, but it is a rather simple one. Perhaps what is needed can be captured efficiently in an automated proof procedure.

The databases considered here were all conventional relational ones. This is what the first-order modal language can handle. But one could consider multiple-valued databases, say, in which entries can be sets. Or for a more complicated example, consider this. Say a record represents a person, and among a person's attributes are these three: `FAVORITE_BOOK`, `FAVORITE_MOVIE`, and `MOST_IMPORTANT`. The first two attributes have the obvious meaning. The `MOST_IMPORTANT` attribute records which that person considers most important in evaluating someone, `FAVORITE_BOOK` or `FAVORITE_MOVIE`. Thus `MOST_IMPORTANT` is an attribute whose value is an attribute. (This example is meant to be easily described, and hence is rather artificial. More realistic examples are not hard to come by.) The modal logic of this paper is really the first-order fragment of a higher-type system, presented in full in [1]. If one uses that, one can easily have sets of objects, or attributes, as entries. Indeed, one can consider much more complex things yet. Of course the proof procedure also becomes more complex, as one would expect. Whether such things are of use remains to be seen.

References

1. Melvin C. Fitting. *Types, Tableaus, and Gödel's God.* 2000. Available on my web site: comet.lehman.cuny.edu/fitting.
2. Melvin C. Fitting and Richard Mendelsohn. *First-Order Modal Logic.* Kluwer, 1998. Paperback, 1999.
3. Saul Kripke. *Naming and Necessity.* Harvard University Press, 1980.
4. Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

$1\,\neg\neg\Diamond[\mathtt{LOCATION} \wedge (\downarrow\mathtt{JOBNUMBER} = 1) \wedge (\downarrow\mathtt{WHERE} = \mathtt{Away})]$ 1.
$1\,\Diamond[\mathtt{LOCATION} \wedge (\downarrow\mathtt{JOBNUMBER} = 1) \wedge (\downarrow\mathtt{WHERE} = \mathtt{Away})]$ 2.
$2\,\mathtt{LOCATION} \wedge (\downarrow\mathtt{JOBNUMBER} = 1) \wedge (\downarrow\mathtt{WHERE} = \mathtt{Away})$ 3.
$2\,\mathtt{LOCATION}$ 4.
$2\,(\downarrow\mathtt{JOBNUMBER} = 1)$ 5.
$2\,(\downarrow\mathtt{WHERE} = \mathtt{Away})$ 6.
$1\,\Box\{(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE})) \supset$
$\langle\lambda x, y.\Box[(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE}) \wedge x = \downarrow\mathtt{WHERE}) \supset$
$(y = \downarrow\mathtt{JOBNUMBER})]\rangle(\downarrow\mathtt{WHERE}, \downarrow\mathtt{JOBNUMBER})\}$ 7.
$2\,(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE})) \supset$
$\langle\lambda x, y.\Box[(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE}) \wedge x = \downarrow\mathtt{WHERE}) \supset$
$(y = \downarrow\mathtt{JOBNUMBER})]\rangle(\downarrow\mathtt{WHERE}, \downarrow\mathtt{JOBNUMBER})$ 8.
$2\,\langle\lambda x, y.\Box[(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE}) \wedge x = \downarrow\mathtt{WHERE}) \supset$
$(y = \downarrow\mathtt{JOBNUMBER})]\rangle(\downarrow\mathtt{WHERE}, \downarrow\mathtt{JOBNUMBER})$ 9.
$2\,\langle\lambda x, y.\Box[(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE}) \wedge x = \downarrow\mathtt{WHERE}) \supset$
$(y = \downarrow\mathtt{JOBNUMBER})]\rangle(\mathtt{WHERE}_2, \mathtt{JOBNUMBER}_2)$ 10.
$2\,\Box[(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE}) \wedge \mathtt{WHERE}_2 = \downarrow\mathtt{WHERE}) \supset$
$(\mathtt{JOBNUMBER}_2 = \downarrow\mathtt{JOBNUMBER})]$ 11.
$1\,\Diamond[\mathtt{LOCATION} \wedge (\downarrow\mathtt{JOBNUMBER} = 2) \wedge (\downarrow\mathtt{WHERE} = \mathtt{Away})]$ 12.
$3\,[\mathtt{LOCATION} \wedge (\downarrow\mathtt{JOBNUMBER} = 2) \wedge (\downarrow\mathtt{WHERE} = \mathtt{Away})]$ 13.
$3\,\mathtt{LOCATION}$ 14.
$3\,(\downarrow\mathtt{JOBNUMBER} = 2)$ 15.
$3\,(\downarrow\mathtt{WHERE} = \mathtt{Away})$ 16.
$3\,[(\mathbf{D}(\mathtt{JOBNUMBER}) \wedge \mathbf{D}(\mathtt{WHERE}) \wedge \mathtt{WHERE}_2 = \downarrow\mathtt{WHERE}) \supset$
$(\mathtt{JOBNUMBER}_2 = \downarrow\mathtt{JOBNUMBER})]$ 17.
$3\,[(\mathtt{WHERE}_2 = \downarrow\mathtt{WHERE}) \supset$
$(\mathtt{JOBNUMBER}_2 = \downarrow\mathtt{JOBNUMBER})]$ 18.

$3\,\neg(\mathtt{WHERE}_2 = \downarrow\mathtt{WHERE})$ 19.
$3\;\;(\mathtt{WHERE}_3 = \mathtt{Away})$ 21.
$3\,\neg(\mathtt{WHERE}_2 = \mathtt{WHERE}_3)$ 23.
$2\;\;(\mathtt{WHERE}_2 = \mathtt{Away})$ 25.
$3\;\;(\mathtt{WHERE}_2 = \mathtt{Away})$ 27.
$3\,\neg(\mathtt{Away} = \mathtt{WHERE}_3)$ 29.
$3\,\neg(\mathtt{Away} = \mathtt{Away})$ 31.
$1\;\;(\mathtt{Away} = \mathtt{Away})$ 33.

$3\;\;(\mathtt{JOBNUMBER}_2 = \downarrow\mathtt{JOBNUMBER})$ 20.
$3\;\;(\mathtt{JOBNUMBER}_2 = \mathtt{JOBNUMBER}_3)$ 22.
$2\;\;(\mathtt{JOBNUMBER}_2 = 1)$ 24.
$3\;\;(\mathtt{JOBNUMBER}_3 = 2)$ 26.
$3\;\;(\mathtt{JOBNUMBER}_2 = 1)$ 28.
$3\;\;(1 = \mathtt{JOBNUMBER}_3)$ 30.
$3\;\;(1 = 2)$ 32.
$1\,\neg(1 = 2)$ 34.
$3\,\neg(1 = 2)$ 35.

Fig. 1. Derivation Example