COSC462 Lecture 13: Transparent languages

Willem Labuschagne
University of Otago

Abstract

We introduce the analogues of nouns and verbs into the object language. The interpretation of sentences now involves denotation in a domain. The expressive power we gain is incremented by successively introducing variables, function symbols, and sorts.

1 Opaque atoms

Consider the simple Light-Fan System with its two components, the light and the fan. On previous occasions we were interested in states of the system determined by which (if any) of two basic facts hold — whether the light is on, and whether the fan is on. It was sufficient to use a language generated by two atoms, p and q.

Now consider the slightly more complicated Control Room System. There are still two components, a light and a fan, but there are more things we want to say about them. Imagine that the light is on a control panel in the control room, and tells the agent whether the fan, which is elsewhere, is on. Of course, bulbs can wear out, wires can short-circuit, and in general things can go wrong. So the states of the system are determined by four basic facts that may or may not be the case:

- The light may be on (i.e. may be shining) or it may not.
- The fan may be on (i.e. may be spinning) or not.
- The light may be functioning properly (shining when it is supposed to and being dark when it is supposed to) or the light may be malfunctioning (may fail to shine when it should shine, or may shine when it shouldn't).
- The fan may be functioning properly or may be malfunctioning.

If we apply our familiar technique of representing knowledge to the Control Room System, we discover a weakness of opaque propositional languages — the language does not make it easy to see when different sentences are about the same component or concern the same property.

Take, for example, a propositional language based on the four atoms in $A = \{p, q, r, s\}$, where p stands for the light being on, q for the fan being on, r for the light functioning correctly, and s for the fan functioning correctly. This is a straightforward way to represent knowledge about the Control Room System, precisely analogous to the way in which we treated the Light-Fan System. But notice that certain things the agent might know are hidden in this language. For instance, the agent surely knows that the same component of the system (the light) is involved in both the fact expressed by p and the fact expressed by r. However, the letters 'p' and 'r' are *opaque* — they don't let anyone peep inside and see that they are both asserting something about the light.

In the metalanguage it is possible to say 'The light is on' and 'The light is functioning normally', and these sentences make it explicit that the same component is involved. Can the agent's knowledge representation language be modified to make it a bit more like English, more *transparent*?

2 Transparent atoms

To build a transparent language suitable for representing information about the Control Room System, we need names for the components. Perhaps we could equip the language with names like 'L' for the light and 'F' for the fan. In order to make sentences, the language would need more than names — it has to have the equivalent of verbs. The verbs of interest for the Control Room System could be built into the language as the strings '$IsOn$' and '$IsNormal$'. Now the four atomic facts about the system could be expressed by putting together the names and verbs as follows.

- $IsOn(L)$ expresses the idea that the light is shining, and is a transparent version of the atom p;
- $IsOn(F)$ expresses the idea that the fan is spinning, and is a transparent version of the atom q;
- $IsNormal(L)$ expresses the idea that the light is functioning properly, and is a transparent version of r;
- $IsNormal(F)$ expresses that the fan is functioning properly, and is a transparent version of atom s.

Names such as F are called *constant symbols*, while verbs such as $IsOn$ are called *predicate symbols*. A transparent propositional language is the simplest kind of *predicate language*, and is really just the usual sort of propositional language, with a single change — the set of atoms is $A = \{IsOn(L),\ IsNormal(L),\ IsOn(F),\ IsNormal(F)\}$ instead of $\{p, q, r, s\}$. As before, compound sentences may be built up by combining the atoms in familiar ways, using the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ (and if required modal operators like $\Box$). Note that although the new atoms are strings of symbols, these strings cannot be confused with compound sentences, because the atoms themselves contain no occurrences of connectives and do not have parentheses enclosing them. Thus the sentences of the language remain unambiguous.

Definition 1 *Let $Cons$ be some set, and let $Pred$ be a function consisting of ordered pairs (P, n) such that the second co-ordinate n is a natural number.*

The set A of ***atomic sentences*** *induced by $Cons$ and $Pred$ consists of all strings $P(c_1, c_2, \ldots, c_n)$ such that*

- $(P, n) \in Pred$
- $c_1, c_2, \ldots, c_n \in Cons$.

The members of $Cons$ are the constant symbols of the language we are building, and they provide names for at least some of the things we want to talk about in the system of interest. Each ordered pair in $Pred$ consists of a predicate symbol P together with its arity n, and is the name of an n-ary relation that is of interest to us.

Example 2 *For the Control Room system, we may take $Cons = \{L, F\}$ and $Pred = \{(IsOn, 1), (IsNormal, 1)\}$.*

In an atom $P(c_1, c_2, \ldots, c_n)$, the constant symbols $c_1, \ldots, c_n$ are the *arguments* of the predicate symbol P, and there are exactly as many arguments in the atom as the arity of P demands. In the special case where the arity n of a predicate symbol P is 0, the predicate symbol takes no arguments. An opaque propositional language may be regarded as having its set of atoms built up from $Cons = \varnothing$ and a set $Pred$ whose pairs are all of the form $(P, 0)$. Thus the 0-ary predicate symbols themselves form the atoms. Clearly, opaque propositional languages are a special case of the class of transparent propositional languages we are now in the process of describing.

Definition 3 *Let A be a set of atomic sentences induced by sets Cons and Pred. Then α is a* ***sentence*** *over A iff one of the following is the case:*

- $\alpha \in A$
- $\alpha = (\neg\beta)$ *where β is already a sentence over A*
- $\alpha = (\beta * \gamma)$ *where β and γ are already sentences over A and $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

  The transparent propositional language L_A generated from A is the set of all sentences over A.

The idea is that, in order to build a transparent propositional language, one begins by specifying some set of constant symbols and some set of predicate symbols together with their arities. Then one builds the atomic sentences of the language by combining each predicate symbol with all possible combinations of the correct number of constants as arguments. Finally one uses the standard connectives to construct compound sentences just as one would in any propositional language.

Example 4 *Here is a simple language. Take $Cons = \{a, b\}$ and $Pred = \{(P, 1)\}$. Then $A = \{P(a), P(b)\}$. Example sentences of the language L_A are $\neg P(a)$, $P(a) \rightarrow P(b)$, and $\neg P(a) \wedge (P(a) \rightarrow P(b))$. As always, we omit parentheses where this harmlessly enhances readability.*

Predicate languages, or transparent languages, come in various flavours but they all start with a set of predicate symbols. Using such a language allows us to understand the allocation of truth values in terms of the *meanings* of the constant and predicate symbols. We always have some intended meaning in mind. A knowledge representation language, to an applied logician, is not merely a set of strings built up according to some grammar. Instead we typically start with some system of interest, and build a language specifically to talk about that system. Our attitude may be summarised by McCarthy's Axiom:

Axiom 5 *No notation without denotation.*

So what do we mean when we speak of denotation? How do the strings of our language get meaning? Recall that the semantics of an opaque propositional language is provided by an ontology (S, V), where S is some set of states and $V : S \longrightarrow W_A$ connects S with the set W_A of valuations $v : A \longrightarrow \{1, 0\}$ associated with the language. In the case of a predicate language, we can say a lot more about what states are and how the valuations arise.

3 Interpretations

We arrive at valuations by first *interpreting* the language. To interpret a predicate language, we specify the denotations of the constant and predicate symbols. One specifies the denotation of a symbol by indicating some object, named by that symbol, inside some collection of things. This collection of things the language is talking about is called the *domain* of the interpretation, or the *universe of discourse*. Usually we would have created the language to talk about some specific domain of things, namely the components of the system we're interested in, and we will call this the *intended* interpretation.

Definition 6 *Let L_A be the transparent propositional language generated from A, where A is the set of atomic sentences induced by some $Cons$ and $Pred$.*

*An **interpretation** of L_A is a pair $\mathcal{D} = (D, den)$ where D is a non-empty set (the domain) and den is a function (the denotation function) assigning*

- *to each constant $c \in Cons$ a member $den(c) \in D$*
- *to each $(P, n) \in Pred$ a subset $den(P, n) \subseteq D^n$.*

What might be an example of an interpretation? Well, here is a very simple possibility for our language about the Control Room System. Take, as domain, the set $\{0, 8\}$. Why? Well, because the number 0 resembles the light and so reminds me of what it represents, and because the number 8 reminds me of the blades of a fan. Thus it seems reasonable to take the iconic representation 0 as the denotation of the constant symbol $Light$ and the iconic representation 8 as the denotation of Fan.

Of course, if all we want is a domain in which there are two things which remind us of the light and the fan respectively, then there is nothing to stop us from taking the domain to be just the set of constant symbols $\{L, F\}$.

What we don't do is to take the domain to be the 'real' light and 'real' fan of the system. There are two reasons. In the first place, there may not be a real system — we may want to able to reason about totally imaginary systems containing winged horses or non-fattening chocolate cake. But even if the system is real, we cannot exhibit the real components to everyone with whom we may wish to discuss the system, so we would in any case have to represent the system by something more convenient — and this cannot simply be the sentences of the knowledge representation language, because that would not give us any certainty that we understand the sentences in the same way. Think of it like this.

If we want to talk to someone about London, it may be impractical to show them the city itself, but useful to show them photographs or a map of the city. The interpretation of a language is like a map of the system rather than the system itself.

There are thus three things for us (the logicians) to keep separate in our minds —

- the real system of interest (e.g. the real city of London or a powerplant),
- a 'map' or iconic representation of the system (typically a mathematical structure, called an interpretation), and
- sentences of the agent's knowledge representation language.

Interpretations allow agents to have a common ontology, so that they can be sure they're talking about the same things. There is a special kind of interpretation that is often convenient to use, namely *term* interpretations.

Definition 7 *If $D = Cons$ and for each $c \in Cons$, we have $den(c) = c$, then $\mathrm{D} = (D, den)$ is called a* ***term interpretation****.*

But why might we want to take the constants of the language as the objects in the domain of interpretation? Well, imagine that you want to talk to a friend about something that happened to you in London. So London is the system of interest. Your intended interpretation may be a map of London, or even a collection of photographs you took while on holiday there. But suppose it is not convenient to use your intended interpretation because you left the photos or map at home. You may still be able to draw a rough sketch labelled by street names and names of buildings to remind him of the area, so that he can visualise your adventure. The rough sketch is a term interpretation.

Here's a second example. Suppose you are arranging an exhibition of butterflies in the museum and have a meeting with the curator. You'd like to show him your intended interpretation of the system, which is a brochure with nice coloured pictures of the butterflies. Unfortunately, at the time you discuss the arrangements with the curator the brochure may still be at the printers. So you draw a rough diagram showing the arrangement of exhibits, making use of the fact that each exhibit is labelled with the name of the species so that you can use the labels instead of accurate pictures of the butterflies. (The species names are the constants in the scientific language used for talking about the butterflies.) The rough sketch is a term interpretation.

The moral is that term interpretations of a language are very easy to form, because the language gives you the objects of the domain. And term interpretations are fairly close to the intended interpretation, because it was on the basis of the intended interpretation that you would have chosen the set of names, $Cons$.

Let us return to our description of how one interprets a language. We first specify the domain — for example, the set $Cons = \{L, F\}$ or the set $E = \{0, 8\}$. The second step is to say what the constant symbols of the language denote, in other words which objects in the domain correspond to which constants in the language. In the case of $Cons$, this is obvious — we let each constant symbol denote itself. So the constant symbol L of the language is taken to denote the element $L \in Cons$. Similarly the constant symbol F of the language is taken to denote the element $F \in Cons$. Nothing could be simpler. In the case of E, we let L denote 0 and F denote 8. Also very simple.

The third step in interpreting a predicate language is to say what the 'verbs' mean, in other words what the predicate symbols denote. (If you think of an interpretation as a sort of sketch, then this part has to do with drawing the lines to connect labelled things.)

The key difference between constant symbols and predicate symbols is that a constant denotes an individual object while predicate symbols always denote sets of things. These sets may be small, for instance having just a single member or even no members at all (the empty set). Or the sets may be large. And the things inside the sets may be individual objects in the domain, or they may be ordered pairs of such elements, or triples, or more complex things even than that. Different choices of denotations give different interpretations of the language. In order to make sensible choices, one must keep in mind the *arity* of predicate symbols.

The arity of a predicate symbol is the number of arguments that the predicate symbol expects to be given. Recall how constant symbols and predicate symbols were combined to form atoms — each predicate symbol had to be combined with a specific number of constant symbols. The predicate symbol $IsOn$ was combined with the single constant F to give the atom $IsOn(F)$. In another language there may be predicate symbols such as $IsLessThan$ which would be combined with two constants at a time, for instance with $Zero$ and $Three$ to form the atom $IsLessThan(Zero, Three)$. Thus the arity of $IsOn$ is 1, while the arity of $IsLessThan$ is 2.

Now when we interpret a language and reach the stage of assigning denotations to the predicate symbols, the rule is the following. A predicate symbol like $IsOn$ which has arity 1 must receive as its denotation a subset of the domain. A predicate symbol of arity 2 must receive a set

of ordered pairs from the domain. A predicate symbol of arity 3 must receive a set of ordered triples from the domain, and so on. More generally, the denotation of a predicate symbol P with arity n in a domain D must be a subset of D^n. By D^n we of course understand the Cartesian product $D \times D \times D \times \ldots \times D$ involving n copies of D, so the members of D^n are n-tuples $(d_1, d_2, \ldots, d_n)$ whose co-ordinates $d_1, d_2, \ldots, d_n$ all live in D. The cases that are of special interest, because they arise most frequently, are those in which $n = 1$ and those in which $n = 2$. Given a predicate symbol P of arity 1, the denotation $den(P, 1)$ must be a subset of D^1, and of course D^1 is just D. Given a predicate symbol P of arity 2, the denotation $den(P, 2)$ must be a subset of $D^2 = D \times D = \{(d_1, d_2) \mid d_1 \in D \text{ and } d_2 \in D\}$.

In the case of our illustrative transparent language, both predicate symbols $IsOn$ and $IsNormal$ have arity 1. Let us choose subsets of the domain $Cons$ on which we build term interpretations. The domain $Cons = \{L, F\}$ has four subsets, namely $\emptyset$, $\{L\}$, $\{F\}$, and $\{L, F\}$. Any of these may be selected as the denotation of $IsOn$, and in a similar fashion any of them may be selected to be the denotation of $IsNormal$. In this way we can get different term interpretations all of which are built on the same domain.

Example 8 *By way of example, let us select* $\{L\}$ *as denotation of* $IsOn$*, and* $\{F\}$ *as denotation of* $IsNormal$*. This gives us one particular term interpretation of the language, namely* $\mathcal{D} = (D, den_D)$ *where*

- *the domain is the set* $D = Cons = \{L, F\}$
- *the constant symbol 'L' denotes the element* L *of the domain, written* $den_D(L) = L$*, and the constant symbol 'F' denotes the element* F *of the domain, written* $den_D(F) = F$
- *the predicate symbol 'IsOn' denotes the subset* $\{L\}$ *of the domain, written* $den_D(IsOn, 1) = \{L\}$*, and 'IsNormal' denotes the subset* $\{F\}$*, written* $den_D(IsNormal, 1) = \{F\}$*.*

Similarly, the domain $E = \{0, 8\}$ has four subsets, any of which might be chosen as denotations of $IsOn$ and $IsNormal$, each choice giving us a different interpretation.

We think of the interpretations as picturing the states of a system. From states we need to go to valuations, so that we can associate truth values with sentences. Fortunately, every interpretation determines a particular valuation in a straightforward way. The interpretation is like a laboratory in which we can go and test which basic facts hold.

Let us see how the term interpretation $\mathcal{D}$ with domain $D = Cons$ determines a valuation.

The 4 atoms are $IsOn(L)$, $IsOn(F)$, $IsNormal(L)$, and $IsNormal(F)$.

To check whether the atom $IsOn(L)$ is true relative to the interpretation $\mathcal{D}$, we must find out whether the element L of the domain D belongs to the denotation of $IsOn$ (if it doesn't, the atom is false relative to the given interpretation). Since the denotation of $IsOn$ is $\{L\}$, and obviously $L \in \{L\}$, it follows that $IsOn(L)$ is true in $\mathcal{D}$.

On the other hand, the atom $IsOn(F)$ is false relative to $\mathcal{D}$, because the denotation in $\mathcal{D}$ of $IsOn$ is $\{L\}$ and $F \notin \{L\}$.

Similarly, $IsNormal(L)$ is false in $\mathcal{D}$ because the denotation in $\mathcal{D}$ of predicate symbol $IsNormal$ is $\{F\}$ and $L \notin \{F\}$.

Finally $IsNormal(F)$ is true in $\mathcal{D}$ because $den_D(IsNormal, 1) = \{F\}$ and $F \in \{F\}$.

So what is the valuation determined by the interpretation? Clearly it is the function $v : A \longrightarrow \{0, 1\}$ given by $v(IsOn(L)) = 1$, $v(IsOn(F)) = 0$, $v(IsNormal(L)) = 0$, and $v(IsNormal(F)) = 1$. As long as we stick to the order in which the atoms were displayed above, always mentioning the light before the fan and the predicate $IsOn$ before $IsNormal$, the valuation v can be abbreviated more digestibly as the binary string 1001.

Definition 9 *Let L_A be a transparent propositional language and $\mathcal{D} = (D, den)$ an interpretation of L_A.*

*The **valuation** determined by $\mathcal{D}$ is the function $v : A \longrightarrow \{0, 1\}$ such that for every $P(c_1, \ldots, c_n) \in A$,*

$v(P(c_1, \ldots, c_n)) = 1$ iff $(den(c_1), \ldots, den(c_n)) \in den(P, n)$.

We write W_A for the set of all valuations of L_A.

Example 10 *Suppose L_A is the simple language with $Cons = \{a, b\}$ and $Pred = \{(P, 1)\}$. The language has four term interpretations $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, and $\mathcal{D}_4$, where*

- *$\mathcal{D}_1 = (Cons, den_1)$ with $den_1(a) = a$, $den_1(b) = b$, and $den_1(P, 1) = \varnothing$*
- *$\mathcal{D}_2 = (Cons, den_2)$ with $den_2(a) = a$, $den_2(b) = b$, and $den_2(P, 1) = \{a\}$*
- *$\mathcal{D}_3 = (Cons, den_3)$ with $den_3(a) = a$, $den_3(b) = b$, and $den_3(P, 1) = \{b\}$*
- *$\mathcal{D}_4 = (Cons, den_4)$ with $den_4(a) = a$, $den_4(b) = b$, and $den_4(P, 1) = \{a, b\}$.*

L_A has two atomic sentences, namely $P(a)$ and $P(b)$. So the valuations v_i determined by the interpretations $\mathcal{D}_i$ are given by

- *$v_1(P(a)) = 0$ and $v_1(P(b)) = 0$, since $den_1(P, 1) = \emptyset$*
- *$v_2(P(a)) = 1$ and $v_2(P(b)) = 0$, since $den_2(P, 1) = \{a\}$*
- *$v_3(P(a)) = 0$ and $v_3(P(b)) = 1$, since $den_3(P, 1) = \{b\}$*
- *$v_4(P(a)) = 1$ and $v_4(P(b)) = 1$, since $den_4(P, 1) = \{a, b\}$.*

If we agree to think of $P(a)$ as the first atomic sentence and $P(b)$ as the second, then the valuations v_1, v_2, v_3, v_4 may be abbreviated by 00, 10, 01, and 11 respectively.

We have seen that every interpretation determines a valuation, and indeed every valuation can be obtained from some interpretation. So as long as we have either an interpretation or a valuation, we can allocate truth values to sentences.

Definition 11 *Let $\mathcal{D} = (D, den)$ be an interpretation of L_A and let v be the valuation determined by $\mathcal{D}$.*

*We say that $\mathcal{D}$ (or v) **satisfies** (or makes true) a sentence α of L_A according to the following rules:*

- *if $\alpha \in A$, i.e. α is an atom $P(c_1, \ldots, c_n)$, then $\mathcal{D}$ (or v) satisfies α iff $(den(c_1), \ldots, den(c_n)) \in den(P, n)$, which is the same as to say that $v(\alpha) = 1$*
- *$\mathcal{D}$ (or v) satisfies $\neg\beta$ iff $\mathcal{D}$ fails to satisfy β*
- *$\mathcal{D}$ (or v) satisfies $\beta \wedge \gamma$ iff $\mathcal{D}$ satisfies both β and γ*
- *$\mathcal{D}$ (or v) satisfies $\beta \vee \gamma$ iff $\mathcal{D}$ satisfies at least one of β and γ*
- *$\mathcal{D}$ (or v) satisfies $\beta \rightarrow \gamma$ iff $\mathcal{D}$ satisfies γ or fails to satisfy β*
- *$\mathcal{D}$ (or v) satisfies $\beta \leftrightarrow \gamma$ iff $\mathcal{D}$ satisfies both β and γ or neither.*

Definition 12 *An **ontology** for a transparent propositional language L_A is a pair (S, V) where S is some set of interpretations of L_A and the labelling function $V : S \longrightarrow W_A$ is the obvious one taking an interpretation $\mathcal{D} \in S$ to the valuation v determined by $\mathcal{D}$.*

In other words, we don't have to work with all the possible interpretations of language L_A, we just pick out those we're interested in and put them into S. For example, we could stick to just the term interpretations if we wanted.

Definition 13 *If $\mathcal{D}$ (or the valuation v determined by $\mathcal{D}$) satisfies α we say that $\mathcal{D}$ (or v) is a* ***model*** *of α.*

We write $\mathcal{M}(\alpha)$ for the subset of S consisting of all the models of α (and also for the subset of W_A consisting of all $v = V(\mathcal{D})$ where $\mathcal{D}$ is a model of α). If $u \in W_A$ is a valuation that satisfies α but $u \neq V(\mathcal{D})$ for any interpretation $\mathcal{D} \in S$, then we will call u a spurious model of α.

The idea of a spurious model is that there may be more valuations in W_A than realisable states of the system in S, and so we want to be able to say of a valuation that, yes, technically it would satisfy α but it can be ignored because it doesn't correspond to any interpretation we're interested in. (We saw this sort of situation earlier, when we encountered the 3 Card System.)

Example 14 *Consider the language L_A where A is induced by $Cons = \{a, b\}$ and $Pred = \{(P, 1)\}$. Let S consist of the four term interpretations $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, and $\mathcal{D}_4$.*

Now $\mathcal{D}_1 \notin \mathcal{M}(P(a))$ since $den_1(P, 1) = \emptyset$ and so $den_1(a) \notin den_1(P, 1)$. We may equally well say that $v_1 \notin \mathcal{M}(P(a))$ since v_1 is the valuation 00 and we have agreed to think of the atomic sentences in the order $P(a)$ first and $P(b)$ second.

On the other hand, $\mathcal{D}_2 \in \mathcal{M}(P(a))$ since $den_2(P, 1) = \{a\}$. Equally, we may say that $v_2 \in \mathcal{M}(P(a))$ since v_2 is the valuation 10.

Moving to a more interesting sentence, $\mathcal{D}_4 \in \mathcal{M}(P(a) \leftrightarrow P(b))$ since $den_4(P, 1) = \{a, b\}$ so that $\mathcal{D}_4$ satisfies both $P(a)$ and $P(b)$, and we may equally well say that $v_4 \in \mathcal{M}(P(a) \leftrightarrow P(b))$.

Let's look at some other important ideas and see whether they are easy to formalise.

Firstly, there is an important predicate symbol we may want to use in our language, namely $(=, 2)$. The idea is that this binary predicate symbol stands for equality.

Definition 15 *If $(=, 2) \in Pred$ then we call L_A a language with* ***equality*** *and insist that, in every interpretation $\mathcal{D} = (D, den)$ of the language, $den(=, 2)$ must be the identity relation $I_D = \{(d, d) \mid d \in D\}$.*

There are also special properties sentences may possess.

Definition 16 *Let $X \subseteq S$ and let $\alpha \in L_A$.*

- *α is* ***valid*** *iff $\mathcal{M}(\alpha) = S$*
- *α is a* ***tautology*** *iff every $v \in W_A$ satisfies α*

- α ***axiomatises*** X *iff* $X = \mathcal{M}(\alpha)$
- α *is a* ***contradiction*** *iff* $\mathcal{M}(\alpha) = \varnothing$

Since S may be smaller than W_A, we may have sentences that are satisfied by all the interpretations in S but which are not tautologies, and so we introduce the term 'valid' to describe them. One of the most important ideas in logic is that of having some subset of interpretations (or valuations) and being able to describe those particular interpretations (or valuations) in the language, at least well enough to distinguish them from any other bunch of interpretations (or valuations. This is what axiomatisability is about. By way of example, we might say that $P(b)$ axiomatises $\{\mathcal{D}_3, \mathcal{D}_4\}$ above.

This brings us to the most important logical concept of all — entailment. We restrict consideration to classical entailment, but there is nothing except lack of space that prevents us from talking about defeasible entailment as well.

Definition 17 *Let* $\alpha, \beta \in L_A$. *Then* $\alpha \models \beta$ *iff* $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$ *and* $\alpha \equiv \beta$ *iff* $\mathcal{M}(\alpha) = \mathcal{M}(\beta)$.

For example, $P(a) \leftrightarrow P(b) \models P(a) \rightarrow P(b)$ but it is not the case that $P(a) \leftrightarrow P(b) \equiv P(a) \rightarrow P(b)$. We may think of equivalence (the relation $\equiv$) as being derived from entailment, because $\alpha \equiv \beta$ iff both $\alpha \models \beta$ and $\beta \models \alpha$.

Exercise 18 *1. Consider* L_A *where* A *is induced by* $Cons = \{a\}$ *and* $Pred = \{(P, 1), (Q, 1)\}$.

- *Write down the atomic sentences and the term interpretations of* L_A. *Give the valuation determined by each term interpretation (you may write valuations as binary strings). Do any other valuations exist, i.e. are there any other functions from* A *to* $\{0, 1\}$*?*
- *Write down a tautology and a contradiction.*
- *Find a sentence entailed by* $P(a)$*, another sentence that entails* $P(a)$*, and a third sentence equivalent to* $P(a)$.

2. Consider L_A *where* A *is induced by* $Cons = \{a, b\}$ *and* $Pred = \{(P, 1), (Q, 1)\}$.

- *Write down the atomic sentences and the term interpretations of L_A. Give the valuation determined by each term interpretation (you may write valuations as binary strings). Do any other valuations exist, i.e. are there any other functions from A to $\{0,1\}$?*
- *Write down a tautology and a contradiction.*
- *Find a sentence entailed by $P(a) \wedge Q(b)$, another that entails $P(a) \wedge Q(b))$, and a third equivalent to $P(a) \wedge Q(b)$.*

3. *Consider L_A where $Cons = \{a, b\}$ and $Pred = \{(=, 2)\}$.*

   - *Write down the atomic sentences and the term interpretations of L_A. Give the valuation determined by each term interpretation (you may write valuations as binary strings). Do any other valuations exist, i.e. are there any other functions from A to $\{0,1\}$?*
   - *Write down a tautology and a contradiction.*
   - *Find a sentence entailed by $= (a, a)$, another that entails $= (a, a)$, and a third equivalent to $= (a, a)$.*

4. *Consider L_A where $Cons = \{\ldots, -2, -1, 0, 1, \ldots\}$ and $Pred = \{(P, 2)\}$. This language would be suitable for representing knowledge about the set of integers, with the predicate symbol P intended to stand for the usual order relation $\leq$. One of the term interpretations of the language is an accurate representation of the set of integers. Which one? Give an example of a term interpretation which is emphatically not an accurate representation of the set of integers. Give a sentence which is true in the inaccurate representation but false in the accurate representation.*

5. *As exercises in knowledge representation, we describe several 'realistic' systems. In each case, decide on suitable sets Cons and Pred. By 'suitable' we mean that the language that results from your choice should be able to express the basic facts that are relevant but need express no others. For convenience you may use letters like 'a', 'b', 'c' as your constant symbols and letters like 'P', 'Q', 'R' as your predicate symbols.*

   - *Consider the Control Room System having two components, a light and a fan, each of which may be on or off, and each of which may be normal or defective. We have described suitable sets Cons and Pred already. Describe a term interpretation*

that represents the state of the system in which the light is off but functioning correctly.

- *Consider a system whose two components are the horses Andy and Bandy. The horses may or may not need to be fed, and they may or may not need to be groomed. Describe suitable set Cons and Pred. Describe a term interpretation that represents the state of the system in which both horses need to be fed but neither horse needs to be groomed.*
- *Consider a system whose two components are the persons Glawen Clattuc and Bodwen Wook. These persons may or may not hold themselves or each other in high esteem. Describe suitable sets Cons and Pred. Describe a term interpretation that represents the state of the system in which Glawen has high self-esteem but neither person holds the other in high esteem and Bodwin knows that he himself is a twit.*
- *Consider a variant of the Light-Fan-Heater System. There are three components — a light, a fan, and a heater. Each component may be either on or off. Each component may be either normal or defective. Describe suitable sets Cons and Pred. Describe a term interpretation that represents the state of the system in which the components are all on and functioning correctly.*
- *Consider the 3 Card System. There are three players (numbered 1, 2, and 3) and three cards (red, green, and blue). Suppose one card is dealt to each player. Such a deal is a state of the system. Describe suitable sets Cons and Pred. Describe a term interpretation that does not represent a state of the system. Which term interpretations represent states?*
- *Consider an urn containing n balls (numbered so that we may distinguish between different balls). Suppose each ball is painted one of k colours. How would you visualise a state of the system? Describe suitable sets Cons and Pred. Describe a term interpretation that does not represent a state of the system. Which term interpretations represent states?*
- *Consider an urn containing n balls, but assume that the balls are not numbered, so that balls of the same colour are indistinguishable. Suppose each ball is painted one of k colours. Now how would you visualise a state of the system? Describe suitable sets Cons and Pred. Describe a term interpretation that does not represent a state of the system. Which term interpretations represent states?*

4 Variables

We have discussed transparent propositional languages built up from atomic sentences that were induced by constants and predicate symbols. Such languages may usefully be interpreted in a very simple way, namely by term interpretations. Term interpretations, and transparent propositional languages, make sense when the system of interest has a *blueprint*.

By a blueprint we mean, intuitively, a way to link each member of a stable collection of components to a name, for example by drawing pictures of the components and labelling them. A system can have a blueprint only if its components are clearly distinguishable and do not fluctuate erratically the way the members of, say, the class of all sparrows change over a relatively brief interval due to births and deaths. The Light-Fan System can be blueprinted, of course. A petrie dish full of bacteria is not a blueprintable system, because the silly little things either die or divide every few minutes. A car is a blueprintable system, because you can imagine it being accompanied by a reference manual containing pictures of the components showing where they fit in. A town is a blueprintable system, because you can imagine drawing a map of it on which every place of interest is identified and its location relative to other places is shown. A business organisation is a blueprintable system, because you could design a database listing every employee together with relevant data about how each fits into the company hierarchy.

Although we have used the term 'blueprint', we shall not usually draw pictures or diagrams. Instead we try to represent states by means of abstract 'mathematical' pictures.

A term interpretation is a very simple way to represent a state of a blueprintable system. But not all systems are blueprintable. What would an unblueprintable system be like, and what sort of transparent language would suit it?

A system such as the set of all beetles is unblueprintable. Although it is a finite system, it is in a sense open-ended — somehow it seems impractical and pointless to attempt to give each component of the system a name of its own, because components are coming into existence and flickering out of existence all the time. The total number of components at a given moment is a random matter, not a matter of system design. In fact, design and structure are lacking in the system. To feel this, just contrast the vague collection of all beetles with the more structured collection of beetle specimens in a museum. The museum collection has the necessary stability and structure to be blueprintable — we can paste a label onto each of the beetles in the display cabinets. But we don't have names for all the elements of unblueprintable systems.

How do we cope in natural language when talking about objects for which we do not have names? We typically use pronouns like 'it' or 'he' or 'she'. We might point at a beetle and say 'It is small' or 'This is small'. Occasionally we have a way to uniquely identify the object by a name or a phrase that serves the same purpose as a name: 'Queen Elizabeth's pet beetle is small'. But normally we use a *variable* like 'it' or 'this', whose denotation has to be established by some form of pointing. In the absence of pointing, it is impossible to decide whether 'it is red' is true or false. Given some form of pointing, we can check whether the thing denoted by the variable 'it' really does have the property in question.

To build a language with variables is not hard. To the set $Cons$ of constants we add a set Var of variables, and we allow both constants and variables to be arguments of predicate symbols.

How many variables? Unblueprintable systems may be either finite or infinite, but even in the former case there is usually no neat upper limit on the number of components. In order to ensure that there are enough variables, it is customary to take the countably infinite set $Var = \{x_1, x_2, \ldots\}$.

What changes when we add variables to a language? Term interpretations are no longer very useful, because the system of interest is probably not blueprintable and so there will be components of the system that do not have names in the language. To use a general interpretation instead of a term interpretation, the domain has to be carefully specified. Also, an interpretation on its own will no longer be sufficient to ensure that every grammatically correct string of the new language has a truth value, because the denotation function doesn't say what the variables mean. An additional step is necessary, in which each variable is given a denotation inside the domain (like pointing to a particular beetle). Prior to this step, the grammatically correct strings will be of two kinds. Some strings will get truth values in the familiar way from the interpretation, and these we continue to call sentences. Others will be like equations — they contain variables and will be true for certain values of the variables, false for other values of the variables. We shall need a word to use for talking about strings of the language that may or may not be sentences, and we choose *well-formed formula*.

Definition 19 *Let $Var = \{x_1, x_2, \ldots\}$ and let $Cons$ be a set disjoint from Var. Let $Pred$ be a function consisting of pairs (P, n).*

By a ***term*** *we understand a constant or a variable, i.e. $Term = Cons \cup Var$.*

By the set of ***atomic formulas*** *induced by $Term$ and $Pred$ we understand the set A of all strings $P(t_1, t_2, \ldots, t_n)$ such that $(P, n) \in Pred$ and $t_1, t_2, \ldots, t_n \in Term$.*

Example 20 *Start with* $Cons = \{a, b\}$ *and* $Pred = \{(P, 1)\}$. *Now add* $Var = \{x_1, x_2, \ldots\}$. *Then* $Term = \{a, b, x_1, x_2, \ldots\}$ *and the set of atomic formulas is* $A = \{P(a), P(b), P(x_1), P(x_2), \ldots\}$.

In case you're wondering, we may still speak of atomic sentences — these are just the atomic formulas that contain no variables, i.e. that are built up from *Cons* and *Pred* as before.

Definition 21 *Let* A *be the set of atomic formulas induced by* $Term$ *and* $Pred$. *Then* α *is a propositional* ***well-formed formula*** *(abbreviated* ***wff****, pronounced 'woof') iff one of the following is the case:*

- $\alpha \in A$
- $\alpha = \neg\beta$, *for some previously constructed wff* β
- $\alpha = \beta * \gamma$, *where* β *and* γ *are previously constructed wffs and* $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

  If α *is a wff in which no variable occurs, we call* α *a* ***sentence****.*

  By L_A *we understand the set of all propositional wffs.*

Example 22 *Let* A *be as before. An example of a wff which is not a sentence is* $P(b) \vee \neg P(x_{25})$, *whereas* $P(b) \wedge \neg P(a)$ *is a sentence.*

Our definition speaks of 'propositional' wffs, because we have used only the familiar propositional connectives to build up wffs from atomic formulas. We could add new operators to the connectives and in that way get other kinds of wffs, such as modal wffs or (coming up next lecture) first-order wffs.

Definition 23 *Let* L_A *be the language of propositional wffs generated from* A, *where* A *is the set of atomic formulas induced by some* $Term$ *and* $Pred$.

- *As before, an* ***interpretation*** *of* L_A *is a pair* $\mathcal{D} = (D, den)$ *where the domain* D *is a nonempty set and the denotation function* den *assigns to each constant* $c \in Cons$ *a member* $den(c) \in D$ *and to each* $(P, n) \in Pred$ *a subset* $den(P, n)$ *of* D^n.
- *Given an interpretation* $\mathcal{D}$ *of* L_A, *a* ***variable assignment*** *is a function* $s : Var \longrightarrow D$.
- *An interpretation* $\mathcal{D}$ *together with a variable assignment* s *is called a* ***state*** $(\mathcal{D}, s)$.

A variable assignment is what must be added to an interpretation in order to ensure that all terms have denotations and that we know what each wff is talking about. Thus a variable assignment is the analog of pointing at anonymous (unnamed) components of a system. The effect of adding a variable assignment to an interpretation is to give a context that makes it clear what each variable means, and this will determine a valuation. First we practice.

Example 24 *Let L_A be generated from $A = \{P(a), P(b), P(x_1), P(x_2), \ldots\}$ as before. One possible interpretation of L_A is $\mathcal{D} = (D, den)$ with $D = \{1, 2, 3\}$, $den(a) = 1$, $den(b) = 2$, and $den(P, 1) = \{1, 3\}$.*

Relative to $\mathcal{D}$ there are many variable assignments. Some are quite simple-minded, like the function $s_1 : Var \longrightarrow \{1, 2, 3\}$ given by $s_1(x) = 2$ for all $x \in Var$. Others have more variety, like the function $s_2 : Var \longrightarrow \{1, 2, 3\}$ given by $s_2(x_i) = 1$ if i is odd and $s_2(x_i) = 2$ if i is even. We notice that neither s_1 nor s_2 is surjective, which means that there are elements of D that never get the chance to be values of variables. But there is nothing to stop a variable assignment from being surjective like the function $s_3 : Var \longrightarrow \{1, 2, 3\}$ given by $s_3(x_1) = 1$, $s_3(x_2) = 2$, and $s_3(x_i) = 3$ if $i > 2$.

States are the engines that drive the allocation of truth values.

Definition 25 *Let L_A be the language of propositional wffs generated from A, where A is the set of atomic formulas induced by some $Term$ and $Pred$.*

Let $\mathcal{D} = (D, den)$ be any interpretation of L_A and $s : Var \longrightarrow D$ a variable assignment.

*The **valuation determined by** $(\mathcal{D}, s)$ is the function $v : A \longrightarrow \{0, 1\}$ such that for every $P(t_1, t_2, \ldots, t_n) \in A$, $v(P(t_1, t_2, \ldots, t_n)) = 1$ iff $(d_1, d_2, \ldots, d_n) \in den(P, n)$ where, for every $i \leq n$, $d_i = den(t_i)$ if $t_i \in Cons$ and $d_i = s(t_i)$ if $t_i \in Var$.*

By W_A we mean the set of all valuations of L_A.

The idea is that *den* is used to find the denotations of constants and s the denotations of variables. Consider in turn the examples of variable assignments given above.

Example 26 *The valuation v_1 determined by $\mathcal{D}$ and s_1 makes $v_1(P(a)) = 1$ since $den(a) = 1 \in den(P, 1)$, makes $v_1(P(b)) = 0$ since $den(b) = 2 \notin den(P, 1)$, and for each $x_i \in Var$, makes $v_1(P(x_i)) = 0$ since $s_1(x_i) = 2 \notin den(P, 1)$.*

The valuation v_2 determined by $\mathcal{D}$ and s_2 makes $v_2(P(a)) = 1$ since $den(a) \in den(P, 1)$, makes $v_2(P(b)) = 0$ since $den(b) \notin den(P, 1)$, and

for each $x_i \in Var$, makes $v_2(P(x_i)) = 0$ if i is even, since then $s_2(x_i) = 2 \notin den(P,1)$, while $v_2(P(x_i)) = 1$ if i is odd, since then $s_2(x_i) = 1 \in den(P,1)$.

The valuation v_3 determined by $\mathcal{D}$ and s_3 makes $v_3(P(a)) = 1$ since $den(a) \in den(P,1)$, makes $v_3(P(b)) = 0$ since $den(b) \notin den(P,1)$, and for each $x_i \in Var$, makes $v_3(P(x_i)) = 1$ if $i = 1$, since $s_3(x_1) = 1 \in den(P,1)$, makes $v_3(P(x_i)) = 0$ if $i = 2$, since $s_3(x_2) = 2 \notin den(P,1)$, while $v_3(P(x_i)) = 1$ if $i > 2$, since then $s_3(x_i) = 3 \in den(P,1)$.

Once we have a valuation, we know how truth values are allocated to wffs. Of course, we are interested only in the valuations determined by states.

Definition 27 *Let L_A be the language of propositional wffs generated from A, where A is the set of atomic formulas induced by some $Term$ and $Pred$.*

Let $\mathcal{D} = (D, den)$ be any interpretation of L_A and let $s : Var \longrightarrow D$ be a variable assignment in $\mathcal{D}$. Let v be the valuation determined by the state $(\mathcal{D}, s)$.

We say that the state $(\mathcal{D}, s)$ satisfies a wff α, or that s satisfies α in $\mathcal{D}$, or that v satisfies α, according to the following rules:

- *s satisfies an atomic formula $P(t_1, t_2, \ldots, t_n)$ in $\mathcal{D}$ iff $(d_1, d_2, \ldots, d_n) \in den(P,n)$ where, for every $i \leq n$, $d_i = den(t_i)$ if $t_i \in Cons$ and $d_i = s(t_i)$ if $t_i \in Var$*
- *s satisfies $\neg\beta$ iff s fails to satisfy β*
- *s satisfies $\beta \wedge \gamma$ iff s satisfies both β and γ*
- *s satisfies $\beta \vee \gamma$ iff s satisfies at least one of β and γ*
- *s satisfies $\beta \rightarrow \gamma$ iff s satisfies γ or failed to satisfy β*
- *s satisfies $\beta \leftrightarrow \gamma$ iff s satisfies both β and γ or neither.*

Note that if α is a sentence, i.e. has no variables occurring in it, then the truth value assigned to α relative to an interpretation $\mathcal{D}$ is independent of the choice of variable assignment — if any variable assignment satisfies α in $\mathcal{D}$, then all variable assignments satisfy α in $\mathcal{D}$. (To see this, observe that in atomic formulas it is only the variables whose denotations are affected by the choice of variable assignment.)

Example 28 *Returning to our previous example, s_1 satisfies $\neg P(x_{100})$ in $\mathcal{D}$ and s_3 satisfies $P(a) \wedge P(x_{100})$ in $\mathcal{D}$. The sentence $P(a)$ is satisfied by every variable assignment in $\mathcal{D}$, since $den(a) = 1$ and $1 \in den(P, 1)$ regardless of the variable assignment.*

Now that we have defined satisfaction, we need to define the notions of ontology and model.

Definition 29 *Let L_A be the language of propositional wffs generated from A, where A is the set of atomic formulas induced by some $Term$ and $Pred$.*

An ontology for L_A is a pair (S, V) where S is some set of states $(\mathcal{D}, s)$ and $V : S \longrightarrow W_A$ is the obvious labelling function taking a state $(\mathcal{D}, s)$ to the valuation v determined by $(\mathcal{D}, s)$.

Let $\mathcal{D} = (D, den)$ be any interpretation of L_A and let $\alpha \in L_A$.

*If $s : Var \longrightarrow \mathcal{D}$ is a variable assignment satisfying α in $\mathcal{D}$, then $(\mathcal{D}, s)$ is a **local model** of α, and so is the valuation determined by $(\mathcal{D}, s)$.*

We write $\mathcal{M}(\alpha)$ for the subset of S consisting of all local models of α (or equivalently we may write $\mathcal{M}(\alpha)$ for the subset of W_A consisting of all valuations determined by states in S that satisfy α).

*If every variable assignment $s : Var \longrightarrow D$ satisfies α in $\mathcal{D}$, then we call $\mathcal{D}$ a **global model** of α.*

Why do we have two notions of model? Local models are more closely associated with valuations. But global models are interesting too. Just for a moment forget about the object languages we've defined and think of what one does with equations in mathematics. Equations are used in two ways. Given an equation like $x^2 = 9$ we may seek the values of x which 'satisfy' the equation. This is like finding the variable assignments or local models that satisfy a wff. We use local models when we have a wff and want to find out under what conditions that wff will be true. On the other hand, given an equation like $x + y = y + x$, we are most likely going to use it as an 'identity', in other words we have in mind some domain like the set of integers in which the equation is satisfied for **all** values of x and y. We would use global models (as opposed to local models) when we have a wff and would like to find out under what interpretations the wff will be 'identically' true.

Let's explore the new notion of global model.

Example 30 *Returning to our previous example, $\mathcal{D}$ is a global model of $\neg P(b)$ since every variable assignment satisfies the wff $\neg P(b)$ in $\mathcal{D}$. To verify this is not hard. Whatever the assignment, say s, the satisfaction*

of $P(b)$ depends on whether $den(b) \in den(P,1)$, which resolutely refuses to be the case quite independently of s, and so s satisfies $\neg P(a)$ in $\mathcal{D}$.

Is $\mathcal{D}$ a global model of $P(a)$? Indeed it is, because every assignment satisfies $P(a)$ in $\mathcal{D}$. As in the case of b above, a is a constant and $den_2(a) = 1 \in den_2(P,1)$ irrespective of the assignment.

On the other hand, $\mathcal{D}$ is not a global model of $P(x_1)$. Some assignments do satisfy this wff, for instance s_2 where $s_2(x_i) = 1$ if i is odd and $s_2(x_i) = 2$ if i is even, because $s_2(x_1) = 1 \in den(P,1)$. But not every assignment will satisfy $P(x_1)$ in $\mathcal{D}$. For instance, if we define s_4 by requiring that $s_4(x_i) = 2$ for $i \leq 1000$ and $s_4(x_i) = 3$ for $i > 1000$, then s_4 fails to satisfy $P(x_1)$ in $\mathcal{D}$.

When variables occur in a wff, it becomes possible for the choice of assignment to change the truth value assigned to the wff. But the examples illustrate a surprising and satisfying fact — the truth value of a sentence (a wff without variables) depends only on the interpretation and is unaffected by the choice of assignment. We see a difference between states and interpretations echoing the difference between wffs and sentences. A sentence divides the set of all interpretations into two complementary subsets consisting respectively of the global models of the sentence and those that are global models of its negation. Similarly a wff containing variables divides the set of states into a set of local models and the complementary set of local nonmodels. But a wff containing variables divides the set of interpretations into three disjoint subsets — those that are global models of the wff, those that are global models of its negation, and those that are 'undecided'.

The notions of local model and global model would allow us to define two different classical entailment relations by analogy with previous definitions in simpler languages, but we shall stick to local models.

Definition 31 *$\alpha \vDash \beta$ iff $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$.*

Exercise 32 *In each of the following, assume that the language contains variables.*

1. *Suppose $Cons = \{a, b\}$ and $Pred = \{(P,1)\}$ so that $Term = \{a, b, x_1, x_2, \ldots\}$ and $A = \{P(a), P(b), P(x_1), P(x_2), \ldots\}$. Describe an interpretation of L_A in which the domain has just one element. Write down all variable assignments. Write down a wff of which the interpretation is a global model. Give details of your reasoning.*

2. *Suppose $Cons = \{a\}$ and $Pred = \{(=,2)\}$, in other words the predicate symbol is intended to represent equality. Write down a wff α such that every global model of α is an interpretation whose domain contains exactly one member.*

3. *Suppose* $Cons = \{a\}$ *and* $Pred = \{(P,1),(Q,1)\}$.

   - *Describe an interpretation* $\mathcal{D}_1 = (D_1, den_1)$ *which is a global model of* $P(a) \wedge \neg Q(a)$ *but not of* $P(x_1) \wedge \neg Q(x_2)$. *Give details of your reasoning.*
   - *Describe an interpretation* $\mathcal{D}_2 = (D_2, den_2)$ *which is a global model of* $P(x_1) \wedge \neg Q(x_2)$. *Give details.*
   - *Let* $\mathcal{D} = (D, den)$ *be the interpretation with* $D = \{0,1,2,\ldots\}$, $den(a) = 0$, $den(P,1) = \{n \mid n \text{ is even}\}$, *and* $den(Q,1) = \{n \mid n \text{ is odd}\}$. *Find variable assignments* s *and* s' *which respectively satisfy and fail to satisfy the wff* $P(x_4)$ *in* $\mathcal{D}$. *Is* $\mathcal{D}$ *a global model of the wff* $P(x_4) \vee Q(x_4)$? *What about* $P(x_4) \vee Q(x_5)$?

4. *Suppose* $Cons = \{a,b\}$ *and* $Pred = \{(P,2)\}$.

   - *Describe an interpretation* $\mathcal{D}_1 = (D_1, den_1)$ *which is a global model of* $P(a,b) \wedge \neg P(b,a)$.
   - *Describe an interpretation* $\mathcal{D}_2 = (D_2, den_2)$ *which is a global model of* $P(x_1,x_1) \wedge \neg P(a,b)$. *Give details of your reasoning.*
   - *Let* $\mathcal{D} = (D, den)$ *be the interpretation with* $D = \{0,1,2,\ldots\}$, $den(a) = 0$, $den(b) = 1$, *and* $den(P,2) = \{(n,m) \mid n \leq m\}$. *Find variable assignments* s *and* s' *which respectively satisfy and fail to satisfy the wff* $P(b,x_1)$. *Is* $\mathcal{D}$ *a global model of the wff* $P(a,x_1)$? *What about* $P(b,x_1) \vee P(x_1,b)$?
   - *Find an example of wffs* α *and* β *such that* $\alpha \models \beta$.

5 Function symbols

Next we consider a way in which to represent knowledge of dependency. The trick applies to both blueprintable and unblueprintable systems, but for simplicity we begin by considering the blueprintable case. Imagine that the simple Light-Fan System, which has just two components, is augmented by the addition of two software switches that allow you to control the light and fan via the Internet. The new Light-Fan-Switches System has four components — the light, the fan, a (software) switch to put the light on or off, and a (software) switch to put the fan on or off.

It would be possible to represent knowledge about this new system by employing the approach described in previous sections. We could take $Cons$ to be a set of names for the four components, say $Cons = \{L, F, S_L, S_F\}$. Together with the predicate symbol $IsOn$ four

atoms are generated, and the sixteen term interpretations of the language determine the various ways in which truth values can be allocated to these atoms.

But consider — the new augmented system was not obtained by adding two arbitrary new components. Each of the new components in some sense 'belongs' to one of the old, i.e. depends for its existence upon one of the old components. Also, the new components were added in a uniform way, each old component being supplemented by the same sort of thing, namely a switch. So there is a single relationship of dependence of new components upon old, rather than a random collection of different relationships. And finally, each of the new components can be uniquely identified by mentioning which old component it depends upon. Thus the relationship of dependence between new components and old is, mathematically speaking, a function.

To each old component, there is assigned exactly one new component. This assignment may be represented by the function f where $f(L) = S_L$ and $f(F) = S_F$. Functional notation provides convenient alternative names for the switches, for we may write $f(L)$ as an alternative name for S_L and $f(F)$ as an alternative name for S_F. Not only do these alternative names make explicit the dependence of the new component upon the old, but we can talk about, say, the fan's switch without even knowing that it is component named S_F that we are talking about. This is a subtle trick routinely used in everyday life, as when we speak of 'The chap whose office is opposite the fire escape' without needing to know his name, or 'The capital of Sri Lanka' without needing to consult an atlas to find out that it is called Colombo. This is particularly convenient for discussions of global politics, as we may speak of 'the president of the USA' or of 'the prime minister of Britain' without knowing which morally challenged intellectual pygmy is currently desecrating that office.

Phrases such as 'the fan's switch' or 'the capital of Sri Lanka' are *definite descriptions*. In natural languages like English, definite descriptions are noun phrases that involve the definite article 'the', and which therefore pick out a single object of some kind. A noun phrase using the indefinite article, such as 'a noun phrase' would not be a definite description, because it doesn't pick out a unique thing. Functions offer a simple way to include definite descriptions in the knowledge representation language, though not the only way. The notation $f(L)$ is a formal analog of the English phrase 'the light's switch'. More generally, such notation makes it possible to write $f(x)$ as an analog of the English phrase 'its switch', which prepares the way for the expression of such potentially useful but slightly vague ideas as 'Some component's switch is off'.

Before stipulating precisely how the object language is to be equipped with symbols that will furnish analogs for definite descriptions in the metalanguage, let us look at three more examples of systems in which some components have this kind of functional relationship with other components.

Let's model the local government of some town. Voters have representatives on the town council, elected according to a system of neighbourhoods or precincts, so that a number of voters have the same representative. The relationship between voters and councillors is functional, each voter being represented by a unique councillor. And the councillor depends on her voters, for Councillor Smith represents the voters of one precinct while Councillor Crowe represents a different precinct. Suppose the function g indicates the relationship of dependence. Then we can speak of the councillor representing a particular voter without needing to know that councillor's name. If the constant d_1 in the object language stands for the elderly Mrs Mackenzie of the quiet suburb Victoria Park, then $g(d_1)$ is her councillor, whomever that may be.

Or suppose you inherit vast wealth and decide to breed racehorses. You will wish to keep track of the sires and dams of your horses. Each horse has exactly one sire, so that the relationship between horses and their sires can be encoded as a function. Similarly, each horse has exactly one dam, giving another functional relationship.

Suppose, finally, that you while away the hours between visits to your stables by pursuing an interest in mathematics, and that your interest is in the set of integers. For every integer n there is a next integer, bigger by 1, called the successor of n. The relationship between integers and their successors is functional. A similar functional relationship obtains between integers and their immediate predecessors. And the relationship between pairs of integers and their sum is also functional.

Having established that systems often contain functional dependencies, let us now consider how to encode these in the object language.

Definition 33 *Let the sets $Cons$, Var, and $Pred$ be given. Let Fun be any function consisting of ordered pairs (f, n), where n is a natural number called the arity of the function symbol f.*

The set $Term$ of terms consists of all strings t such that one of the following is the case:

- $t \in Cons$
- $t \in Var$
- $t = f(t_1, t_2, \ldots, t_n)$ *where* $(f, n) \in Fun$ *and* $t_1, t_2, \ldots, t_n$ *are previously generated terms.*

Given the set $Term$, the next step is to define the set of atomic formulas and then the set of well-formed formulas. This is routine.

Definition 34 *An atom is a string of the form $P(t_1, t_2, \ldots, t_m)$ where $(P, m) \in Pred$ and $t_1, \ldots, t_m$ are terms.*

Wffs are built up from the atoms by connectives in the usual way.

If the intention were to construct a knowledge representation language for a blueprintable system, the set Var would be omitted and neither the terms nor the atoms of the resulting language would contain variables. The inclusion or omission of function symbols is independent of the inclusion or omission of variables, and depends only on whether we, the logicians, want to take advantage of a functional relationship between components.

Let us examine an example of a language with function symbols.

Example 35 *Take $Cons = \{a\}$, $Var = \{x_1, x_2, \ldots\}$, $Pred = \{(P, 1)\}$, and $Fun = \{(f, 1)\}$. The set of terms is built up in stages. In the first stage, the following are included: $a, x_1, x_2, \ldots$ During the second stage the following more complex terms are added to those constructed previously: $f(a), f(x_1), f(x_2), \ldots$ The third stage adds $f(f(a)), f(f(x_1)), f(f(x_2)), \ldots$ In this fashion terms of tremendous complexity may be built up, in a million steps or more. The set $Term$ consists of all the terms that can be constructed in finitely many steps, no matter how long it may take to actually do so. Using $Term$, the atoms of the language are the following, where we arrange them according to the complexity of the terms appearing in them:*

$P(a), P(x_1), P(x_2), \ldots$
$P(f(a)), P(f(x_1)), \ldots$
$P(f(f(a))), P(f(f(x_1))), \ldots$
And so on.

In the definition of $Term$ we have permitted Fun to contain pairs of the form $(f, 0)$. What is a function of arity 0?

Well, let's put it in perspective first. A *binary* function is a function $f : X \longrightarrow Y$ where $X = Y_1 \times Y_2$ for some sets Y_1 and Y_2. In general an n-ary function is a function of the form $f : X \longrightarrow Y$ where $X = Y_1 \times Y_2 \times \ldots Y_n$ so that the inputs to the function are n-tuples $(y_1, y_2, \ldots, y_n)$ with $y_i \in Y_i$. Often we are interested in the special case of $Y = Y_1 = Y_2 = \ldots = Y_n$. A function $f : Y^n \longrightarrow Y$ is called an n-ary *operation* on Y, and it is easy to find examples in which $n = 1, 2$, or even 3. In order to attach meaning to the case where $n = 0$ we need to know that in set theory $Y^0 = \{\varnothing\}$, no matter what the set Y may be. So a function

$f : Y^0 \longrightarrow Y$ has just one (uninteresting) input, and in effect selects a member of the set Y (the output value $f(\varnothing)$). Note that the 0-ary function f doesn't actually take a member of Y as input argument.

Why might 0-ary functions be useful? Well, the basic idea is that a term is an expression of the language that, when the language is interpreted, will denote an individual object in the domain. For function symbols of arity $n \geq 1$, the terms built up with their aid depend for their denotations upon the denotations of other terms. For example, the term $f(a)$ depends for its denotation upon a previous decision about the denotation of the constant a. But, like a constant, a 0-ary function f is interpreted directly, its denotation being independent of the denotations assigned to other terms. Thus you may think of constants as really being 0-ary functions, so that languages containing constants constitute a special case of languages containing function symbols.

Let us formalise the idea that terms should denote individual members of the domain of interpretation.

Definition 36 *Let L_A be the set of propositional wffs built up with the aid of the usual connectives from A, where A is the set of atomic formulas induced by $Term$ and $Pred$, and $Term$ is built up with the help of Fun.*

An interpretation of L_A is a pair $\mathcal{D} = (D, den)$ where the domain D is a nonempty set and the denotation function den is such that

- *den assigns to every $c \in Cons$ a member $den(c) \in D$*
- *den assigns to every $(f, n) \in Fun$ an n-ary operation $den(f, n) : D^n \longrightarrow D$*
- *den assigns to every $(P, m) \in Pred$ a subset $den(P, m) \subseteq D^m$.*

Given an interpretation $\mathcal{D} = (D, den)$, a variable assignment is, as before, a function $s : Var \longrightarrow D$. We continue to refer to a pair $(\mathcal{D}, s)$ as a state.

An interpretation tells us what the constants denote, and in fact does more — it tells us what the denotations are of all 'ground' terms, the terms built up without the use of variables. As one would expect, the denotations of terms containing variables are determined by a variable assignment s.

Example 37 *Let L_A be the language built up from $Cons = \{a\}$, $Var = \{x_1, x_2, \ldots\}$, $Pred = \{(P, 1)\}$, and $Fun = \{(f, 1)\}$. We can give two very different interpretations $\mathcal{D}$ and $\mathcal{E}$ of L_A.*

Take $\mathcal{D} = (D, den_D)$ to have $D = \{113\}$ and den_D such that $den_D(a) = 113$, $den_D(P, 1) = \emptyset$, and $den_D(f, 1) : D \longrightarrow D$ the function $\{(113, 113)\}$. Thus the domain D has a single member, the number 113. And so the denotation function den_D must make the constant a act as a name for 113. Note also that there is only one function from $\{113\}$ to $\{113\}$, namely that which assigns the output 113 to the input 113. Thus den_D has no alternative but to make this function the denotation of the function symbol f. Now we can say what the denotation of a term like $f(a)$ is — we simply go to the function denoted by f, feed it the thing denoted by a, and see what it spits out. In this case, $den_D(f, 1)$ is the function which assigns to the input $den_D(a) = 113$ the corresponding output $den_D(f, 1)(113) = 113$. So $f(a)$ denotes 113 too. So does $f(f(a))$. So does $f(f(f(a)))$. And so forth.

In contrast, let $\mathcal{E} = (E, den_E)$ be such that $E = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, the set of all integers, and den_E such that $den_E(a) = 0$, $den_E(P, 1) = \{n \mid n \text{ is even}\}$, and $den_E(f, 1) : E \longrightarrow E$ is the successor function $\{(n, n + 1) \mid n \in E\}$. What does the term $f(a)$ denote in this new interpretation? Well, go to the function denoted by f, and feed it the thing denoted by a. The function symbol f denotes the successor function $\{(n, n + 1) \mid n \in E\}$, and the constant a denotes the number 0, so $f(a)$ denotes the successor of 0, namely the number 1. Similarly $f(f(a))$ denotes 2, $f(f(f(a)))$ denotes 3, and so on.

As far as terms containing variables are concerned, the procedure is similar but relative to a particular assignment. Consider again the interpretation $\mathcal{E}$, and let s be the assignment assigning to every variable x_i the element $i - 100$ in E. What does the term $f(x_1)$ denote? Well, $s(x_1) = 1 - 100 = -99$, and $den_E(f, 1)$ is the successor function, so $f(x_1)$ denotes the number $-99 + 1 = -98$.

From states we get valuations in the usual way.

Definition 38 *The valuation determined by a state $(\mathcal{D}, s)$ is the function $v : A \longrightarrow \{0, 1\}$ defined as follows:*

For every $P(t_1, t_2, \ldots, t_m) \in A$, the valuation v assigns the truth value 1 iff $(d_1, d_2, \ldots, d_m) \in den(P, m)$, where each d_i is the member of D denoted by the term t_i and is calculated by distinguishing between three cases:

- *if t_i is a constant, say c, then $d_i = den(c)$*
- *if t_i is a variable, say x, then $d_i = s(x)$*
- *if t_i is a complex term of the form $f(r_1, \ldots, r_n)$, where $r_1, \ldots, r_n$ are terms and $(f, n) \in Fun$, then we first calculate the denotations of $r_1, \ldots, r_n$ and then apply the function $den(f, n)$ to them.*

The definition looks more complicated than it really is. Just keep firmly in mind that complex terms are built up recursively, starting from constants and variables. The denotations of constants are revealed to us by *den*, and the denotations of variables are revealed by s. With this as starting point, we can work out the denotation of any complex term just by recapitulating the steps by which the complex term was constructed.

Example 39 *Let L_A be the familiar language built up from $Cons = \{a\}$, $Var = \{x_1, x_2, \ldots\}$, $Pred = \{(P, 1)\}$, and $Fun = \{(f, 1)\}$. Take the interpretation $\mathcal{E} = (E, den_E)$ with $E = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, $den_E(a) = 0$, $den_E(P, 1) = \{n \mid n \text{ is even}\}$, and $den_E(f, 1) : E \longrightarrow E$ the successor function $\{(n, n+1) \mid n \in E\}$. Let s be the assignment sending each variable x_i to the number $i - 100$.*

What truth value does the corresponding valuation v assign to the atom $P(a)$? The term a denotes the number $den_E(a) = 0$, and $0 \in den_E(P, 1)$ because 0 is even, so $v(P(a)) = 1$.

What about the truth value of $P(f(x_2))$? Well, does the number denoted by $f(x_2)$ belong to the set $den_E(P, 1) = \{n | n \text{ is even}\}$? Since $s(x_2) = 2 - 100 = -98$, and $den_E(f, 1)$ is the successor function, the denotation of $f(x_2)$ is -97, which is not even and so does not belong to $den_E(P, 1)$. Thus v assigns to $P(f(x_2))$ the truth value 0.

Next we define local models and global models.

Definition 40 *As before, an ontology is a pair (S, V) where S is a collection of states $(\mathcal{D}, s)$ and $V : S \longrightarrow W_A$ is the obvious labelling function that takes a state to the valuation it determines.*

Truth values are allocated to the wffs of L_A in usual manner, starting from the truth values allocated to atomic formulas by the valuation v determined by $\mathcal{D}$ and s, and treating the connectives in the familiar way.

If the truth value 1 is assigned to a wff α, then the assignment s satisfies α in the interpretation $\mathcal{D}$ and we call $(\mathcal{D}, s)$ a local model of α (and we so call also the valuation v).

$\mathcal{M}(\alpha)$ is the subset of S consisting of local models of α.

If every assignment s satisfies α in $\mathcal{D}$, then $\mathcal{D}$ is a global model of α.

So the conditions for satisfaction are the same as for languages without function symbols. The additional complexity introduced by function symbols has its effect at the level of atomic formulas and valuations, because the assignment of a truth value to an atomic formula requires us to find out what the terms denote, and this becomes more of an effort when there are function symbols around. Once truth values have been allocated to atomic formulas, everything proceeds as normal.

Exercise 41 *In each of the following, assume that the language has variables.*

1. *Let L_A be the language with $Cons = \{a, b\}$, $Pred = \{(P, 1), (Q, 1)\}$, and $Fun = \{(f, 1), (g, 1)\}$.*

   - *Describe, as best you can, the terms of L_A.*
   - *Describe, as best you can, the atomic formulas of L_A.*
   - *Give an interpretation $\mathcal{D}$ of L_A whose domain has exactly 3 members.*
   - *Give an assignment s in $\mathcal{D}$ and the valuation determined by $(\mathcal{D}, s)$.*
   - *Give one example of a wff having $\mathcal{D}$ as a global model.*

2. *Let L_A be the language with $Cons = \{a\}$, $Pred = \{(P, 2)\}$, and $Fun = \{(f, 2)\}$.*

   - *Describe, as best you can, the terms of L_A.*
   - *Describe, as best you can, the atomic formulas of L_A.*
   - *Give an interpretation $\mathcal{D}$ of L_A having as domain a set with infinitely many members.*
   - *Give an assignment s in $\mathcal{D}$ and the valuation determined by $(\mathcal{D}, s)$.*
   - *Give an example of a wff satisfied by all assignments in $\mathcal{D}$, an example of a wff satisfied by some assignments in $\mathcal{D}$ but not by others, and an example of a wff satisfied by no assignment in $\mathcal{D}$.*

6 Sorts

The use of function symbols to build complex terms has a sometimes inconvenient aspect — a function symbol is applied over and over, perhaps producing more new terms than needed or wanted. Let us contrast two systems.

Suppose the system of interest is the set of natural numbers. One of the most characteristic features of the system is the functional relationship between natural numbers and their successors. If we represent this relationship by a function symbol $(s, 1)$, then complex terms like $s(s(s(x)))$ can be built, and these make perfect sense. For example, $s(s(s(x)))$ stands for 'the successor of the successor of the successor of

x', which we know is just the number obtained by adding 3 to x, whatever x may be.

On the other hand, consider the augmented Light-Fan-Switches System which contains a switch for each of the two original components. An important feature of the system is the functional relationship between each of the 'old' components (the light and the fan) and the 'new' components (their switches). If we represent this relationship by a function symbol $(f, 1)$, then the language not only gains terms like $f(x)$ standing for 'the switch of x' but also terms like $f(f(f(x)))$ which have no obvious meanings as far as the intended system is concerned. After all, the switch of the fan doesn't itself have a switch that in turn has a switch.

A nice way to prevent such meaningless repetition is to use a *many-sorted* language. In a many-sorted language there are different 'sorts' or 'types' of constants and variables, which may be thought of as denoting different categories of things. Function symbols also have a sort associated with them, stipulating the sort of input argument to which the function symbol may be applied and the sort of term which results from the application. Since the input and output sorts may be different, it follows that iterated application can be blocked.

As an illustration, consider the Light-Fan-Switches System. We build a many-sorted language for it. Let us agree that there are two sorts of terms, either 'old component' or 'new component'. Let constants L and F be of sort 'old component' and let Fun consist of a function symbol $(f, 1)$ that eats arguments of sort 'old component' and spits out terms of sort 'new component'. Now $f(L)$ is a legitimate term of sort 'new component', and it becomes illegal to apply f again to produce terms like $f(f(L)), f(f(f(L))), \ldots$

Definition 42 *Suppose a set Sort is given, and that its members are called sorts (or types). A many-sorted alphabet has sorts allocated to the sets Cons, Var, Fun, Term, and Pred as follows:*

- *for each sort $\sigma \in Sort$, there is a (possibly empty) subset of Cons containing the constant symbols of sort σ*
- *for each sort $\sigma \in Sort$, there are infinitely many variables of sort σ*
- *for every k+1-tuple $\langle \pi_1, \ldots, \pi_{k+1} \rangle$ of sorts, there is a (possibly empty) subset of Fun containing the k-ary function symbols (f, k) of sort $\langle \pi_1, \ldots, \pi_{k+1} \rangle$*
- *if the terms $t_1, t_2, \ldots, t_k$ are of sorts $\pi_1, \pi_2, \ldots, \pi_k$ respectively, and the function symbol (f, k) has the sort $\langle \pi_1, \ldots, \pi_{k+1} \rangle$, then the complex term $f(t_1, \ldots, t_k)$ has the sort π_{k+1}*

- *for every n-tuple $\langle \pi_1, \ldots, \pi_n \rangle$ of sorts, there is a (possibly empty) subset of Pred containing the n-ary predicate symbols (P, n) of sort $\langle \pi_1, \ldots, \pi_n \rangle$.*

Sorts are merely labels, and so the elements of *Sort* may be anything — numbers, names, whatever.

To keep track of the different sorts of variables, we may either use different letters, with $x_1, x_2, \ldots$ for one sort and $y_1, y_2, \ldots$ for another sort, or we may write the sort as a superscript: $x_1^\pi, x_2^\pi, \ldots$

The sort $\langle \pi_1, \ldots, \pi_{k+1} \rangle$ associated with a function symbol (f, k) tells us the sorts $\pi_1, \ldots, \pi_k$ of the k input arguments and the sort π_{k+1} of the output.

Example 43 *In order to represent knowledge of the Light-Fan-Switches system, we might use a set Sort with two members, O (for 'Old') and N (for 'New'). Let $Cons = \{L, F\}$ where both L and F are of sort O. Let Var consist of $x_1, x_2, \ldots$ of sort O and $y_1, y_2, \ldots$ of sort N. Let $Fun = \{(f, 1)\}$ and, in order to reflect the intention that f should stand for 'the switch of', associate with f the sort $\langle O, N \rangle$. Let Pred $= \{(P, 1), (Q, 1)\}$ and associate with P the sort O, with Q the sort N, where the intention is that P tells us whether the light and fan are on, and Q tells us whether the switches are on. So $P(L)$ says that the light is shining, and $Q(f(F))$ says that the fan's switch is in the 'on' position. Note the sort of the complex term $f(F)$. Since f is of sort $\langle O, N \rangle$ and the input argument F is of sort O, the output $f(F)$ is of sort N. Further application of f to produce terms such as $f(f(F))$ is blocked, since the sort of $f(F)$ is not that of input arguments to which f may be applied.*

We have not yet said what the purpose is of the sort $\langle \pi_1, \ldots, \pi_n \rangle$ of a predicate symbol (P, n).

Definition 44 *The set A of atomic formulas induced by a many-sorted alphabet consists of all strings $P(t_1, t_2, \ldots, t_n)$ where (P, n) is a predicate symbol of sort $\langle \pi_1, \ldots, \pi_n \rangle$ and each term t_i is of sort π_i.*

We notice that the use of sorts not only tends to cut down on the proliferation of complex terms, it also has a constraining effect on atoms, because we can no longer attach a predicate symbol to arbitrary terms.

Definition 45 *A many-sorted transparent propositional language is the set of all well-formed formulae over the set of atomic formulas A induced by a many-sorted alphabet, where a (propositional) wff is a string α which is one of the following:*

- *an atomic formula*
- *of the form* $\neg\beta$ *where* β *is a wff*
- *of the form* $(\beta * \gamma)$ *where* β *and* γ *are wffs and* $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Next we define interpretations, and it is here that the effect of sorts is seen most clearly. In the domain of interpretation, there will be subdomains corresponding to the sorts and a term of sort σ will denote an object in the appropriate subdomain.

Definition 46 *An interpretation of a many-sorted language is a pair* $\mathcal{D} = (D, den)$ *such that for every sort* $\sigma \in Sort$ *there is a nonempty subset* $D_\sigma \subseteq D$, *and*

- *every constant symbol* c *of sort* σ *denotes an element* $den(c) \in D_\sigma$
- *every function symbol* (f, k) *of sort* $\langle \pi_1, \ldots, \pi_{k+1} \rangle$ *denotes a function* $den(f, k) : D_{\pi_1} \times \ldots \times D_{\pi_k} \longrightarrow D_{\pi_{k+1}}$
- *every predicate symbol* (P, n) *of sort* $\langle \pi_1, \ldots, \pi_n \rangle$ *denotes a subset* $den(P, n) \subseteq D_{\pi_1} \times \ldots \times D_{\pi_n}$.

Given $\mathcal{D}$, *a variable assignment is a function* $s : Var \longrightarrow D$ *such that if* x *is a variable of sort* σ, *then* $s(x) \in D_\sigma$.

Example 47 *Here is an interpretation of the many-sorted language of the Light-Fan-Switches System. Let* $D = \{1, 2, 3, 4\}$. *Since* $Sort = \{O, N\}$, *we need to specify two subdomains of* D. *Let* $D_O = \{1, 2\}$ *and let* $D_N = \{3, 4\}$. *Now let den be such that* $den(L) = 1$ *and* $den(F) = 2$. *Moreover let* $den(f, 1)$ *be the function from* D_O *to* D_N *given by* $den(f, 1)(1) = 3$ *and* $den(f, 1)(2) = 4$. *And let* $den(P, 1) = \emptyset$ *and* $den(Q, 1) = \emptyset$.

This interpretation is a semantic picture of the Light-Fan-Switches system in a particular state. The light and the fan are modelled by the elements 1 and 2 of D_O, the subdomain of old components. Their switches are modelled by 3 and 4 respectively. The denotations of P and Q tell us that the light is not shining, the fan is not spinning, and the switches are both in the 'off' position. Intuitively, this interpretation makes wffs like $P(L)$ and $Q(f(L))$ false. In order to allocate a truth value to a wff such as $P(x_1)$ we would need to specify an assignment s. Let us formalise our intuition regarding the allocation of truth values.

Definition 48 *An interpretation $\mathcal{D}$ and an assignment s together determine a valuation $v : A \longrightarrow \{0, 1\}$ which assigns to an atomic formula $P(t_1, t_2, \ldots, t_n)$ the truth value 1 iff $(d_1, d_2, \ldots, d_n) \in den(P, n)$ where each d_i is the denotation of t_i and is calculated in the usual recursive way.*

The truth values of wffs built up from the atomic formulas are allocated as before, and if a wff α receives the truth value 1 relative to $\mathcal{D}$ and s, then s satisfies α in $\mathcal{D}$ and $(\mathcal{D}, s)$ is a local model of α.

Should α be satisfied by every s in $\mathcal{D}$, $\mathcal{D}$ is a global model of α.

Exercise 49 *For simplicity,assume that the languages of these exercises are built up **without** using any variables (since the intended interpretations are blueprintable).*

1. *Consider the many-sorted language for the Light-Fan-Switches System and the interpretation with $D = \{1, 2, 3, 4\}$ in which 1 and 2 stand for the light and fan while 3 and 4 stand for the light's switch and the fan's switch respectively. Describe denotations of $(P, 1)$ and $(Q, 1)$ such that the resulting interpretation is a model of a sentence (wff without variables) saying 'The light's switch is in the on position but the light is not shining'. What is the sentence?*

2. *The 3 Card System: There are three players (numbered 1, 2, and 3) and three cards (red, green, and blue). Suppose one card is dealt to each player. Such a deal is what we want to represent by a state of the system.*

   - *Describe a many-sorted alphabet for a language in which to represent knowledge about the 3 Card System. The language should have atomic formulas of the form 'Player so-and-so has a card of colour such-and-such'.*
   - *Give an interpretation representing the state in which player 1 gets the red card, player 2 the blue, and player 3 the green. Give a sentence of which this interpretation is a model.*
   - *Give an ontology for the language, i.e. say which interpretations correspond to states.*

7 Glossary

- **blueprintable system** — a nice sort of system to work with, in which it is obvious what the components are and so each component can get a name.

- **constant symbol** — the formal analogue of a name.
- **equality** — the very useful predicate symbol $(=, 2)$.
- **interpretation** — a domain D of objects plus a function den assigning denotations to constant, predicate symbols, and (if relevant) also to function symbols.
- **model** — a local model of α is a state $(\mathcal{D}, s)$ that satisfies α, and a global model of α is an interpretation $\mathcal{D}$ such that every pair $(\mathcal{D}, s)$ satisfies α.
- **predicate language** — a language in which atoms are built up from predicate symbols.
- **predicate symbol** — the formal analogue of a verb, usually represented by a pair (P, n).
- **satisfaction** — the word we use to indicate that a state (or the valuation determined by the state) makes a wff true.
- **state** — an interpretation $\mathcal{D} = (D, den)$ together with a variable assignment s in D.
- **term** — an expression in the language that denotes an object; the expression may be a constant or a variable or a more complex term built up from constants and variables by means of function symbols, e.g. something like $x + 3$.
- **term interpretation** — a particularly simple kind of interpretation, in which the domain D is the set $Cons$ of constant symbols.
- **transparent language** — a predicate language, called 'transparent' because the structure of the atoms is visible.
- **variable assignment** — a function $s : Var \longrightarrow D$ giving every variable a denotation in some domain, rather like pointing to show what 'it' is.
- **well-formed formula (wff)** — the kind of thing a language has when there are variables.