Databases and Higher Types

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. Generalized databases will be examined, in which attributes can be sets of attributes, or sets of sets of attributes, and other higher type constructs. A precise semantics will be developed for such databases, based on a higher type modal/intensional logic.

1 Introduction

In some ways this is an eccentric paper—there are no theorems. What I want to do, simply stated, is present a semantics for relational databases. But the semantics is rich, powerful, and oddly familiar, and applies to databases that are quite general. It is a topic whose exploration I wish to recommend, rather than a finished product I simply present.

Relational databases generally have entities of some kind as values of attributes, though it is a small stretch to allow sets of entities as well. I want to consider databases that stretch things further, allowing attributes to have as values sets of sets of entities, and so on, but further, I also want to allow sets of attributes, sets of sets of attributes, and so on. There are quite reasonable examples showing why one might find such things desirable, at least at low levels, and a very simple one will be given below.

It is not enough to just allow eccentric attribute values—a semantics must also be supplied to give them meaning. And rather than looking to some version of classical logic, I will show that modal logic provides a very natural tool. Of course it must be higher type modal logic, to encompass the kinds of things I have been talking about. I will use the one presented in [3], which mildly generalizes work of Montague [8–10] and Gallin [5].

This paper is a sequel to [2], in which a modal/intensional approach to databases is developed in some detail at the first-order level. Once a full hierarchy of types is introduced things become complex, and no more than a sketch can be presented here. In particular, though a tableau system exists for the modal logic I use, it will not be discussed in this paper.

It may be of interest that I did not get into this line of work from the database side. I began with attempts to treat various classic philosophical problems as simply as possible in a modal context—work culminating in [4]. This, in turn, led to an interest in higher type modal logics, connected with a desire to understand Gödel's ontological argument, [6]. My work on this can be found in [3]. Databases

came in, unnoticed, by a side door. But they are at the party, and it may be they will have a good time.

2 A Sample Database

In order to illustrate the higher-type constructs, I'll create a miniature database of some complexity. I will take ground-level entities to be strings. Let's say the Locomobile Company[1] still exists, and manufactures cars, motorcycles, and pianos. Table 1 shows the start of a database—more attributes will be added later.

IDNumber	Item	Cylinders	Engine	Colors	Air	Config
1	automobile	2	{A, B}	{red, green, black}	{no}	$\perp$
2	automobile	4	{A}	{green, black}	{yes, no}	$\perp$
3	motorcycle	2	{C, D}	{blue, black}	$\perp$	$\perp$
4	piano	$\perp$	$\perp$	$\perp$	$\perp$	{upright, grand}

Table 1. Locomobile Sales List

Notice that in Table 1 some of the attributes have values that are ground objects—`Cylinders`, say—while some are sets of ground objects—`Engine` types, for instance. An entry of $\perp$ indicates an attribute that is not relevant to the particular item.

In the table above, let us say that for the 2 cylinder automobile the choice of engine type, `A` or `B`, is not up to the buyer, since both are functionally equivalent. But the choice of `Colors`, naturally, would be up to the customer. Similarly for the 4 cylinder. But let's say that for the motorcycle, the engine type is something the customer chooses. Then let us have an additional attribute telling us, for each record, which (other) attributes are customer chosen. Rather than repeating the whole table, I'll just give this single additional attribute in Table 2.

Notice that in Table 2, the `Customer` attribute has as values sets of attributes. Finally, many of the attributes for an item can be irrelevant, as has been indicated by $\perp$. Rather than explicitly having an 'undefined' value in our semantics, instead let us add additional attributes telling us which of the other attributes are relevant.

Values of the `Relevant0` attribute, in Table 3, are sets of attributes whose values are ground level objects, values of `Relevant1` are sets of attributes whose values are sets of ground level objects, and values of `Relevant2` are sets of

[1] The actual company was founded in 1899 to manufacture steam powered cars. It moved to luxury internal combustion automobiles in 1902, and went into receivership in 1922. When active, they manufactured four cars a day.

IDNumber	Customer
1	{Colors}
2	{Colors, Air}
3	{Engine, Colors}
4	{Configuration}

Table 2. Locomobile Customer Attribute

IDNumber	Relevant0	Relevant1	Relevant2
1	{IDNumber, Item, Cylinders}	{Engine, Colors, Air}	{Customer}
2	{IDNumber, Item, Cylinders}	{Engine, Colors, Air}	{Customer}
3	{IDNumber, Item, Cylinders}	{Engine, Colors}	{Customer}
4	{IDNumber, Item}	{Configuration}	{Customer}

Table 3. Locomobile Relevancy Attribute

attributes whose values are sets of attributes whose values are sets of ground level objects.

Finally, all this is really an instance of a relation schema, and that schema has a few constraints which I've implicitly been obeying. Clearly `IDNumber` is a key attribute. Also, an attribute belongs to the `Relevant0`, `Relevant1`, or `Relevant2` attribute of a record if and only if the attribute is defined for that record, that is, has a value other than $\perp$. I'll come back to the notion of constraint later on.

3 Higher Order Modal Logic

Shifting gears abruptly (something the Locomobile did smoothly) I now present a sketch of a higher order modal logic, taken from [3], and derived from [11] via [5]. The machinery is somewhat complex, and space here is limited. See [3] for a fuller discussion of underlying ideas.

I'll start with the notion of types. The key feature here is that there are both intensional and extensional types.

Definition 1. The notion of a type, extensional and intensional, is given as follows.

1. 0 is an *extensional type*.
2. If $t_1, \ldots, t_n$ are types, extensional or intensional, $\langle t_1, \ldots, t_n \rangle$ is an *extensional type*.
3. If t is an extensional type, $\uparrow t$ is an *intensional type*.

A *type* is an intensional or an extensional type.

As usual, 0 is the type of ground-level objects, unanalyzed "things." The type $\langle t_1, \ldots, t_n \rangle$ is for n-ary relations in the conventional sense, where the components are of types $t_1, \ldots, t_n$ respectively. The type $\uparrow t$ is the unfamiliar piece of machinery—it will be used as the type of an intensional object which, in a particular context, determines an extensional object of type t. All this will be clearer once models have been presented.

For each type t I'll assume there are infinitely many variable symbols of that type. I'll also assume there is a set C constant symbols, containing at least an equality symbol $=^{\langle t,t \rangle}$ for each type t. I denote the higher-order language built up from C by $L(C)$. I'll indicate types, when necessary, by superscripts, as I did with equality above.

In formulating a higher order logic one can use comprehension axioms, or one can use explicit term formation machinery, in effect building comprehension into the language. I'll follow the later course, but this means terms cannot be defined first, and then formulas. Instead they must be defined in a mutual recursion. Most of the items below are straightforward, but a few need comment. First, concerning the term formation machinery mentioned above, predicate abstraction, it should be noted that $\langle \lambda\alpha_1, \ldots, \alpha_n.\Phi \rangle$ is taken to be a term of *intensional* type. Its meaning can vary from world to world, simply because the behavior of the formula Φ changes from world to world. Second, there is a new piece of machinery, $\downarrow$, mapping intensional terms to extensional ones. Think of it as the "extension of" operator—at a possible world it supplies the extension there for an intensional term.

Definition 2. Terms and formulas of $L(C)$ are defined as follows.

1. A constant symbol or variable of $L(C)$ of type t is a term of $L(C)$ of type t. If it is a constant symbol, it has no free variable occurrences. If it is a variable, it has one free variable occurrence, itself.
2. If Φ is a formula of $L(C)$ and $\alpha_1, \ldots, \alpha_n$ is a sequence of distinct variables of types $t_1, \ldots, t_n$ respectively, then $\langle \lambda\alpha_1, \ldots, \alpha_n.\Phi \rangle$ is a term of $L(C)$ of the intensional type $\uparrow\langle t_1, \ldots, t_n \rangle$. It is called a predicate abstract, and its free variable occurrences are the free variable occurrences of Φ, except for occurrences of the variables $\alpha_1, \ldots, \alpha_n$.
3. If τ is a term of $L(C)$ of type $\uparrow t$ then $\downarrow\tau$ is a term of type t. It has the same free variable occurrences that τ has.
4. If τ is a term of either type $\langle t_1, \ldots, t_n \rangle$ or type $\uparrow\langle t_1, \ldots, t_n \rangle$, and $\tau_1, \ldots, \tau_n$ is a sequence of terms of types $t_1, \ldots, t_n$ respectively, then $\tau(\tau_1, \ldots, \tau_n)$ is a formula (atomic) of $L(C)$. The free variable occurrences in it are the free variable occurrences of $\tau, \tau_1, \ldots, \tau_n$.
5. If Φ is a formula of $L(C)$ so is $\neg\Phi$. The free variable occurrences of $\neg\Phi$ are those of Φ.
6. If Φ and Ψ are formulas of $L(C)$ so is $(\Phi \wedge \Psi)$. The free variable occurrences of $(\Phi \wedge \Psi)$ are those of Φ together with those of Ψ.
7. If Φ is a formula of $L(C)$ and α is a variable then $(\forall\alpha)\Phi$ is a formula of $L(C)$. The free variable occurrences of $(\forall\alpha)\Phi$ are those of Φ, except for occurrences of α.

8. If Φ is a formula of $L(C)$ so is $\Box\Phi$. The free variable occurrences of $\Box\Phi$ are those of Φ.

Other connectives, quantifiers, and modal operators have the usual definitions.

The next thing is semantics. Actually, the only modal logic I'll need will be **S5**, for which the accessibility relation, $\mathcal{R}$, is an equivalence relation, but it does no harm to present the general case now. Note that the ground-level domain, $\mathcal{D}$, is not world dependent—in effect, type-0 quantification is *possibilist* and not *actualist*.

Definition 3. An *augmented Kripke frame* is a structure $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ where $\mathcal{G}$ is a non-empty set (of *possible worlds*), $\mathcal{R}$ is a binary relation on $\mathcal{G}$ (called *accessibility*) and $\mathcal{D}$ is a non-empty set, the (ground-level) *domain*.

Next I say what the objects of each type are, relative to a choice of ground-level domain and set of possible worlds. In classical higher order logic, *Henkin models* are standard. In these, rather than having all objects of higher types, one has "enough" of them. It is well-known that a restriction to "true" higher order classical models gives a semantics that is not axiomatizable, while Henkin models provide an axiomatizable version. A similar thing happens here, but the definition of the modal analog of Henkin models is fairly complex, because saying what it means to have "enough" objects requires serious effort. I will just give the "true" model version—the Henkin generalization can be found in [3]. But I also note that, in applications to databases, ground level domains will often be finite.

Definition 4. Let $\mathcal{G}$ be a non-empty set (of possible worlds) and let $\mathcal{D}$ be a non-empty set (the ground-level domain). For each type t, the collection $[\![t, \mathcal{D}, \mathcal{G}]\!]$, of objects of type t with respect to $\mathcal{D}$ and $\mathcal{G}$, is defined as follows ($\mathcal{P}$ is the powerset operator).

1. $[\![0, \mathcal{D}, \mathcal{G}]\!] = \mathcal{D}$.
2. $[\![\langle t_1, \ldots, t_n \rangle, \mathcal{D}, \mathcal{G}]\!] = \mathcal{P}([\![t_1, \mathcal{D}, \mathcal{G}]\!] \times \cdots \times [\![t_n, \mathcal{D}, \mathcal{G}]\!])$.
3. $[\![\uparrow t, \mathcal{D}, \mathcal{G}]\!] = [\![t, \mathcal{D}, \mathcal{G}]\!]^{\mathcal{G}}$.

O is an *object of type* t if $O \in [\![t, \mathcal{D}, \mathcal{G}]\!]$. O is an *intensional* or *extensional* object according to whether its type is intensional or extensional.

Now the terminology should be a little clearer. If O is extensional, it is a relation in the conventional sense. If O is intensional, it is a mapping that assigns an object to each possible world, that is, its designation can vary from state to state. Next we move to models, and remember, these are "true" models, and not a Henkin version. Much of this looks quite technical, but it reflects reasonable intuitions and, in fact, an intuitive understanding will be sufficient for this paper.

Definition 5. A *model for the language* $L(C)$ is a structure $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I} \rangle$, where $\langle \mathcal{G}, \mathcal{R}, \mathcal{D} \rangle$ is an augmented frame and $\mathcal{I}$ is an *interpretation*, which meets the following conditions.

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

Definition 6. A mapping v is a *valuation* in the model $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I}\rangle$ if v assigns to each variable α of type t some object of type t, that is, $v(\alpha) \in [\![t, \mathcal{D}, \mathcal{G}]\!]$. An α *variant* of v is a valuation that agrees with v on all variables except α. Similarly for $\alpha_1, \ldots, \alpha_n$ *variant*.

Finally, designation of a term, and truth of a formula, are defined by a simultaneous recursion.

Definition 7. Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{R}, \mathcal{D}, \mathcal{I}\rangle$ be a model, let v be a valuation in it, and let $\Gamma \in \mathcal{G}$ be a possible world. A mapping $(v * \mathcal{I} * \Gamma)$, assigning to each term an object that is the designation of that term at Γ is defined, and a relation $\mathcal{M}, \Gamma \Vdash_v \Phi$ expressing truth of Φ at possible world Γ are characterized as follows.

1. If A is a constant symbol of $L(C)$ then $(v * \mathcal{I} * \Gamma)(A) = \mathcal{I}(A)$.
2. If α is a variable then $(v * \mathcal{I} * \Gamma)(\alpha) = v(\alpha)$.
3. If τ is a term of type $\uparrow t$ then $(v * \mathcal{I} * \Gamma)(\downarrow\tau) = (v * \mathcal{I} * \Gamma)(\tau)(\Gamma)$
4. If $\langle \lambda\alpha_1, \ldots, \alpha_n.\Phi\rangle$ is a predicate abstract of $L(C)$ of type $\uparrow\langle t_1, \ldots, t_n\rangle$, then $(v * \mathcal{I} * \Gamma)(\langle \lambda\alpha_1, \ldots, \alpha_n.\Phi\rangle)$ is an intensional object; it is the function that assigns to an arbitrary world Δ the following member of $[\![\langle t_1, \ldots, t_n\rangle, \mathcal{D}, \mathcal{G}]\!]$:

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

5. For an atomic formula $\tau(\tau_1, \ldots, \tau_n)$,
   (a) If τ is of an intensional type, $\mathcal{M}, \Gamma \Vdash_v \tau(\tau_1, \ldots, \tau_n)$ provided $\langle (v * \mathcal{I} * \Gamma)(\tau_1), \ldots, (v * \mathcal{I} * \Gamma)(\tau_n)\rangle \in (v * \mathcal{I} * \Gamma)(\tau)(\Gamma)$.
   (b) If τ is of an extensional type, $\mathcal{M}, \Gamma \Vdash_v \tau(\tau_1, \ldots, \tau_n)$ provided $\langle (v * \mathcal{I} * \Gamma)(\tau_1), \ldots, (v * \mathcal{I} * \Gamma)(\tau_n)\rangle \in (v * \mathcal{I} * \Gamma)(\tau)$.
6. $\mathcal{M}, \Gamma \Vdash_v \neg\Phi$ if it is not the case that $\mathcal{M}, \Gamma \Vdash_v \Phi$.
7. $\mathcal{M}, \Gamma \Vdash_v \Phi \wedge \Psi$ if $\mathcal{M}, \Gamma \Vdash_v \Phi$ and $\mathcal{M}, \Gamma \Vdash_v \Psi$.
8. $\mathcal{M}, \Gamma \Vdash_v (\forall\alpha)\Phi$ if $\mathcal{M}, \Gamma \Vdash_{v'} \Phi$ for every α-variant v' of v.
9. $\mathcal{M}, \Gamma \Vdash_v \Box\Phi$ if $\mathcal{M}, \Delta \Vdash_v \Phi$ for all $\Delta \in \mathcal{G}$ such that $\Gamma\mathcal{R}\Delta$.

4 A Modal Interpretation

So far two separate topics, databases and modal logic, have been discussed. It is time to bring them together. I'll show how various database concepts embed naturally into a modal setting. Think of the database as having an informal semantics, and the embedding into modal logic as supplying a precise, formal version.

First of all, think of a record in a database as a possible world in a modal model. This is not at all far-fetched—conceptually they play similar roles. When dealing with databases, records are behind the scenes but are not first-class objects. That is, an answer to a query might be a record *number*, but it will

not be a record. In a modal logic possible worlds have a similar role—they are present in the semantics, but a modal language does not refer to them directly.

There is no reason to assume some records outrank others, whatever that might mean, so I'll take the accessibility relation to be the one that always holds. This means our modal operators are those of **S5**.

In the little Locomobile database considered earlier, ground-level objects were strings. I'll carry that over to the modal setting now—the ground level domain will consist of strings. Clearly this choice is not a critical issue.

Attributes are a key item in interpreting a database modally. Fortunately there is a natural counterpart. An attribute assigns to each record some entity of an appropriate kind. In a modal model, an interpreted constant symbol of intensional type assigns to each possible world an object of an appropriate type. I'll simply provide an intensional constant symbol for each attribute, and interpret it accordingly.

By way of illustration, let's create a modal language and model corresponding to the particular database presented in Section 2. It is an example that is sufficiently general to get all the basic ideas across.

To specify the language, it is enough to specify the set C of constant symbols, and their respective types. These will be ground level strings, which give us type 0 constant symbols, and various attributes, which give us intensional constant symbols of various types. The strings from the Locomobile example are `1`, `2`, `3`, `4`, `automobile`, `motorcycle`, `piano`, `A`, `B`, `C`, `D`, `red`, `green`, `black`, `blue`, `yes`, `no`, `upright`, `grand`, all of which are taken as type 0 constant symbols. The attributes provide the following higher type constant symbols: `IDNumber`, `Item`, and `Cylinders`, all of type $\uparrow 0$; `Engine`, `Colors`, `Air`, and `Config`, all of type $\uparrow\langle 0\rangle$; `Customer`, of type $\uparrow\langle\uparrow\langle 0\rangle\rangle$; `Relevant0`, of type $\uparrow\langle\uparrow 0\rangle$; `Relevant1`, of type $\uparrow\langle\uparrow\langle 0\rangle\rangle$; and `Relevant2`, of type $\uparrow\langle\uparrow\langle\uparrow\langle 0\rangle\rangle\rangle$.

Now that we have our modal language, $L(C)$, the next job is to create a specific modal model, corresponding to the Locomobile tables.

Let $\mathcal{G}$ be the set $\{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4\}$, where the intention is that each of these corresponds to one of the four records in the database given in Section 2. Specifically, Γ_i corresponds to the record with an `IDNumber` of i. As noted above, I'll use an **S5** logic, so $\mathcal{R}$ simply holds between any two members of $\mathcal{G}$.

Let $\mathcal{D}$ be the set of strings used in the Table entries of Section 2, specifically, {`1`, `2`, `3`, `4`, `automobile`, `motorcycle`, `piano`, `A`, `B`, `C`, `D`, `red`, `green`, `black`, `blue`, `yes`, `no`, `upright`, `grand`} (thus these are treated as both constant symbols of the language and as members of the ground level domain).

Finally the interpretation $\mathcal{I}$ is specified. On constant symbols of type 0, $\mathcal{I}$ is the identity function—such constant symbols designate themselves. For instance, $\mathcal{I}(\texttt{piano}) = \texttt{piano}$. And for the intensional constant symbols, we make them behave as the Locomobile tables of Section 2 specify. For instance, $\mathcal{I}(\texttt{IDNumber})$ is the function that maps Γ_1 to $\mathcal{I}(\texttt{1}) = \texttt{1}$, Γ_2 to $\mathcal{I}(\texttt{2}) = \texttt{2}$, and so on. $\mathcal{I}(\texttt{Engine})$ is the function that maps Γ_1 to $\{\mathcal{I}(\texttt{A}), \mathcal{I}(\texttt{B})\} = \{\texttt{A}, \texttt{B}\}$, Γ_2 to $\{\mathcal{I}(\texttt{A})\} = \{\texttt{A}\}$, Γ_3 to $\{\mathcal{I}(\texttt{C}), \mathcal{I}(\texttt{D})\} = \{\texttt{C}, \texttt{D}\}$, and has some arbitrary value on Γ_4. Likewise

$\mathcal{I}(\texttt{Relevant0})$ is the function that maps Γ_1, Γ_2 and Γ_3 to $\{\mathcal{I}(\texttt{IDNumber}), \mathcal{I}(\texttt{Item}), \mathcal{I}(\texttt{Cylinders})\}$, and maps Γ_4 to $\{\mathcal{I}(\texttt{IDNumber}), \mathcal{I}(\texttt{Item})\}$

This completes the definition of a language and a model corresponding to the database of Section 2. I'll call the model $\mathcal{M}_L$ from now on.

5 Queries

Databases exist to be queried. With higher type constructs present, a careful specification of behavior is needed to determine how queries behave. Modal models take care of this very simply, since we have a precise definition of truth available. The question is how to translate queries into the modal language. I'll give some natural language examples of queries for the Locomobile database, and then I'll provide formal versions in the modal language $L(C)$ specified in Section 4. For each I'll consider how the formal version behaves in the model $\mathcal{M}_L$ that was constructed in Section 4. It will be seen that the formal behavior matches intuition quite nicely.

Example 8. Query: Which items have 2 cylinders? Here and in the other examples, I'll use an item's IDNumber to uniquely identify it. As a first attempt at formalizing this query, we might ask for the value of the attribute `IDNumber` in worlds where the value of `Cylinders` is `2`. In effect, the modal operators $\Box$ and $\Diamond$ act like quantifiers over possible worlds, or records. And we can ask for the value of an attribute at a world by using the extension-of operator, $\downarrow$. This leads us to the following type $\uparrow\langle 0\rangle$ predicate abstract, in which α is a variable of type 0, and $=$ is the equality symbol of type $\langle 0, 0\rangle$.

$$\langle \lambda\alpha.\Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]\rangle \tag{1}$$

The problem with this is that the `Cylinders` attribute is undefined for pianos in Table 1. In [2] I specifically allowed partially defined objects, but with a full hierarchy of higher types available, I thought better of that approach here. Instead I introduced "relevancy" attributes. An entry of $\perp$ in a table indicates an irrelevant attribute; no value can have a meaning for the record. In a modal model constant symbols of intensional type are total, but values corresponding to $\perp$ are entirely arbitrary, and should not be considered in queries. Consequently, (1) must be revised to the following.

$$\langle \lambda\alpha.\Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]\rangle \tag{2}$$

Since this is the first example, I'll do it in some detail, beginning with a verification that (2) is well-formed. For later examples, things will be more abbreviated.

The constant symbol `IDNumber` is of type $\uparrow 0$, so $\downarrow\texttt{IDNumber}$ is of type 0, by part 3 of Definition 2. The variable α is of type 0 and $=$ is of type $\langle 0, 0\rangle$, so

$= (\texttt{IDNumber}, \alpha)$ is an atomic formula by part 4 of Definition 2. This we write more conventionally as $(\downarrow\texttt{IDNumber} = \alpha)$. In a similar way $(\downarrow\texttt{Cylinders} = \texttt{2})$ is an atomic formula. Finally, `Relevant0` is of type $\uparrow\langle\uparrow 0\rangle$ and `Cylinders` is of type $\uparrow 0$, so $\texttt{Relevant0}(\texttt{Cylinders})$ is an atomic formula by part 4 of Definition 2 again. It follows that $\Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]$ is a formula. Then (2) is a predicate abstract of type $\uparrow\langle 0\rangle$, by part 2 of Definition 2.

Now if τ is a constant of type 0, by part 4 of Definition 2,

$$\begin{aligned}\langle\lambda\alpha.\Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge \\ \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]\rangle(\tau)\end{aligned} \tag{3}$$

is a formula. The claim is, it is valid in the model $\mathcal{M}_L$ if and only if τ is `1` or `3`, which is exactly what we would expect intuitively. (Valid in the model means it is true at each world of it.) I'll check this in some detail for `3`.

Let Γ be an arbitrary world of the model, and let v be an arbitrary valuation. I want to verify the following.

$$\begin{aligned}\mathcal{M}_L, \Gamma \Vdash_v \langle\lambda\alpha.\Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge \\ \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]\rangle(\texttt{3})\end{aligned}$$

By part 5*a* of Definition 7, this is equivalent to

$$\begin{aligned}(v * \mathcal{I} * \Gamma)(\texttt{3}) \in (v * \mathcal{I} * \Gamma)(\langle\lambda\alpha.\Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge \\ \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]\rangle)(\Gamma)\end{aligned}$$

Now, $(v * \mathcal{I} * \Gamma)(\texttt{3}) = \mathcal{I}(\texttt{3}) = \texttt{3}$, so by part 4 of Definition 7 we must show

$$\mathcal{M}_L, \Gamma \Vdash_w \Diamond[(\downarrow\texttt{IDNumber} = \alpha) \wedge \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})]$$

where w is the α-variant of v such that $w(\alpha) = \texttt{3}$. And this is so because we have the following.

$$\mathcal{M}_L, \Gamma_3 \Vdash_w (\downarrow\texttt{IDNumber} = \alpha) \wedge \texttt{Relevant0}(\texttt{Cylinders}) \wedge (\downarrow\texttt{Cylinders} = \texttt{2})$$

I'll check two of the components. To verify that

$$\mathcal{M}_L, \Gamma_3 \Vdash_w (\downarrow\texttt{IDNumber} = \alpha)$$

we need

$$\langle(w * \mathcal{I} * \Gamma_3)(\downarrow\texttt{IDNumber}), (w * \mathcal{I} * \Gamma_3)(\alpha)\rangle \in (w * \mathcal{I} * \Gamma_3)(=).$$

But $(w*\mathcal{I}*\Gamma_3)(\downarrow\texttt{IDNumber}) = (w*\mathcal{I}*\Gamma_3)(\texttt{IDNumber})(\Gamma_3) = \mathcal{I}(\texttt{IDNumber})(\Gamma_3) = \texttt{3}$, and $(w * \mathcal{I} * \Gamma_3)(\alpha) = w(\alpha) = \texttt{3}$. And equality symbols are always interpreted as equality on extensional objects.

Finally I'll verify that

$$\mathcal{M}_L, \Gamma_3 \Vdash_w \texttt{Relevant0}(\texttt{Cylinders}).$$

This will be the case provided we have the following, by part 5*a* of Definition 7.

$$(w * \mathcal{I} * \Gamma_3)(\texttt{Cylinders}) \in (w * \mathcal{I} * \Gamma_3)(\texttt{Relevant0})(\Gamma_3)$$

Now, $(w*\mathcal{I}*\Gamma_3)(\texttt{Cylinders}) = \mathcal{I}(\texttt{Cylinders})$, and $(w*\mathcal{I}*\Gamma_3)(\texttt{Relevant0})(\Gamma_3) = \mathcal{I}(\texttt{Relevant0})(\Gamma_3) = \{\mathcal{I}(\texttt{IDNumber}), \mathcal{I}(\texttt{Item}), \mathcal{I}(\texttt{Cylinders})\}$, and we are done.

Equation (3) has been verified in the case where τ is `1`. The case where it is `3` is similar. If τ is `2`, it fails because of the $(\downarrow\texttt{Cylinders} = \texttt{2})$ clause. And the case where τ is `4` fails because of the $\texttt{Relevant0}(\texttt{Cylinders})$ clause.

I'll conclude the section with a few more examples of somewhat greater complexity. There will be no detailed analysis for these.

Example 9. Query: what choices does a customer have when purchasing a four-cylinder car? This turns into the following predicate abstract, where α is of type $\uparrow\langle 0\rangle$ and β is of type 0. (I've omitted relevancy clauses because `Item` is always relevant, and for `automobile` items `Cylinders` is always relevant. These will be among the various constraints discussed in the next section.)

$$\begin{aligned}\langle \lambda\alpha, \beta.\Diamond[(\downarrow\texttt{Item} = \texttt{automobile}) \wedge (\downarrow\texttt{Cylinders} = \texttt{4}) \\ \wedge \texttt{Customer}(\alpha) \wedge \alpha(\beta)]\rangle\end{aligned} \tag{4}$$

Abbreviating (4) by τ, we have $\tau(\tau_1, \tau_2)$ is valid in $\mathcal{M}_L$ just in case $\langle \tau_1, \tau_2\rangle$ is one of

$$\begin{gathered}\langle \texttt{Colors}, \texttt{green}\rangle \\ \langle \texttt{Colors}, \texttt{black}\rangle \\ \langle \texttt{Air}, \texttt{yes}\rangle \\ \langle \texttt{Air}, \texttt{no}\rangle\end{gathered}$$

Example 10. Query: what features can a customer choose, that are available for more than one product? This gives us the following predicate abstract, in which α is of type $\uparrow\langle 0\rangle$, and β, γ, and δ are of type 0.

$$\begin{aligned}\langle \lambda\alpha, \beta.&\texttt{Customer}(\alpha) \wedge \\ &(\exists\gamma)(\exists\delta)\{\neg(\gamma = \delta) \wedge \\ &\Diamond[(\uparrow\texttt{IDNumber} = \gamma) \wedge \alpha(\beta)] \wedge \\ &\Diamond[(\uparrow\texttt{IDNumber} = \delta) \wedge \alpha(\beta)]\}\rangle\end{aligned} \tag{5}$$

Equation (5) validly applies, in $\mathcal{M}_L$, to just the following.

$$\begin{gathered}\langle \texttt{Colors}, \texttt{green}\rangle \\ \langle \texttt{Colors}, \texttt{black}\rangle\end{gathered}$$

6 Constraints

The Locomobile example is really an instance of a database scheme. In order to qualify as an instance, certain constraints must be met. So far, these have been

implicit, but now it is time to state them precisely. This provides additional examples of the modal machinery at work.

I've been treating `IDNumber` as a key attribute. I now want to make this a formal requirement. Ordinarily, to say something is a key is to say there cannot be two records that have a common value on this attribute. In a modal setting, this means the constant symbol `IDNumber` cannot be interpreted to have the same value at two possible worlds. But possible worlds cannot be referred to directly in our modal language. What we can say instead is that, in any model, worlds agreeing on a value for `IDNumber` must agree on every attribute. Since we have a full type theory here, this cannot be said with a single formula—we need an infinite family of them, one for each intensional type. Consider the following formula, where α is of type $\uparrow t$, x is of type 0 and y is of type t.

$$(\forall\alpha)\langle\lambda x, y.\Box[(x =\downarrow\texttt{IDNumber}) \supset (y =\downarrow\alpha)]\rangle(\downarrow\texttt{IDNumber}, \downarrow\alpha) \tag{6}$$

Requiring validity of (6) in a model is equivalent to requiring that two worlds where `IDNumber` is interpreted identically are worlds that agree on values of all intensional attributes of type $\uparrow t$. For the Locomobile example, we only need 5 instances: for types $\uparrow 0$, $\uparrow\langle 0\rangle$, $\uparrow\langle\uparrow 0\rangle$, $\uparrow\langle\uparrow\langle 0\rangle\rangle$, and $\uparrow\langle\uparrow\langle\uparrow\langle 0\rangle\rangle\rangle$.

Formula (6) is actually of more general interest than would appear at first glance. In [2] I noted that such a formula is a *relative rigidity* expression—it requires that all intensional objects of type $\uparrow t$ be rigid relative to `IDNumber`. Such requirements can be more elaborate, requiring rigidity relative to some combination of attributes. They can also be less elaborate, requiring absolute rigidity. As such, they relate to Kripke's notion of *rigid designator* in [7], but a further discussion would take us too far afield here.

In Example 9 I noted that `Item` should always be relevant. Clearly so should `IDNumber`. I also noted that `Cylinders` should be relevant for items that were automobiles. This means we should require validity of the following.

$$\begin{aligned}&\texttt{Relevant0}(\texttt{Item})\\&\texttt{Relevant0}(\texttt{IDNumber})\\&(\downarrow\texttt{Item} = \texttt{automobile}) \supset \texttt{Relevant0}(\texttt{Cylinders})\end{aligned}$$

To be precise, for a modal model to be considered as an instance of the Locomobile scheme, the various constraints above must be valid formulas in it.

This can be turned into a proof-theoretic condition as well. Consider the tables of Section 2 again. It is not hard to see that the first line of Table 1 corresponds to the following formula.

$$\begin{aligned}\Diamond[&(\downarrow\texttt{IDNumber} = 1) \wedge (\downarrow\texttt{Item} = \texttt{automobile}) \wedge (\downarrow\texttt{Cylinders} = 2) \wedge\\&\texttt{Engine}(\texttt{A}) \wedge \texttt{Engine}(\texttt{B}) \wedge\\&\texttt{Colors}(\texttt{red}) \wedge \texttt{Colors}(\texttt{green}) \wedge \texttt{Colors}(\texttt{black}) \wedge\\&\texttt{Air}(\texttt{no})]\end{aligned}$$

Similarly for the other lines, and tables. Now, to say we have presented an instance of the Locomobile database scheme amounts to saying the constraint formulas given earlier, combined with the various formulas derived from the tables and representing individual records, make up a consistent set.

Consistency can, of course, be checked using a proof procedure, and the higher type modal logic used here does have a tableau system, see [3]. But that system is complete relative to a Henkin model version of our semantics, and is not complete relative to the "true" semantics given in Section 3. Also, a tableau procedure is not a decision method. I leave it as an open problem whether, for formulas of the particular forms that arise in database applications, a decision procedure can be extracted from the tableau method.

7 Conclusion

As promised, I have not proved any theorems. I have, however, provided a precise modal semantics that can be applied naturally to databases containing higher type constructs. Issues of practicability of implementation have been ignored. Issues of decidability for fragments directly applicable to databases have been ignored. I wanted to present the basics with the hope that others would find the subject of sufficient interest to pursue questions like these. I hope I have succeeded, at least a little.

References

1. S. Feferman, J. John W. Dawson, W. Goldfarb, C. Parsons, and R. N. Solovay, editors. *Kurt Gödel Collected Works, Volume III, Unpublished Essays and Lectures*. Oxford University Press, New York, 1995.
2. M. C. Fitting. Modality and databases. Forthcoming, LNCS, Tableaux 2000, 2000.
3. M. C. Fitting. *Types, Tableaus, and Gödel's God*. 2000. Available on my web site: comet.lehman.cuny.edu/fitting.
4. M. C. Fitting and R. Mendelsohn. *First-Order Modal Logic*. Kluwer, 1998. Paperback, 1999.
5. D. Gallin. *Intensional and Higher-Order Modal Logic*. North-Holland, 1975.
6. K. Gödel. Ontological proof. In Feferman et al. [1], pages 403–404.
7. S. Kripke. *Naming and Necessity*. Harvard University Press, 1980.
8. R. Montague. On the nature of certain philosophical entities. *The Monist*, 53:159–194, 1960. Reprinted in [11], 148–187.
9. R. Montague. Pragmatics. pages 102–122. 1968. In *Contemporary Philosophy: A Survey*, R. Klibansky editor, Florence, La Nuova Italia Editrice, 1968. Reprinted in [11], 95–118.
10. R. Montague. Pragmatics and intensional logic. *Synthèse*, 22:68–94, 1970. Reprinted in [11], 119–147.
11. R. H. Thomason, editor. *Formal Philosophy, Selected Papers of Richard Montague*. Yale University Press, New Haven and London, 1974.