COSC462 Lectures 14 and 15: First-order languages

Willem Labuschagne
University of Otago

Abstract

Transparent propositional languages containing variables are enriched by the addition of the quantifiers 'for all' and 'for some'. Local, global, and universal versions of the notion of model are distinguished. Reification is introduced in the context of the situation calculus.

1 Quantifiers

Let us contrast blueprintable and unblueprintable systems, from the point of view of knowledge representation.

In the case of a **blueprintable system**, it is in principle possible to use a transparent propositional language in which every component of the system has a name. The prototypical example is the Light-Fan System, for which we might use a language based on an alphabet which includes the set $Cons = \{L, F\}$ and the set $Pred = \{(P, 1)\}$ with the intention that P should stand for the predicate 'is on'. Semantically, states of the system can be represented by term interpretations. Syntactically, states such as 10 can be represented by state descriptions such as $P(L) \wedge \neg P(F)$. Furthermore it is possible to express the ideas of 'all' and 'some' by conjunctions and disjunctions respectively. For example, to express that all the components of the Light-Fan System are on, the agent may write $P(L) \wedge P(F)$. Similarly, to express the idea that some component is on, the agent may write $P(L) \vee P(F)$.

In the case of an **unblueprintable** system, it is not possible (or for some reason is unattractive) to include a name for every component, so that we cannot restrict consideration to term interpretations and cannot represent states syntactically by state descriptions. Moreover, it is no longer possible to express 'all' by conjunction or 'some' by disjunction. In

the absence of names for all components, we introduce variables. In order to enable the language to express 'all' and 'some' we shall now enrich the language further by adding *quantifiers*. This is done as follows. For every variable x, we add the *universal* quantifier $\forall x$, which may be read 'for all x' or 'for every x', and we add the *existential* quantifier $\exists x$, read as 'for some x' or 'there exists an x such that' or 'for at least one x'. (In the case of a sorted language, $\forall x^\sigma$ is read 'For every x of sort σ' and $\exists x^\sigma$ is read 'For at least one x of sort σ'.) Languages with quantifiers are called *first-order* languages, whereas languages without quantifiers are propositional languages.

Definition 1 *Suppose that A is the set of atoms induced by an alphabet comprising sets $Sort$, $Cons$, Fun, (each of which may possibly be empty), Var and $Pred$.*

*The **first-order** language L_A is the set of all well-formed formulae (wffs) α where a string α is a wff iff one of the following is the case:*

- $\alpha \in A$
- $\alpha = \neg\beta$ *for some previously constructed wff* β
- $\alpha = (\beta * \gamma)$ *for some previously constructed wffs* β *and* γ *and for some* $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
- $\alpha = \forall x(\beta)$ *or* $\alpha = \exists x(\beta)$ *for some* $x \in Var$ *and some previously constructed wff* β.

What is the effect of adding quantifiers to a language? Quantifiers bind variables. We shall presently define the difference between bound and free variables, but for the moment we may view binding from the perspective of vagueness. Intuitively, the change from a sentence like $P(L)$ (*i.e.* the light is on) to a wff like $P(x)$ (*i.e.* x is on) introduces a degree of vagueness. This vagueness can be reduced in two ways. Semantically, we may use a variable assignment s in some domain D to say what x denotes. Syntactically, we may change the wff $P(x)$ by applying a quantifier to produce a new and less vague wff, say, $\exists x P(x)$ (*i.e.* some component is on).

We are using the word 'vagueness' here in the sense of 'having many possible meanings'. A wff like $P(x_1)$ is vague because what it means, and hence its truth or falsity, is sensitive to the choice of variable assignment — typically, it would be satisfied in an interpretation by some variable assignments but not by others. But a quantified wff such as $\exists x_1 P(x_1)$ is not vague, does not depend for its meaning on any variable assignment, and turns out to be satisfied either by all assignments or by none. For

example, if we have an interpretation (D, den) in which $den(P, 1) \neq \varnothing$, the wff $\exists x_1 P(x_1)$ will be satisfied by all variable assignments in D. If we have an interpretation (D, den) for which $den(P, 1) = \varnothing$, the wff $\exists x_1 P(x_1)$ will be satisfied by none. We might therefore say that a quantified wff like $\exists x_1 P(x_1)$ has just one meaning in an interpretation.

Here is another way to think about the effect of quantifiers. In a transparent propositional language having suitable function and predicate symbols we can formulate equations like $x^2+2x+1 = 0$ and by considering variable assignments we can find values of x that solve the equation. Such an equation is like a question that asks whether an interpretation possesses members capable of satisfying the relevant constraints. But there is a second possible use for equations. Once quantifiers are available, we can formulate identities like $\forall x(x^2 + 2x + 1 = (x + 1)^2)$, which represent laws holding across an entire interpretation. Whereas an unquantified equation is like a question, an identity is a powerful assertion.

Exercise 2 *1. Consider the first-order language with* $Cons = \{a\}$ *and* $Pred = \{(P, 1), (Q, 1)\}$.

- *What are the atoms of this language?*
- *How would you express the idea that everything has both the property P and the property Q? Is this intuitively the same as saying that everything has property P and everything has property Q?*
- *How would you express the idea that everything has property P or property Q? Is this intuitively the same as saying that everything has property P or everything has property Q?*
- *Think of the predicate symbol P as standing for "is a man", Q for "is mortal", and the constant a as being an abbreviation for "Socrates". Write down a wff expressing the idea that if all men are mortal and Socrates is a man then Socrates is mortal.*

2. Consider the first-order language with $Cons = \{a, b\}$, $Fun = \{(f, 1)\}$, *and* $Pred = \{(P, 2)\}$*:*

- *What are the atoms of this language?*
- *Think of the function symbol f as an abbreviation for "the mother of" and think of the predicate symbol P as standing for the relationship "is an ancestor of". How would you express the idea that there is someone who is an ancestor of b's mother?*

3. *An alien spaceship has crashed in the desert and you have been called in as consultant to a nearby military base, where the wreckage has been locked away from public scrutiny in order to prevent mass hysteria and the Moral Decay of the Youth. (Presumably you are sufficiently decrepit not to be so sensitive to moral decay, or perhaps you are just expendable.) Before your eyes there is a complicated mechanism. It has one readily identifiable component — a light — but for the rest it is not obvious what the components are. You are interested, for the time being, in two things. Since the light shines intermittently, it would seem that some of the components are 'on', utilising energy. And so you are interested in whether components are on or off. Also, bits of the wreckage look badly twisted, so that it seems likely that some of the components may be broken. So you are interested in whether components are defective or not.*

    - *Describe suitable sets Cons and Pred and write down a sentence expressing the idea that all components which are on, are functioning correctly.*

4. *Here is a description of a biological system taken from a problem called Schubert's Steamroller, which is named after Len Schubert and appeared in Pelletier FJ (1986): 75 Problems for testing automatic theorem provers, Journal of Automated Reasoning 2:191-216 (and see also errata in Pelletier (1988): Journal of Automated Reasoning 4:235-236). It is actually a benchmark problem for automated reasoning algorithms, but offers a good exercise in knowledge representation for humans.*

    Here is the puzzle:

    Wolves, foxes, birds, caterpillars, and snails are animals, and there exist some of each. Also there exist some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are in turn much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

    If we were doing automated reasoning algorithms, the challenge would be to get your algorithm to prove that ***there exists an animal that likes to eat a grain-eating animal****. However, let's stick to knowledge representation.*

Write down wffs that express the information provided, and a final wff expressing the existence of an animal that likes to eat a grain-eating animal. (Some of the English sentences are ambiguous, and you will have to decide for yourself how to resolve the ambiguity.)

2 Satisfaction

We now want to discuss how truth values are assigned to wffs.

Given a first-order language L_A, the wffs without quantifiers are familiar things — they form the transparent propositional language (with variables) generated by A. Only the wffs in which quantifiers occur are new. Accordingly, an interpretation of the first-order language L_A is precisely the same kind of pair $\mathcal{D} = (D, den)$ as before, but we need to think carefully about the way truth values ought to be allocated to wffs with quantifiers. Since variables are involved, we would expect variable assignments to play a crucial role in this allocation. As before, variable assignments are functions from Var to the domain of interpretation D.

The intuition that guides the definition of satisfaction may be illustrated as follows. We want to say (roughly) that $\exists x P(x)$ is true in $\mathcal{D}$ if there is at least one value of x such that $P(x)$ is true, in other words if there is at least one variable assignment in D satisfying $P(x)$. And we want to say (roughly) that $\forall x P(x)$ is true in $\mathcal{D}$ if all values of x make $P(x)$ true, i.e. if every variable assignment in D satisfies $P(x)$. What makes things more complicated than this is that there may be other variables besides x.

Consider the system of integers $\{\ldots, -1, 0, 1, \ldots\}$ and the wff $\exists x(y + x = 0)$. The wff says 'There is at least one value of x which is a solution to the equation $y + x = 0$'. Bearing in mind that x is not the only variable lurking in the equation, when should $\exists x(y + x = 0)$ be satisfied by a variable assignment s relative to the interpretation $\mathcal{D}$? Well, if some value for x can be found, not necessarily that which s itself assigns to x, so that an assignment giving this value to x and in all other respects behaving like s will satisfy $y + x = 0$. For example, if we have the set of integers $D = \{\ldots, -1, 0, 1, \ldots\}$ and s is the assignment which gives to every variable the value 113, then s satisfies $\exists x(y + x = 0)$ because the assignment s' which gives x the value -113, and gives every other variable (including y) the value 113, satisfies the shorter wff $y + x = 0$.

Similarly, for the wff $\forall x(x + y = y + x)$ to be true, every value of x should make the wff true, because the universal wff $\forall x(x+y = y+x)$ says 'Every value of x satisfies the equation $x + y = y + x$'. Thus $\forall x(x + y = y + x)$ should be satisfied by an assignment s in $\mathcal{D}$ if every possible value of x allows $x + y = y + x$ to be satisfied, provided additionally that

the assignment of values to other variables is strictly according to the original assignment s.

Let us formalise this intuition.

Definition 3 *An interpretation $\mathcal{D} = (D, den)$ and variable assignment $s : Var \longrightarrow D$ together determine a valuation $v : A \longrightarrow \{0, 1\}$ which assigns to an atom $P(t_1, t_2, \ldots, t_n)$ 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.*

*Assignment s is said to **satisfy** the atom $P(t_1, t_2, \ldots, t_n)$ in $\mathcal{D}$ iff the valuation v assigns to $P(t_1, t_2, \ldots, t_n)$ the truth value 1.*

Nonatomic wffs of L_A are satisfied by assignment s in the interpretation $\mathcal{D}$ according to the following:

- *s satisfies $\alpha = (\neg\beta)$ iff s does not satisfy β*
- *s satisfies $\alpha = (\beta \wedge \gamma)$ iff s satisfies both β and γ*
- *s satisfies $\alpha = (\beta \vee \gamma)$ iff s satisfies at least one of β and γ*
- *s satisfies $\alpha = (\beta \rightarrow \gamma)$ iff s satisfies γ or fails to satisfy β*
- *s satisfies $\alpha = (\beta \leftrightarrow \gamma)$ iff s satisfies both β and γ or else satisfies neither*
- *s satisfies $\alpha = \exists x(\beta)$ iff β is satisfied by some assignment s' such that if $y \neq x$, then $s(y) = s'(y)$*
- *s satisfies $\alpha = \forall x(\beta)$ iff β is satisfied by every assignment s' such that if $y \neq x$, then $s(y) = s'(y)$*

*An **ontology** for a first-order language L_A is a pair (S, V) where S is some set of **states** $(\mathcal{D}, s)$ and V is the obvious labelling function mapping states to the valuations in W_A determined by them.*

*The state $(\mathcal{D}, s)$ is a **local model** of α, written $(\mathcal{D}, s) \in \mathcal{M}(\alpha)$, iff α is satisfied by s in $\mathcal{D}$.*

*$\mathcal{D}$ is a **global model** of α iff α is satisfied by every assignment s in $\mathcal{D}$.*

As you might suspect, we could now define two different kinds of classical entailment, one based on local models and the other on global models. We shall keep things simple and use local models only, since they correspond to states and thus to valuations.

Definition 4 *$\alpha \models \beta$ iff $\mathcal{M}(\alpha) \subseteq \mathcal{M}(\beta)$ and $\alpha \equiv \beta$ iff $\mathcal{M}(\alpha) = \mathcal{M}(\beta)$.*

Exercise 5 *1. Recall the first-order language with* $Cons = \{a\}$ *and* $Pred = \{(P, 1), (Q, 1)\}$*:*

- *Describe four interpretations* $\mathcal{D}_1$*,* $\mathcal{D}_2$*,* $\mathcal{D}_3$*, and* $\mathcal{D}_4$ *such that the domain* D_1 *is a set with one element,* D_2 *has two elements,* D_3 *has one thousand elements, and* D_4 *has infinitely many elements.*
- *For each of the four interpretations, give an example of a variable assignment.*
- *For each of the four variable assignments, work out whether it satisfies the following wffs in the relevant interpretation:*

  (a) $Q(x_1)$
  (b) $\exists x_1 P(a)$
  (c) $\exists x_1 P(x_1)$
  (d) $\exists x_1 P(x_2)$
  (e) $\forall x_1 P(x_1)$
  (f) $Q(a) \vee \neg \forall x_3 Q(x_3)$

2. Recall the first-order language with $Cons = \{a, b\}$*,* $Fun = \{(f, 1)\}$*, and* $Pred = \{(P, 2)\}$*:*

- *Describe four interpretations* $\mathcal{E}_1$*,* $\mathcal{E}_2$*,* $\mathcal{E}_3$*, and* $\mathcal{E}_4$ *such that the domain* E_1 *is a set with one element,* E_2 *has two elements,* E_3 *has one thousand elements, and* E_4 *has infinitely many elements.*
- *For each of the four interpretations, give an example of a variable assignment.*
- *For each of the four variable assignments, work out whether it satisfies the following wffs in the corresponding interpretation:*

  (a) $P(x_1, f(x_1))$
  (b) $\exists x_1 P(a, x_1)$
  (c) $\forall x_1 P(a, x_1)$
  (d) $\exists x_1 P(b, f(x_1))$
  (e) $\exists x_1 P(f(x_1), f(f(b)))$

3. You are a versatile individual, not limited to the analysis of flying saucers. As a political consultant, you are keeping a keen eye on the forthcoming election of the President of Upper Slimeria. There are two official candidates, the rotund lawyer Sacher Tort

and the reformed serial murderer and smoker Puffy Hackenslice. You are interested in who will vote for whom. Unfortunately Slimeria last had a census a century ago, before the civil war, so it is unclear how many voters there are (not to mention what their names might be). Sadly, Slimerian schools have concentrated their curriculum on practical subjects such as the maintenance of automatic weapons, so the electorate are rather fuzzy about electoral procedure, and as likely to vote for the local gunsmith or butcher as for an official candidate.

- *Describe suitable sets Cons and Pred and write down a wff expressing the hope that no voter will vote for themselves and that every voter will vote for exactly one of the two official candidates. Give an example of a global model of this wff. Also give an example of an interpretation that is not a global model of this wff.*

4. *Having made your fortune as a consultant on aliens and politicians (not as incongruous a combination as it might seem, because both harbour the same ambition, namely to take over the world, and then the galaxy), you decide to retire to Auckland and apply the insights into deviant personalities you have so painfully gained over the years to the training of used-car salespersons. Being a flexible sort of individual, you don't mind that fluctuations in the job market result in trainees dropping out to become real estate agents instead or in redundant real estate agents joining your classes, so that you are unable to keep a fixed roster of students. Your course emphasises three golden rules. Always treat the customer with the sort of familiarity that the rest of society reserves for intimate friendships. Never try to sell a customer two different cars simultaneously. And never become involved in acrimonious disputes with customers, although disputes with your fellow salespersons are of course to be expected.*

   - *Describe a many-sorted language that allows you to distinguish between cars, salespersons, and customers. Equip the language with a set Pred that includes equality. Write down a wff expressing each of the three golden rules. Give an example of a global model of the three wffs. Also give an example of an interpretation that is not a global model of the set of three wffs.*

5. *Your training of salespersons doesn't occupy much of your time, and you decide to indulge a dormant interest in mathematics by*

exploring the system of natural numbers (non-negative integers). Having failed to master the usual numerals to the base ten, you are relieved to learn that every natural number can be described as being either zero or the successor of a smaller natural number. You find yourself fascinated by the relationship 'is smaller than'.

- *Describe a set $Cons$ and a set Fun that would allow you to restrict attention to term interpretations (regrettably not finite ones) and a set $Pred$ that would allow you to write down wffs expressing your knowledge of the following properties of the system:*
  - *No natural number is smaller than itself.*
  - *Every natural number is smaller than its successor.*
  - *If one natural number is smaller than another, the second number is not smaller than the first.*
  - *If one natural number is smaller than a second, which in turn is smaller than a third, then the first natural number is smaller than the third.*
  - *For every pair of distinct natural numbers, one is smaller than the other.*
  - *Zero is not the successor of any natural number.*
  - *Different natural numbers have different successors.*
- *Give an example of a term interpretation that is a global model of the set of wffs. Give an example of a global model which is not a term interpretation. Write down a wff which is true in your first global model (the term interpretation) but not in the second.*

6. *Assume a first-order language L_A having a constant a and a predicate symbol $(P, 1)$. Show that*
   - $P(a) \models \exists x_1 P(x_1)$
   - $\forall x_1 P(x_1) \models P(a)$
   - $\forall x_1 P(x_1) \equiv \neg\exists x_1 \neg P(x_1)$

7. *Assume a first-order language L_A having no constants and a binary predicate symbol $(Q, 2)$.*

   Show that $\forall x_1 \exists x_2 Q(x_1, x_2) \not\models \exists x_2 \forall x_1 Q(x_1, x_2)$.

   (Hint: Construct an interpretation in which some assignment satisfies the former wff but not the latter. Keep it simple.)

3 Sentences

Previously, in languages without quantifiers, sentences were wffs in which no variables occurred. The introduction of quantifiers adds a new kind of sentence in which variables do occur but are bound.

Definition 6 *An occurrence of a variable x in α is free iff one of the following cases applies:*

- *α is $P(t_1, \ldots, t_n)$ and x occurs in a term t_i*
- *$\alpha = \neg\beta$ and x occurs free in β*
- *$\alpha = (\beta * \gamma)$ where $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ and x occurs free in β or γ*
- *$\alpha = \exists y(\beta)$ and x occurs free in β but $x \neq y$*
- *$\alpha = \forall y(\beta)$ and x occurs free in β but $x \neq y$*

*An occurrence of a variable which is not free is bound. If no variable occurs free in the wff α, then α is a **first-order sentence**.*

Thus a first-order sentence is a wff all of whose variable occurrences (if any) are bound. For example, $\exists x_1(\exists x_2(P(x_1) \wedge \neg P(x_2))$ is a sentence. To see that $\exists x_1(\exists x_2(P(x_1) \wedge \neg P(x_2))$ is a sentence, note firstly that x_1 occurs free in $P(x_1)$ and x_2 occurs free in $P(x_2)$. Thus x_1 and x_2 occur free in $P(x_1) \wedge \neg P(x_2)$. But only x_1 occurs free in $\exists x_2(P(x_1) \wedge \neg P(x_2))$, the occurrence of x_2 in $P(x_2)$ now being bound by the quantifier $\exists x_2$. And no variable occurs free in $\exists x_1(\exists x_2(P(x_1) \wedge \neg P(x_2))$.

Every quantifier has a *scope*, namely the previously constructed wff to which the quantifier applies. A wff like $\exists x_2(P(x_1) \wedge \neg P(x_2))$ is of the form $\exists x(\beta \wedge \gamma)$, with the scope of the quantifier extending over $\beta \wedge \gamma$. In contrast a wff like $\exists x_2(P(x_1)) \wedge \neg P(x_2)$ has the form $\exists x(\beta) \wedge \gamma$, and so the scope of $\exists x_2$ is now limited to $\beta = P(x_1)$, which means that the occurrence of x_2 in $P(x_2)$ is free.

Recall that in a propositional language, two valuations that agree on the atoms in a sentence α will both satisfy α or neither satisfy α (see Lemma 3 of Lecture 3). First-order languages have an analogue of this useful fact.

Theorem 7 *Let $\mathcal{D}$ be an interpretation and s, s' assignments which agree at all variables (if any) occurring free in the wff α. Then s satisfies α in $\mathcal{D}$ iff s' satisfies α in $\mathcal{D}$.*

Proof. *The proof is by induction on the number of steps in the construction of wff α.*

Suppose α is an atom $P(t_1, \ldots, t_n)$. Let s and s' be two assignments that agree on all the variables occurring free in α. Then $s(x) = s'(x)$ for every variable x that occurs in $P(t_1, \ldots, t_n)$, because every variable in an atom occurs free. The denotations $d_1, \ldots, d_n$ in D of the terms $t_1, \ldots, t_n$ are thus the same, whether calculated using s or s'. And so $(d_1, \ldots, d_n) \in den(P, n)$ or not, independently of whether it is s or s' that is used. It follows that s and s' both satisfy the atom $P(t_1, \ldots, t_n)$ or both fail to satisfy $P(t_1, \ldots, t_n)$. In other words, s satisfies α iff s' does.

Now suppose that the theorem holds for wffs constructed in k or fewer steps, and let α be any wff constructed in $k+1$ steps. Then α is formed by using, in the last construction step, one of the connectives or quantifiers. We take each case in turn.

Let s and s' be assignments such that $s(x) = s'(x)$ for every variable that occurs free in $\alpha = \neg\beta$. By the induction hypothesis, if two assignments agree at all the variables occurring free in β then β is either satisfied by both assignments or by neither. Since s and s' agree at all variables occurring free in β, either both s and s' satisfy β or neither do. In the former case, neither s nor s' satisfy $\neg\beta$, while in the latter both satisfy α.

Let s and s' agree on the variables occurring free in $\alpha = \beta \wedge \gamma$. By the induction hypothesis, for each of β and γ separately it is the case that if two assignments agree on the variables occurring free in the wff, the assignments either both satisfy the wff or neither satisfy the wff. Since s and s' agree at all variables occurring free in β and in γ, either both s and s' satisfy β or neither do, and the same holds for γ. Thus s and s' both satisfy $\beta \wedge \gamma$ or neither do.

The cases of $\beta \vee \gamma$, $\beta \rightarrow \gamma$, $\beta \leftrightarrow \gamma$ are similar to that of $\beta \wedge \gamma$.

Let s and s' agree on the variables occurring free in $\alpha = \forall x(\beta)$. By the induction hypothesis, if two assignments agree on the variables occurring free in β, then β is satisfied by both or by neither. But do s and s' agree on the variables free in β? Perhaps not, because x may occur free in β. After all, s and s' only have to agree on the variables occurring free in $\forall x(\beta)$, and these are all those free in β except for x, which may or may not occur free in β but certainly does not occur free in $\forall x(\beta)$. For each $d \in D$, let us indicate by $s(x \mapsto d)$ the assignment which assigns to every variable $y \neq x$ the same value as s does, but assigns to x the value d (which may be different from the value assigned to x by s). Similarly $s'(x \mapsto d)$ is the assignment identical to s' except possibly where x is concerned and which gives to x the value d. By the induction hypothesis, since $s(x \mapsto d)$ and $s'(x \mapsto d)$ agree on all the variables that occur free in β, either both satisfy β or neither do. Now recall that if

s satisfies $\forall x(\beta)$ *then* $s(x \mapsto d)$ *has to satisfy* β *for every choice of* d. *So* $s'(x \mapsto d)$ *satisfies* β *for every choice of* d. *But this means that* s' *satisfies* $\forall x(\beta)$. *The converse direction is similar. Thus* s *satisfies* $\forall x(\beta)$ *iff* s' *does.*

The case of $\exists x(\beta)$ *is treated along the same lines, but is simpler because you only have to worry about a single choice of* d. ■

Corollary 8 *Suppose* α *is a sentence and* $\mathcal{D}$ *an interpretation. Then* α *is either satisfied by all assignments* s *in* $\mathcal{D}$ *or by none. Hence* $\mathcal{D}$ *is a model either of* α *or of* $\neg\alpha$.

Remark 9 *For a sentence* α, *the distinction between local and global models disappears — if* $(\mathcal{D}, s)$ *is a local model of* α, *then* $\mathcal{D}$ *is a global model of* α. *Thus, for sentences, we may simply speak of* ***models***.

Exercise 10 *1. Suppose you have a first-order language* L_A *having no constants and just one predicate symbol, namely* $(=, 2)$. *You may use infix notation to write atoms as* $x = y$ *rather than* $=(x, y)$. *Equality and the quantifiers can be used to modulate the* ***size of models***. *Write down*

- *a sentence* α *such that every model of* α *has at least one element in its domain*
- *a sentence* β *such that every model of* β *has exactly two elements in its domain*
- *a sentence* γ *such that every model of* γ *has at most three elements in its domain.*

4 Situations

Consider the Light-Fan-Switches System, in which there is a switch for the light and another for the fan. It is possible for an agent to change the state of the system by depressing one of the switches. In order to represent *dynamic* knowledge (knowledge of how the system changes as a result of actions), the agent may wish to use a special kind of many-sorted first-order language called a *situation calculus*.[1]

A situation calculus has several sorts.

[1] The original situation calculi were invented by John McCarthy and did not resemble very closely the languages discussed so far. McCarthy was the inventor of LISP, the first functional programming language, and this seems to have influenced his notation. The rest of the logic-based AI community, led by Vladimir Lifshitz, soon recast the situation calculi in a more orthodox way.

The name tells us we need a sort that we will call *Situation*. The idea is that a situation term denotes a state of the system, in other words stands for a 'snapshot' of the system at some moment in time. This is quite a big jump from using terms merely to denote components like the fan or its switch. By way of a concrete example, recall that the Light-Fan-Switches System has four components and therefore (assuming that we are interested only in whether components are on or off) has 16 possible states: the state in which all four components are on, 4 states in which exactly three components are on, 6 states in which exactly two components are on, 4 states in which exactly one component is on, and the state in which zero components are on. (We may wish to rule out some of these states because the system blueprint reveals it to be impossible for, say, the light to be on when its switch is off, and so on, but for the moment we are concerned with the logically possible states rather than the application of fixed information to exclude states.) And so we could put into the agent's knowledge representation language 16 constants of sort *Situation*, each representing a different state of the system. Let us suppose these constants to be $S_1, \ldots, S_{16}$.

A situation calculus also has a sort that we will call *Action*. Terms of this sort are intended to denote the actions an agent may perform to change the state of the system. In the case of the Light-Fan-Switches System, we may suppose that the agent can do two things: press the light's switch and press the fan's switch. We ignore as irrelevant a multitude of other possible actions such as taking a hammer to the system in a paroxysm of rage, painting the components in pastel colours, or reconnecting the wires in novel ways. Let us put into the language two constants of sort *Action*, each representing one of the actions we regard as relevant: A_1 and A_2.

A situation calculus has a special function symbol that we will call *Result*. Since an agent can change the state of the system by performing an action, it is essential that the language be able to express descriptions such as 'the state that results from performing action A_1 in state S_5'. By putting a binary function symbol *Result* of sort (*Action*, *Situation*, *Situation*) into the language, we make it possible to use a term like $Result(A_1, S_5)$ to denote the situation that results from applying the action called A_1 when the system is in the state called S_5.

A situation calculus has a sort that we will call *Fact*. The idea is that states of a system can be distinguished from one another by their properties. Consider for example the state in which the the light is on, the light's switch is on, the fan's switch is on, but the fan is off. These basic facts distinguish this particular state from all 15 of the other possible states. By cleverly including terms of sort *Fact*, we make it

possible to differentiate between situations by referring to the different facts that obtain in each case. In the case of the Light-Fan-Switches System, we would need a term representing the fact that the light is on, a term saying that the light's switch is on, a term saying that the fan is on, and a term saying that the fan's switch is on. Let us put into the language four constants of sort *Fact*: F_1, F_2, F_3, F_4.

Let us pause to remark that we have again taken a big step. Previously, we would have expressed the fact that the light is on by a wff of our language. Now we are proposing to express the same fact by a term, not by a wff. To appreciate the novelty, recall that when a language is interpreted, the terms denote objects in the domain of interpretation whereas wffs do not. Something very abstract, namely a proposition expressing that the light is on, will now be included in every interpretation as a member of the domain, whereas the domain has hitherto consisted of elements representing such concrete things as the light itself. This trick of including, amongst the terms of the language and amongst the members of the domain, representations not merely of concrete components but also of abstract properties, is called *reification*. The word is derived from the Latin 'res', meaning 'thing', and so reification is the process by which we *thingify* some abstract property.

A situation calculus has a special predicate symbol that we will call $Holds$. The idea is that if we wish to use terms of sort *Fact* to distinguish between situations, then the language needs to have the capacity to express that a fact does or does not hold in a situation. The predicate symbol $Holds$ is binary and has sort (*Fact*, *Situation*). To express, for instance, that fact F_2 holds in the situation that results when the agent performs action A_1 in situation S_5, the atom $Holds(F_2, Result(A_1, S_5))$ suffices.

We now have the basis for a language in which an agent can represent the knowledge that may be needed for planning how to manipulate the Light-Fan-Switches System in order to achieve goals (such as the goal of having the light on but the fan off). By way of example, let us consider how to represent knowledge of the way in which the fan's switch affects the fan's functioning. We shall assume that switches are devices that toggle, i.e. that reverse their effect every time they are pressed. We need a wff saying that pressing the fan's switch in a situation in which the fan is off will result in the fan's being on, and pressing the fan's switch in a situation in which the fan is on will result in the fan's being off. The wffs almost write themselves, once we have decided that our intended interpretation is one in which A_2 denotes the action of pressing the fan's switch and F_2 the fact that the fan is on. We use x^S as a variable of sort *Situation*.

- $\forall x^S(\neg Holds(F_2, x^S) \rightarrow Holds(F_2,\ Result(A_2, x^S))$
- $\forall x^S(Holds(F_2, x^S) \rightarrow \neg Holds(F_2,\ Result(A_2, x^S))$

Sentences like these express some of the facts that hold after an action has been taken, but do not identify the resulting situation unambiguously. Even if we know in what situation or state a given action has been taken, the sentences above are incomplete guides to the resulting situation because they do not express the common-sense idea that nothing else changes. As humans we tend to take for granted that pressing the switch on the light will affect the light but have no effect on the fan. It is interesting to consider how we would inject this common-sense idea into a robot. The obvious solution is to use additional sentences which spell out in painful detail everything that does not change when an action is performed. Such additional sentences are called frame axioms. The problem with frame axioms is that if there are n actions and m facts one will on average need nm frame axioms. This is not only clumsy but can lead to inefficiencies for agents who have to use some sort of reasoning algorithm to do their planning. So one would be very interested in an alternative to frame axioms, and the search for such an alternative in the AI community during the late 1970s was one of the factors leading to the invention of non-monotonic logic.

In order to become familiar with the situation calculus, you should wrestle with the following extended exercise, which introduces a standard illustrative AI system, the Blocks World. When doing this exercise, use a situation calculus as described above, don't regress to what you may have seen in a 300-level AI textbook.

Exercise 11 *Consider a system consisting of three wooden blocks on a table. One of the blocks is painted red, another green, and the third blue. The interesting thing about the blocks is whether one is on top of another. More precisely, we are interested in whether a block is on another block or is on the table, and in whether a block has another on top of it or is clear. Thus one state of the Blocks World system is that in which each of the three blocks is on the table and is clear (has no block on top of it). Another state is that in which the red block is on the table, the green block is on the red, and the blue block is on the green, forming a little tower.*

An agent can change the state of the Blocks World system in three ways. A block may be ***moved*** *from the top of one block to the top of another. A block may be picked up from the table and* ***stacked*** *on top of another block. And a block may be* ***unstacked*** *by taking it off the top of another block and placing it on the table.*

1. *Draw diagrams to represent each of the thirteen possible states of the Blocks World. (If it is impractical to use different colours for the blocks, label them R, G, or B.)*

2. *Describe a situation calculus suitable for representing knowledge of the Blocks World.*

   (Hint: You know what sorts the language must have in order to be a situation calculus, and you may add sorts such as Block to those. Now decide, for each sort, what you need besides variables. For example, consider the sort Action. You want an action that takes any block x from the top of block y to the top of block z. The action, i.e. what happens, depends on x, y, and z. So don't try to represent this action by a constant. Instead use a function symbol that takes three terms of sort Block as input and delivers a term of sort Action.)

3. *Give sentences to express the following:*

   - *If one block is on another, then it is not the case that the second block is clear.*
   - *Preconditions for successful execution of actions:*
     - *If a block is on the table and is clear, and if a second block is clear, then the result of stacking the first block on the second is a situation in which the first block is indeed on the second.*
     - *If a block is on another block but is itself clear, then the result of unstacking it is a situation in which it (the first block) is on the table.*
     - *If a block is on another block but is itself clear, and if the third block is clear, then the result of moving the first block from the second to the third is a situation in which the first block is indeed on the third block.*
   - *Failure:*
     - *If the precondition for successful execution of the the action of stacking one block onto another is not satisfied, then the action of stacking does not change the situation in any relevant way.*
     - *If the precondition for successful unstacking is not satisfied, then the attempt to unstack does not change the situation in any relevant way.*

– *If the precondition for successful execution of the action of moving a block from the top of one to the top of another is not satisfied, then the action of moving does not change the situation in any relevant way.*

- *Frame axioms: How might one express the knowledge that an action affects only the 'relevant part' of the system? For example, unstacking the red block from its position on top of the green block should not cause the green block to mysteriously appear stacked on top of the red. Give some examples of frame axioms, but don't try to give them all. Estimate how many there would be.*

4. *Give an interpretation of the language that is a model of the sentences above.*

5 Modal foundations for first-order logic

In this brief section the aim is not to teach you new skills but to give you new insight into old skills, and generally to give you a unifying perspective on logic languages.

We have encountered three basic kinds of logic language — propositional languages, first-order languages, and modal languages. Ordinary propositional logic is contained within first-order logic, because all you have to do is remove the quantifiers and variables, and the semantics of transparent propositional logic involves just interpretations without variable assignments. Similarly, propositional languages can be seen as sublanguages L_A of modal languages $L_A^{[n]}$ in which we simply ignore the sentences containing modal operators. The usual semantics of propositional logic may also be seen to be a special case of the possible worlds semantics. Kripke models are triples of the form $(S, \{R_i | i \leq n\}, V)$ which consist of some set S of possible worlds, some bunch of binary accessibility relations on S, and a labelling function V connecting members of S with valuations. To get the semantics of a propositional language we simply ignore the accessibility relations (just as we ignore the modal operators) to get an ontology (S, V).

Now we shall show that first-order logic may also be regarded as a special case of modal logic. (Thus all our logic languages are really modal languages, possibly simplified by ignoring some bits and pieces.) Syntactically, we are going to regard quantifiers as modal operators. Semantically, we know what we mean by the ontology (S, V) of a first-order language, and we now want to say how this can be viewed as a Kripke model $(S, \{R_i | i \leq n\}, V)$.

Well, it turns out that we need infinitely many accessibility relations, but then everything is easy. Take any first-order language and pick any interpretation $\mathcal{D} = (D, den)$ of the language. Consider the structure $(S, \{R_x | x \in Var\}, V)$ where

- S is the set of all states $(\mathcal{D}, s)$ obtainable from variable assignments $s : Var \longrightarrow D$
- for every variable x, the corresponding binary relation R_x consists of all pairs (s, s') such that s and s' agree on all variables except possibly x
- V is a function that assigns to every state $(\mathcal{D}, s)$ the valuation determined by it in the obvious way.

This structure is a Kripke model which allows us to regard the quantifiers as modal operators and is equivalent to the original first-order interpretation $\mathcal{D}$ in the sense that a variable assignment s will satisfy a wff α in the interpretation $\mathcal{D}$ iff α is satisfied at the possible world s in the Kripke model $(S, \{R_x | x \in Var\}, V)$.

Consider the quantifiers $\forall x$ and $\exists x$. We know that a wff of the form $\forall x\beta$ is satisfied by a variable assignment s in $\mathcal{D}$ iff β is satisfied by every assignment s' such that $(s, s') \in R_x$ and similarly $\exists x\beta$ is satisfied by s iff β is satisfied by at least one s' such that $(s, s') \in R_x$. Thus $\forall x$ is just the modal operator $[x]$ and $\exists x$ is $\langle x \rangle$ for each variable x.

What are the insights to which we are led by this view of first-order logic?

In the first place, our experience with modal operators suggests to us that certain sentences may be 'universally' true, i.e. have every interpretation as model. For example, sentences of the form $\forall x(\beta \rightarrow \gamma) \rightarrow (\forall x\beta \rightarrow \forall x\gamma)$ should be universally true because, in modal logic, the schema $K = \Box(\beta \rightarrow \gamma) \rightarrow (\Box\beta \rightarrow \Box\gamma)$ is satisfied at every world in every Kripke model. This is easily rephrased for a multimodal language as the schema $[i](\beta \rightarrow \gamma) \rightarrow ([i]\beta \rightarrow [i]\gamma)$. But if every first-order interpretation may be regarded as a Kripke model, then the instances of this schema must perforce be satisfied by every local model $(\mathcal{D}, s)$, for every first-order interpretation $\mathcal{D}$. And the instances are precisely the wffs of form $\forall x(\beta \rightarrow \gamma) \rightarrow (\forall x\beta \rightarrow \forall x\gamma)$. As a similar example, it is a fact of modal logic that $\Box\beta \leftrightarrow \neg\Diamond\neg\beta$ is satisfied at every world of every Kripke model, which suggests we should be able to show that $\forall x\beta \leftrightarrow \neg\exists x\neg\beta$ is universally true. Thus we are led to a better understanding of what to expect from first-order logic by our experience with modal logic.

There is a second and more subtle insight, one that eluded logicians for almost the whole of the 20th century. First-order interpretations $\mathcal{D} =$

(D, den) can be regarded as Kripke models $(S, \{R_x | x \in Var\}, V)$. But we can get more Kripke models by varying any of the three parameters S, $\{R_x | x \in Var\}$, or V. In particular, we can vary S. In contrast, first-order logic has traditionally assumed that S **must** be the set of all states $(\mathcal{D}, s)$ where we use all possible $s : Var \longrightarrow D$. This is too restrictive.

Here is an example of the sort of application that invites us to use only some variable assignments instead of all. If the states in S include all the functions $s : Var \longrightarrow D$, then one can think of the change from a variable assignment s to an accessible assignment s', where $(s, s') \in R_x$, as a kind of *independent* update of the value of the variable x. Independent, because the values of all the other variables remain as they were recorded by s. But suppose one wishes to represent knowledge that in natural language might be expressed by 'He and she designed the house they built on the hill.' Bearing in mind that 'he', 'she', and 'they' are variables, say represented by x, y, and z in a formal language, let us consider the effect of update. If we have an assignment s recording that x has as its value Tom, y as its value Sally, and z as its value the pair (Tom,Sally), and if we wish to move to a new assignment recording a new value for x, say Dick, then the value of z must be updated at the same time. In other words, update is no longer something that affects variables independently. To model this interdependence of variables, we would allow S to consist of states that involve only some of the functions $s : Var \longrightarrow D$ rather than all of them. From the point of view of possible worlds semantics, this is fine. But until the possible worlds view was adopted, no-one considered the option of defining a semantics for first-order logic in which only some of the variable assignments may be taken into account. If one looks at the metatheory of logic, one discovers that some of the most famous results actually depend on the hidden assumption that S contains states built from all assignments. So the new insight has far-reaching implications for logic.

To conclude, the possible world semantics provides a unifying framework for all the logic languages we have investigated so far, and makes sense even if we do not choose to have modal operators.

Should you wish to look more deeply into the way a possible worlds view provides a fruitful foundation for logic, read Johan van Benthem: 'Modal Foundations for Predicate Logic', *Logic Journal of the Interest Group in Pure and Applied Logic (L.J. of the IGPL)* **5**(2):259-286 1997.

6 Afterthoughts

First-order languages offer a flexible tool for knowledge representation, and in many ways resemble the familiar propositional languages — for example, first-order languages satisfy a Compactness Theorem, although

we have not proved it. Similarly, it is possible to define reasoning algorithms satisfying first-order counterparts of the Soundness and Completeness Theorems. Nevertheless, the increased power and expressiveness of first-order languages come at a price, and notions such as satisfaction are more complicated.

In Artificial Intelligence some of the most popular first-order knowledge representation languages are situation calculi. The use of such languages revealed the Frame Problem, which may be thought of as follows: "Whenever something *might* change from one moment to another, we have to find some way of stating that it doesn't change whenever anything changes. And this seems silly, since almost all changes in fact change very little of the world. One feels that there should be a more economical and principled way of succinctly saying what changes an action makes, without having to explicitly list all the things it doesn't change as well; yet there doesn't seem to be any other way to do it."

The quote is from an article by Patrick Hayes: What the Frame Problem is and isn't. The article appears on pages 123-137 in the book Pylyshyn ZW (editor): *The Robot's Dilemma — The Frame Problem in Artificial Intelligence*, Ablex Publishing Company 1987. Unfortunately the book is not in our library, but if you're interested, you can get it on interlibrary loan.

The Frame Problem is one of the factors that led to the development of nonmonotonic logic, *i.e.* logics that formalise defeasible reasoning. Nonmonotonic logic allows the Frame Problem to be solved roughly as follows. Instead of having lots of axioms each saying that this or that doesn't change when such and such happens, one can simply use a preference ordering on interpretations in which the more normal states are those in which things stay the same unless the action was specifically aimed at them. So it becomes a defeasible belief that turning the ignition key of your car won't break the living-room window. And of course it should be only a defeasible belief, not an absolute axiom, because if your rival in crime has attached a bomb to your car, turing the key would under those exceptional circumstances cause your living-room window to shatter.

Situation calculi may seem to talk about time, but they are not temporal languages since the situation terms represent states, not time instants. Thus a situation calculus doesn't distinguish between a system remaining in a state for one instant or for a billion years. On the other hand, temporal logics don't say what causes a change of state, whereas the situation calculus explicitly talks about the actions that may cause such changes. A situation calculus is thus a kind of *dynamic* logic.