LANGUAGE AND PROOFS FOR PREDICATE LOGIC

1

QUESTIONS

1. For each of the following sentences, find a formula in the language of propositional logic that best represents its structure.

   Use the following dictionary: Fx is x *is a footballer*; Bx is x *is a biped*, a is *Sócrates*, Txy is x *is taller than* y, Oxy is x *is older than* y.

   a) Some footballer isn't a biped.
   b) No footballer is a biped.
   c) Some footballer is taller than Sócrates.
   d) No footballer is taller than any biped older than Sócrates.

2. Check that you understand how substitution works by performing these substitutions:

   a) $(Fx)[x/a]$
   b) $(Fx)[y/a]$
   c) $(\forall x(Fx \to Lxy))[y/b]$
   d) $(\exists xFx \wedge Gx)[x/y]$
   e) $((Fx \to \exists y(Lxy \wedge (Gz \vee \forall xHx)))[x/z])[z/a]$

   What is wrong with $(\forall x(Fx \to Lxy))[y/x]$?

3. Are these trees *proofs*? Check whether the rules are correctly applied, and particularly, whether the eigenvariable conditions are satisfied by each $\forall$I and $\exists$E step.

$$\dfrac{\dfrac{\dfrac{\forall x(Fx \wedge \exists y\neg Fy)}{Fa \wedge \exists y\neg Fy}\,\forall E}{\exists y\neg Fy}\,\wedge E \qquad \dfrac{\dfrac{\forall x(Fx \wedge \exists y\neg Fy)}{Fa \wedge \exists y\neg Fy}\,\forall E}{Fa}\,\wedge E \quad [\neg Fa]^1}{\dfrac{\bot}{\bot}\,\neg E}\,\exists E^1$$

$$\dfrac{\forall x(Fx \vee Gx)}{Fa \vee Ga}\,\forall E \qquad \dfrac{\dfrac{[Fa]^1}{\forall xFx}\,\forall I}{\forall xFx \vee \forall xGx}\,\vee I \qquad \dfrac{\dfrac{[Ga]^2}{\forall xGx}\,\forall I}{\forall xFx \vee \forall xGx}\,\vee I \qquad \vee E^{1,2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[Ga]^3 \quad [\neg Ha]^1}{Ga \wedge \neg Ha}\wedge I}{\exists x(Gx \wedge \neg Hx)}\exists I \qquad \neg\exists x(Gx \wedge \neg Hx)}{\bot}\neg E}{\neg\neg Ha}\neg I^1}{Ha}\text{DNE}}{Fa \vee Ha}\vee I$$

$$\dfrac{\dfrac{\forall x(Fx \vee Gx)}{Fa \vee Ga}\forall E \quad \dfrac{[Fa]^2}{Fa \vee Ha}\vee I \qquad \dfrac{Fa \vee Ha}{Fa \vee Ha}\vee I}{Fa \vee Ha}\vee E^{2,3}$$

$$\dfrac{Fa \vee Ha}{\forall x(Fx \vee Hx)}\forall I$$

4. Construct proofs for the following arguments:

   a) $\forall x(Fx \to Gx), \forall x(Gx \to Hx) \succ \forall x(Fx \to Hx)$
   b) $\exists x Fx \succ \forall x(Fx \to Gx) \to \exists x Gx$
   c) $\exists x Fx, \forall x Gx \succ \exists x(Fx \wedge Gx)$
   d) $\forall x(Fx \to x = a) \succ \exists x(Fx \wedge Gx) \to Ga$
   e) $\forall x \forall y \forall z((Rxy \wedge Ryz) \to Rxz), \neg\exists x Rxx \succ \forall x \forall y(Rxy \to \neg Ryx)$

   Warning: Proofs for the final one will be significantly longer than proofs for the rest.

5. Construct proofs for the following arguments. (Beware, for these you may need to employ DNE, and they might be quite complicated.)

   a) $\forall x(p \vee Fx) \succ p \vee \forall x Fx$
   b) $\neg\exists x(Fx \wedge \neg Gx), \neg\exists x(Gx \wedge \neg Hx) \succ \forall x(\neg Fx \vee Hx)$
   c) $\succ \exists x(Fx \to \forall y Fy)$

   Note that a proof for the last may be much more difficult to find than proofs for the first two.

KEY CONCEPTS AND SKILLS

- ☐ You need to understand the difference between predicates, names, variables, and quantifiers.
- ☐ You should be able to read and understand formulas in the language of predicate logic.
- ☐ You should understand the definition of *substitution*, when a term is free for a variable in a formula, and why the definition of substitution is restricted to the cases where the terms substituted in the formula be free for the variable substituted *for*.
- ☐ You should be able to *read* proofs using any or all of the quantifier rules, and be able to check that the eigenvariable condition is satisfied when the $\forall$I and $\exists$E rules are used.
- ☐ You should be able to *construct* simple tree proofs using all the inference rules.

MODELS FOR PREDICATE LOGIC AND SOUNDNESS

2

QUESTIONS

1. Consider a first-order language with names a and b and with predicates M (one-place), B (two-place). Take a model with domain $D = \{r, p, s\}$, where I interprets the names and predicates like this:

	I
a	r
b	p

	M
r	1
p	0
s	1

B	r	p	s
r	0	0	1
p	1	0	0
s	0	1	0

   (a) First, check the following formulas for truth or falsity in this model, explaining your reasoning:

$$Ma \quad Bab \quad Bab \to Bba$$

   (b) Now, for each of the following formulas

$$\forall y My \quad \exists x Bxa \quad \exists x(Bxa \wedge Bax)$$

   find its truth value in the model, explaining how you evaluate the quantifier using the different possible assignments of values to the variable bound by that quantifier. (For example, $\exists x Mx$ is *true* in this model, since there is *some* assignment of values to the variable x (namely, $x : r$) according to which Mx is true.)

   (c) Finally, assess the following formulas for truth or falsity in the model, being careful to respect the orders in which the quantifiers occur in each formula. Make sure to explain each step of your reasoning.

$$\forall x \forall y(Bxy \to \neg Byx) \quad \exists x \forall y Bxy \quad \forall x(Mx \vee \exists y(My \wedge Bxy))$$

2. Consider the language with a two-place predicate R. Consider the domain $D = \{k, l, m\}$, and the three different models for our language given by these three different ways to interpret R on that domain:

$I_1(R)$	k	l	m
k	0	0	1
l	1	0	0
m	0	1	0

$I_2(R)$	k	l	m
k	0	1	1
l	1	0	1
m	1	1	0

$I_3(R)$	k	l	m
k	0	1	1
l	0	0	1
m	0	0	0

So, we have three models: $M_1 = \langle D, I_1 \rangle$, $M_2 = \langle D, I_2 \rangle$ and $M_3 = \langle D, I_3 \rangle$. For each of the following arguments, establish which (if any) of M_1, M_2 and M_3 are *counterexamples* to those arguments.

(a) $\succ \forall x \forall y (Rxy \rightarrow Ryx)$

(b) $\forall x \exists y Rxy \succ \forall x Rxx$.

(c) $\forall x \exists y Rxy \succ \exists x \forall y Rxy$

3. Construct models (on a domain of whatever size you like) to give counterexamples to the following arguments — if they have counterexamples — and if they do not, explain why they do not, either by showing directly that there is no such counterexample, or by providing a *proof* for the argument.

(a) $\exists x Fx, \exists x Gx \succ \exists x (Fx \wedge Gx)$

(b) $\exists x Fx, \forall x Gx \succ \exists x (Fx \wedge Gx)$

(c) $\succ \exists x (Fx \rightarrow \forall y Fy)$

(d) $\forall x \exists y Rxy \succ \forall y \exists x Rxy$

(e) $\forall x \forall y (Rxy \rightarrow \exists z (Rxz \wedge Rzy)), \forall x \exists y Rxy, \forall x \neg Rxx \succ \bot$

4. Look at the proof of the *soundness theorem* in the Class Notes. In that proof, I went through the inductive steps for the rules $\wedge I$, $\rightarrow I$, $\vee E$ and $\forall I$. Write out the reasoning for (some of) the other rules too:

- Choose two (at least) of $\wedge E$, $\rightarrow E$, $\vee I$, DNE, and Refl first.
- Choose one (at least) of $\neg E$ and $\bot E$ next.
- Then check $\neg I$ (its reasoning should be a lot like that for $\rightarrow I$).
- Choose one of $\forall E$ and $\exists I$.
- And finally, check $\exists E$.

KEY CONCEPTS AND SKILLS

☐ You should understand the definition of a *model* for a first-order language (with a domain and an interpretation of each predicate and each name in the language)

☐ You should be able to check the truth or falsity of a formula in a model.

☐ You should understand the concept of validity in first-order predicate logic, and be able to construct counterexamples to simple invalid arguments, and for simple valid arguments, be able to demonstrate that they are indeed valid.

☐ You understand the soundness theorem, and have a grasp of how the soundness theorem is proved, and how to complete the steps we did not cover in the lectures or the text.

COMPLETENESS FOR PREDICATE LOGIC 3

QUESTIONS

1. Here are some examples of unprovable arguments (there is no proof from the premises to the conclusion). For each argument, find a *model* (a *domain* and the interpretation of the *predicates* and *names* of the language into in that domain) that makes the premises *true* and the conclusion *false*.

   (a) $\exists x(Fx \to Gx), \forall xFx \succ \forall xGx$

   (b) $\exists x\neg Fx, Ft_1, Ft_2, Ft_3, \ldots \succ \bot$ (where $t_1, t_2, \ldots$ are *all* the terms in the language).

   (c) $\forall x\forall y\forall z((Sxy \wedge Syz) \to Sxz), \forall x\exists ySxy, \forall x\exists ySyx, \neg\exists xSxx \succ \bot$.

   (d) $\forall x\forall y\forall z((Sxy \wedge Syz) \to Sxz), \forall x\forall y(Sxy \to \neg Syx), \forall x\exists ySxy, \forall x\exists ySyx, \neg\exists xSxx, \forall x\forall y(Sxy \to \exists z(Sxz \wedge Szy)) \succ \bot$

2. Here is a collection of statements which, when pieced together in the right order, form the skeleton of the proof of the completeness theorem. Decide the *best* way to link the statments together to construct the proof. Then explain the connections between the statements, as you have arranged them.

   (i) If $X \nvdash A$ then for some maximal A-avoiding and witnessed set X^* where $X \subseteq X^*$, we have $X^* \nvdash A$.

   (ii) If X^* is a maximal A-avoiding and witnessed set of formulas, then $\exists xB(x) \in X^*$ iff for some name n, $B(n) \in X^*$; $\forall xB(x) \in X^*$ iff for every name n, $B(n) \in X^*$.

   (iii) If X^* is a maximal A-avodinig set of formulas, then if $X^* \vdash B$, then $B \in X^*$.

   (iv) If $X \vDash A$ then $X \vdash A$.

   (v) If X^* is a maximal consistent and witnessed set X^*, then there is some model $\mathfrak{M}_{X^*}$ making true all the formulas in X^* and only those formulas.

   (vi) If $X \nvdash A$ then $X \nvDash A$.

   (vii) If X^* is a maximal A-avoiding set of formulas, then $\neg B \in X^*$ iff $B \notin X^*$; $B \wedge C \in X^*$ iff $B \in X^*$ and $C \in X^*$; $B \vee C \in X$ iff $B \in X^*$ or $C \in X^*$; $B \to C \in X^*$ iff $B \notin X^*$ or $C \in X^*$; and $\bot \notin X^*$.

3. One part of proving the completeness theorem is proving statement (i) in our previous question, showing that if $X \nvdash A$ then there

is some maximal A-avoiding and witnessed set $X^* \supseteq X$. This is shown by adding a collection of new names to the language, enumerating the expanded language, and considering each formula one-by-one for inclusion to the larger set we are forming, starting with our original set X. We add a formula if (and only if) when adding it, we still cannot prove A. And if the formula is existentially quanitifed (of the form $\exists x B(x)$) then if we add it, we also add $B(n)$ for some fresh name n.

The set X^* you end up with depends not just on the set X you start with, but also the order of formulas in the enumeration.

There is no way for us to constrcuct a set X^* in a seminar, but we can *start*. Consider the set $X = \{\forall x(Fx \vee Gx), \neg\forall xGx\}$, from which we cannot prove $\forall xFx$. Our aim will be to see what happens when we try to construct a maximal $\forall xFx$-avoiding set X^*, starting from the two-element set X. Add the names $c_1, c_2, c_3, \ldots$ to the language, and show that if you consider enumerations starting with:

$$Fc_1, \neg Fc_1, \forall xFx, \exists xFx, \forall xGx, \exists xGx, \ldots$$

and with

$$\neg Fc_1, Fc_1, \forall xGx, \exists xGx, \forall xFx, \exists xFx, \ldots$$

the sets you construct using this process (of considering each formula one-by-one, adding it to your set if you can, while avoiding $\forall xFx$) are different, and are inconsistent with each other.

4. Read these statements involving the identity predicate, and try to establish the logical relationships between the formulas. (You might find drawing a diagram useful.) I'll give you a hint to start. Formula (b) (which says that there are two different things with property F) entails formula (a) (which says that there is a thing with property F), but not *vice versa*.

   (a) $\exists xFx$

   (b) $\exists x\exists y(Fx \wedge Fy \wedge x \neq y)$

   (c) $\exists x\forall y(Fy \rightarrow x = y)$

   (d) $\exists x\exists y(Fx \wedge Fy \wedge Gx \wedge \neg Gy)$

5. Some people have asked for more proof practice, so here are some more for you to try, using the identity predicate. Find *proofs* for these arguments using the logical rules of identity.

   (a) $a = b, b \neq c \succ a \neq c$

   (b) $\forall x\forall y((Fx \wedge Fy) \rightarrow x = y), \exists x(Fx \wedge Gx) \succ \neg\exists x(Fx \wedge \neg Gx)$

   (c) $\neg\exists xFx \succ \exists x\forall y(Fy \rightarrow y = x)$

   (d) $Ix(Fx, Gx), Ix(Fx, Hx) \succ Ix(Fx, Gx \wedge Hx)$.

   Here, $Ix(A(x), B(x))$ is an abbreviation for the 'definite description' sentence $\exists x((A(x) \wedge \forall y(A(y) \rightarrow y = x)) \wedge B(x))$, which states that there is one and only one object x satisfying $A(x)$, and that thing satisfies $B(x)$ too: i.e. *the* A *is a* B.

KEY CONCEPTS AND SKILLS

- ☐ You should undersrtand what it takes for a set of formulas to be A-*avoiding*, to be *maximal* A-*avoiding*, and to be *witnessed*.
- ☐ You should understand the distinctive properties of maximal A-avoiding sets, witnessed maximal A-avoiding sets, and how these properties are proved.
- ☐ You should understand the completeness theorem, and have a grasp of how the completeness theorem is proved, and how to complete the steps we did not cover in the lectures or the text.
- ☐ You should understand the difference between the *soundness* and *completeness* theorems.
- ☐ You need to understand the syntax of identity statements, and function symbols.

FUNCTIONS, COUNTABILITY AND DIAGONALISATION

4

QUESTIONS

1. Consider these functions from ω (the natural numbers) to ω. Are they injective, surjective or bijective or none of these?

   - f, where $f(x) = x + 2$
   - g, where $g(x) = x/2$ when x is even, and $(x - 1)/2$ when x is odd.
   - h, where $h(x) = x + 1$ when x is even, and $x - 1$ when x is odd.
   - i, where $i(x) = 0$ when x is even and 1 when x is odd.

2. For each of the sets mentioned below, decide whether they are *finite*, *countable* or *uncountable*. In the case of a finite set, try to say how many elements the set has. In the case of a countable, set, try to specify an enumeration. In the case of an uncountable set, give an explanation of why this set is uncountable.

   (a) The set of all finite *sets* of symbols, taken from some finite alphabet A.

   (b) The set of all finite *strings* of symbols, taken from a finite alphabet A. (For example, if the alphabet A contains the symbols a, b and c, then the strings include a, b, c, ab, aa, c, ca, bb, ba, abc, $aababb$, $acbcbb$, ... etc.)

   (c) The set of all *finite trees* of elements of a set C, where the set C is countable.

   (d) The set of all formulas in the language of predicate logic, with one-place predicates F, G, two-place predicates R and $=$, the names a, b and c, and a countable collection of variables x_0, $x_1, x_2, \ldots$

   (e) The set of *proofs* in predicate logic, where the formulas in these proofs are taken from a finite language.

3. For these sets of *functions*, decide whether they are *finite*, *countable* or *uncountable*. In the case of a finite set, try to say how many elements the set has. In the case of a countable, set, try to specify an enumeration. In the case of an uncountable set, give an explanation of why this set is uncountable.

   (a) The set of all functions from a *finite* set A to a *finite* set B.

(b) The set of all *bijections* from a *finite set* A to a finite set B.

(c) The set of all functions from a *finite* set A to a *countable* set E.

(d) The set of all functions from a *countable* set E to a *finite* set A.

(e) The set of all of the *functions* from a *countable* set D to a *countable* set E.

4. Recall that a set T of formulas in some language $\mathcal{L}$ is a *theory* if and only if for each formula A in $\mathcal{L}$, if $T \vdash A$, then $A \in T$. How many different theories are there in the following languages?

   (a) Take the language of all formulas made out of the propositional atoms p, q and r, operated on with the connectives $\wedge$, $\vee$, $\rightarrow$ and $\neg$.

   (b) Take the language of all formulas made out of the countable collection of propositional atoms p_0, p_1, p_2, ..., operated on with the connectives $\wedge$, $\vee$, $\rightarrow$ and $\neg$.

   (c) Take the language of all formulas made from the one place predicate F, the two-place predicates R and $=$, the names a, b and c and a countable supply of variable, using the connectives and the quantifiers.

KEY CONCEPTS AND SKILLS

☐ You need to be able to work with function symbols, and get comfortable with the syntax of arithmetic, with the function symbols $'$, $+$ and $\times$.

☐ You need to understand the difference between *injective*, *surjective* and *bijective* functions.

☐ You need to understand the definition of an *enumeration*, and how this is used to define the class of *countable* sets.

☐ You need to understand the difference between finite, countably infinite and uncountable sets, and to be able to classify simple sets as finite, countably infinite, or uncountable.

☐ You need to understand Cantor's Diagonalisation argument, and how it proves that some sets are uncountable.

5 COMPACTNESS AND THE DOWNWARD LÖWENHEIM–SKOLEM THEOREM

QUESTIONS

1. This task will be to apply the compactness theorem to show that another concept cannot be expressed in the language of predicate logic. This is the concept of a *descending* $<$*-chain*. There is a descending $<$-chain if and only if there is an infinite sequence a_0, a_1, a_2, a_3, a_4, ... such that

$$\cdots \quad a_4 < a_3 \quad a_3 < a_2 \quad a_2 < a_1 \quad a_1 < a_0$$

   For the first part of this task, consider the model $\langle \omega, I \rangle$ of the language containing the one-place relation $<$ (with a countable supply of variables, the identity predicate, and the usual connectives and quantifiers) with the domain consisting of the natural numbers, and where $I(<)(n, m)$ is 1 if and only if the natural number n is indeed less than m. Verify that there is no descending $<$-chain in this model.

   Once you've done that, consider the set of sentences in this language (containing the relation $<$, the identity predicate, the variables, quantifiers and connectives) that is true in the model $\langle \omega, I \rangle$. Call that set of sentences $O_{<,\omega}$ (the *order* facts about $<$ in ω). Then add the names a_0, a_1, a_2, ... to our language, and consider the set

$$O_{<,\omega} \cup \{a_1 < a_0, a_2 < a_1, a_3 < a_2, a_4 < a_3, \ldots\}$$

   Use the *compactness theorem* to show that this set is consistent.

   Finally, explain why this shows that there is no sentence in the langauge of predicate logic (with identity) which is expresses the fact that there is no descending $<$-chain. (That is, there is no sentence in the language of predicate logic, with identity, that is true in a model iff there is no descending $<$-chain in that model.)

2. In this task you explore consequences of the Downard Löwenheim–Skolem Theorem. We start with the model $\langle \mathbb{R}, I \rangle$ whose domain is the set of *real numbers*. These are the numbers that we use to measure *continuous quantities*. We represent them by, possibly infinite, strings of decimal numerals, including a decimal point, and either prefixed by a '–' (for the negative numbers), or not. A key feature of these real numbers is not only that we can add, subtract, multiply and divide them freely, but that bounded sets or sequences of these

numbers have a number at their *limit*. In the usual language that we have for real numbers, we have at least the following concepts:

$$0 \qquad 1 \qquad + \qquad \times \qquad =$$

First, show how we do not need to have separate symbols for *subtraction* or *division*, because we can already find a sentence in our language equivalent to

$$x \textit{ minus } y \textit{ equals } z$$

and one equivalent to

$$x \textit{ divided by } y \textit{ equals } z$$

Once you've done that, show that we can also find a sentence equivalent to $x < y$. (HINT: you might like to observe the fact that in our model $\langle \mathbb{R}, I \rangle$, for any number x, its square, $x \times x$ is always greater than or equal to zero.)

And lastly, show that there is a sentence equivalent to

$$x \textit{ is the square root of } y$$

We can apply Cantor's Theorem to show that $\mathbb{R}$ is uncountable. However, the *theory* of all sentences in our language that are true in the model $\langle \mathbb{R}, I \rangle$ is, by the Downward Löwenheim–Skolem Theorem, also true in some countable model.

What is such a model *like*? How does it differ from $\langle \mathbb{R}, I \rangle$? What objects must there be in its domain? What, if anything, is 'missing' from this model?

KEY CONCEPTS AND SKILLS

- ☐ You must be able to *state* the compactness theorem, and you should be able to reconstruct the structure of its proof, and undersand the proof in detail.
- ☐ You should understand what it means for a concept to be *expressible* in the language of first order predicate logic.
- ☐ You should understand how the compactness theorem is used in the proof that some particular concepts (such as the 'there are infinitely many' quantifier) are not expressible in the language of first order predicate logic.
- ☐ You must be able to *state* the Downward Löwenheim–Skolem Theorem, and you should be able to reconstruct the structure of its proof, and undersand the steps of that proof.

SECOND-ORDER LOGIC 6

QUESTIONS

1. Recall the *rock, paper, scissors* model from Week 2 Question 1. Let's use this model to evaluate the *second*-order language, with the same defined predicates M and B. Verify that the following statements are true in the second-order language defined on this model, taking care to explain your reasoning in terms of the rules we use to interpret second-order quantifier expressions.

$$\exists f_1 \forall x B f_1(x) x \qquad \exists f_2 \forall x B x f_2(x)$$

(I have used parentheses to make clear that in these quantifier expressions f_1 and f_2 are *one*-place function variables.)

2. In this question, for any two-place predicates P and Q, *Repeat*(P, Q) is the formula:

$$\forall x \forall y (Pxy \rightarrow Qxy) \wedge \forall x \forall y \forall z ((Pxy \wedge Qyz) \rightarrow Qxz)$$

and, when P^* is another two-place predicate, *TC*(P, P^*) is the second-order formula:

$$\textit{Repeat}(P, P^*) \wedge \forall Q(\textit{Repeat}(P, Q) \rightarrow \forall x \forall y (P^* xy \rightarrow Qxy))$$

Consider the following model on the domain $D = \{\alpha, \beta, \gamma, \delta, \epsilon, \zeta\}$. (These Greek letters are *alpha, beta, gamma, delta, epsilon* and *zeta*.)

I(R)	α	β	γ	δ	ϵ	ζ
α	0	0	1	0	0	0
β	0	0	1	0	0	0
γ	0	0	0	1	0	0
δ	0	0	0	0	1	1
ϵ	0	0	0	0	0	0
ζ	0	0	0	0	0	0

I(S)	α	β	γ	δ	ϵ	ζ
α	0	1	1	1	1	1
β	0	0	1	1	1	1
γ	0	0	0	1	1	1
δ	0	0	0	0	1	1
ϵ	0	0	0	0	0	1
ζ	0	0	0	0	0	0

First: verify that *Repeat*(R, S) holds in this model, explaining your work, step-by-step.

Second: Find some interpretation for a predicate R^* such that *TC*(R, R^*) is true.

Challenge: Is $\forall R \exists S$ *TC*(R, S) true in every second-order model? Why or why not?

3. If f is an n-place function symbol, interpreted in some model M, we can interpret an $n+1$ predicate E_f *corresponding* to f in the following way:

$$I(E_f, \alpha)(d_1, \cdots, d_n, e) = 1 \text{ iff } I(f, \alpha)(d_1, \ldots, d_n) = e$$

That is, E_f is the $n+1$-place predicate describing the *extension* of the function f. For example, for the two-place predicate $+$, E_+ is a three-place predicate, where E_+xyz states that x plus y is z.

First: verify that for any m-place predicate R, whenever x is not free in the terms s_i and t_j:

$$\begin{aligned} I(Rs_1 \cdots s_{l-1}(ft_1 \cdots t_n)s_{l+1} \cdots s_m, \alpha) \quad = \\ I(\exists x(E_f t_1 \cdots t_n x \wedge Rs_1 \cdots s_{l-1} x s_{l+1} \cdots s_m, \alpha) \end{aligned}$$

That is, we can replace a function-application term somewhere inside a predication using an existental quantifier binding a fresh variable, and fixed by the predicate correspinding to the function term. We can use this equivalence repeatedly to rewrite formulas with function symbols into formulas using only predicates. Here is an example:

The formula $x + (y \times z)' = w$ is equivalent to

$\exists x_1(E_\times(y, z, x_1) \wedge x + x_1' = w)$, which is equivalent to

$\exists x_1(E_\times(y, z, x_1) \wedge \exists x_2(E_\prime(x_1, x_2) \wedge x + x2 = w))$, which is, in turn, equivalent to

$\exists x_1(E_\times(y, z, x_1) \wedge \exists x_2(E_\prime(x_1, x_2) \wedge \exists x_3(E_\times(x, x_2, x_3) \wedge x_3 = w)))$, and this formula contains *no* function terms.

Your task: do the same thing, to find a function-term-free formula equivalent to $(x + y') = (x + y)'$, using the predicates $E_\prime$ (two-place) for the extension of the successor function, and E_+ (three-place) for the extension of the addition function.

KEY CONCEPTS AND SKILLS

- ☐ You must be able to understand the *syntax* of second-order formulas, recognising the difference between variables for terms (first-order) and predicates and function-symbols (second-order).
- ☐ You must understand the *truth conditions* for second-order formulas, and be able to evaluate simple second-order formulas (relative to assignments of values to variables) in a given small second-order model.
- ☐ You must understand the increased expressive power of the second-order vocabulary, including how the concepts of *identity*, *finitude* and *countability* can be expressed by second-order formulas.
- ☐ You must understand how this means that second-order logic must not satisfy the conditions for the compactness theorem to apply, and hence, that there is no sound and complete finitary proof system for second-order logic.

RECURSIVE FUNCTIONS 7

QUESTIONS

1. Read these definitions of functions, and answer the following questions about them. (1) How many inputs does the function take? (2) Can you tell immediately whether the function is *total* (defined on every input) or possibly *partial* (undefined on some inputs) (3) Calculate the value of the function on a few simple inputs. (4) Can you describe the function that is calculated, in more familiar terms?

   Recall: z is the one-place *zero* function ($z(x) = 0$). s is the *successor function* ($s(x) = x + 1$). id^n_3 is the n-place function that takes n inputs and returns the mth of them.

   $\mathrm{Cn}[f, g_1, \ldots, g_m]$ composes one m-place function f with m n-place functions g_1 to g_m, so

   $$\mathrm{Cn}[f, g_1, \ldots, g_m](x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

   $\mathrm{Pr}[f, g]$ defines an $n+1$ place function by primitive recursion with the n-place base case function f and $n+2$-place step function g, so

   $$\mathrm{Pr}[f, g](x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$$
   $$\mathrm{Pr}[f, g](x_1, \ldots, x_n, y{+}1) = g(x_1, \ldots, x_n, y, \mathrm{Pr}[f, g](x_1, \ldots, x_n, y))$$

   $\mathrm{Mn}[f]$ defines an n-place possibly *partial* function by minimising the $n+1$-place function f. So,

   > $\mathrm{Mn}[f](x_1, \ldots, x_n)$ is the least y where $f(x_1, \ldots, x_n, y) = 0$, if there is such a y, and $\mathrm{Mn}[f](x_1, \ldots, x_n)$ is undefined, if there is no such y.

   a) $\mathrm{Cn}[z, \mathrm{id}^3_1]$
   b) $\mathrm{Cn}[\mathit{sum}, \mathrm{id}^3_3, \mathrm{Cn}[z, \mathrm{id}^3_1]]$ (where *sum* is the two-place addition function)
   c) $\mathrm{Cn}[\mathit{prod}, \mathrm{id}^3_2, \mathrm{Cn}[\mathit{sum}, \mathrm{id}^3_3, \mathrm{Cn}[z, \mathrm{id}^3_1]]]$ (where *prod* is the two-place product function)
   d) $\mathrm{Pr}[\mathrm{Cn}[s, z], \mathrm{Cn}[\mathit{prod}, \mathrm{id}^3_1, \mathrm{id}^3_3]]$
   e) $\mathrm{Mn}[\mathrm{Cn}[\mathit{rsub}, \mathrm{id}^2_1, \mathrm{Cn}[f, \mathrm{id}^2_2]]]$ (where *rsub* is the two-place restricted subtraction function, and f is some one-place function)

2. The *triangular numbers* are defined like this: $t(0) = 0$, $t(1) = 0 + 1$, $t(2) = 0 + 1 + 2$, $t(3) = 0 + 1 + 2 + 3$, … $t(n + 1) =$

$t(n)+n+1$, and so on. Using the fact that the addition function is $\mathrm{Pr}[\mathrm{id}^1_1, \mathrm{Cn}[s, \mathrm{id}^3_3]]$, write out a full definition of the triangular number function t, using the definition of the addition function, and the fact that t is defined by primitive recursion.

Once you've done that, it should be easy to define the *factorial* function, given as follows: $0! = 1$, $1! = 1 \times 1 = 1$, $2! = 1 \times 2$, ... $(n+1)! = n! \times (n+1)$, and so on. Feel free to use the definition of the multiplication function: $\mathrm{Pr}[z, \mathrm{Cn}[\mathit{sum}, \mathrm{id}^3_3, \mathrm{id}^3_1]]$, where *sum* is the addition function.

3. Spend some time thinking (and talking) about how to convert a definition of a recursive function (in terms of the basic functions and constructors) into a recipe instructing someone how to *calculate* the value of the function, given specific inputs. What kinds of information does someone who follows such a recipe need to keep track of while following it?

KEY CONCEPTS AND SKILLS

- ☐ You must be able to *read* definitions of recursive functions involving the basic functions and the constructors. To do this, you need to be able to determine the *arity* of such a function (how many inputs it expects) and to determine the output of the function on small inputs.
- ☐ You should understand how to rewrite specifications of simple recursive functions in terms of a formal definition in terms the basic functions and the constructors.

REGISTER MACHINES 8

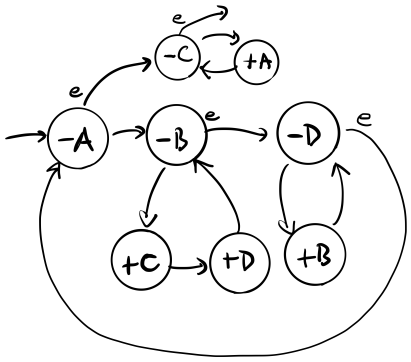
QUESTIONS

These questions ask you to work in *groups*. You can work in groups of size two to four. Use the break in the middle of Tuesdays lecture class to form your group. (If you miss the lecture and *cannot* find a group to join for yourself, email me by the end of Thursday and I'll find a group for you.) Each member of a group should submit the *same* PDF file, which should contain at the top of the first page the student numbers of every group member. (This is to ensure that the mark in MMS can be attached to the submission.)

1. Consider the following two register machine programs. Draw flow graphs for the programs. In your groups, select a few different initial states of the registers, and in each case, see if you can describe what the programs execute.

1. $\langle -A; 2, - \rangle$	1. $\langle -A; 2, - \rangle$
2. $\langle +B; 3 \rangle$	2. $\langle -B; 3, 1 \rangle$
3. $\langle -A; 4, 2 \rangle$	3. $\langle +C, 2 \rangle$
4. $\langle +B; 1 \rangle$	

2. Here is the program for a register machine.



   I claim that this program calculates the *multiplication function*: whenever you start the machine with n rocks in register A, and m rocks in register B, it will terminate execution of the program with $n \times m$ rocks in register A.

   Discuss this in your groups. What can you do to *conclusively prove* that that this program calculates the multiplication function, or, if I have made a mistake, how you could *conclusively refute* the claim.

3. Design a register machine that, when given two numbers as inputs, returns the smaller of the two as an output.

KEY CONCEPTS AND SKILLS

- ☐ You must be able to *read* register machine programs.
- ☐ You should be able to to convert flow diagrams for register machines to lists of instructions, and *vice versa*.
- ☐ You should be able to execute simple register machine programs on small input values.
- ☐ You should understand the difference between program *termination* and programs that continue executing indefinitely.
- ☐ You should be able to understand what it is for a register machine to compute a (possibly partial) function, and be able to verify for a simple register machine that computes a given function, that the register machine indeed computes that function.
- ☐ You should understand, at a high level, how we show that register machines compute all the recursive functions, and how all recursive functions can be computed by a register machine. .

ROBINSON'S ARITHMETIC 9

QUESTIONS

1. Consider each of the following *theories*, and figure out which of the properties (*consistent*, *complete*, *deductively defined*, *finite model property*) each theory has.

   a) The theory consisisting of all *logically true* sentences in the language of arithmetic. (This is the set of sentences in the language $=$, $\underline{0}$, $'$, $+$, $\times$, with the usual quantifiers and connectives.)

   b) Consider this theory describing the legal states of the game *noughts and crosses*. The language consists of nine names (p_1, p_2,…, p_9) for each position on the board, predicates O (nought) and X (cross) for describing whether a position is filled with a nought or a cross, and two proposition letters (or 0-place predicates) N and C which indicate whether 'noughts' or 'crosses' have won. The theory consists of all statements which are true in *every intended model*, where an *intended model* of the theory is the state of the game board at some after some number of moves.

   So, for example, if 'noughts' has moved first and placed an O in the first (top left) square, but 'crosses' has not moved yet, then Op_1 is true, but in *this* model Xp_2 and Op_2 is false. Our *theory* consists of all the formulas true in *all* the intended models. So, Op_1 is not in our theory (it's true in some of these models and false in others). However, $(Op_1 \wedge Op_2 \wedge Op_3) \rightarrow N$ *is* in the theory because in any model in which there are Os in the first three positions, noughts has won. Similarly, $\neg\exists x(Ox \wedge Xx)$ is true, since no square *both* has an O and an X at any stage of the game. So is $p_1 \neq p_2$ since in any state of the game, p_1 and p_2 are different positions on the board.

   c) The theory of the real line, in the language of arithmetic. This is the set of all sentences true in the model with the domain $\mathbb{R}$ of all *real numbers*, with the same language we used for Robinson's arithmetic, where each term has its usual meaning, except we add an extra predicate 'N' which applies to all and only the natural numbers (0, 1, 2, etc.) So, in this theory, different sentences will be true: for example, this theory contains $\exists x(x' = \underline{0})$, since there is a real number (namely -1) where that number, plus one, is 0. Similarly, it contains $\exists x(x \times x = \underline{0}'')$, since there is a square root of two — a number that when you multiply it by itself, you get 2, whereas $\exists x(Nx \wedge x' = \underline{0})$ is false in this model, since -1 is not a natural number.

2. Here are the axioms of Robinson's arithmetic:

   [SUCCESSOR AXIOMS]
   Q1: $\forall x \forall y(x' = y' \rightarrow x = y)$;

Q2: $\forall x(\underline{0} \neq x')$;
Q3: $\forall x\big(x \neq \underline{0} \to \exists y(x = y')\big)$.

[ADDITION AXIOMS]
Q4: $\forall x(x + \underline{0} = x)$;
Q5: $\forall x \forall y\big(x + y' = (x + y)'\big)$.

[MULTIPLICATION AXIOMS]
Q6: $\forall x(x \times \underline{0} = \underline{0})$;
Q7: $\forall x \forall y(x \times y' = (x \times y) + x)$.

You can use these axioms in proofs, like this: a proof that $\underline{2}+\underline{1} = \underline{3}$, or in long hand, $\underline{0}'' + \underline{0}' = \underline{0}'''$.

$$\dfrac{\dfrac{\dfrac{\text{Q5}}{\underline{0}'' + \underline{0}' = (\underline{0}'' + \underline{0})'}\,\forall E \qquad \dfrac{\dfrac{\dfrac{\text{Q4}}{\underline{0}'' + \underline{0} = \underline{0}''}\,\forall E \qquad \dfrac{[F\underline{0}''']^1 \qquad [F\underline{0}''']^2}{\underline{0}''' = \underline{0}'''}\,{=}I^{1,2}}{(\underline{0}'' + \underline{0})' = \underline{0}'''}\,{=}E}{\underline{0}'' + \underline{0}' = \underline{0}'''}\,{=}E$$

Write your own proofs of $\underline{2} + \underline{1} \neq \underline{2}$ and $\underline{1} \times \underline{2} = \underline{2}$.

3. Remember, a function $f : \omega^n \to \omega$ is represented by the formula $\phi(\vec{x}, y)$ with the variables $\vec{x}$ and y free, in the theory T whenever

$$\text{If } f(\vec{n}) = m \text{ then } T \vdash \forall y(\phi(\underline{\vec{n}}, y) \leftrightarrow y = \underline{m}).$$

Here are four formulas:

a) $(x_1 = \underline{0} \wedge y = \underline{0}) \vee (x_1 = y')$
b) $(x_1 < x_2 \wedge y = \underline{0}) \vee (x_1 = x_2 + y)$
c) $(x_2 = \underline{0} \wedge y = \underline{0}) \vee \exists u(u < x_2 \wedge x_1 = y \times x_2 + u)$
d) $(x_2 = \underline{0} \wedge y = x_1) \vee (y < x_2 \wedge \exists u(u \leq x_1 \wedge x_1 = u \times x_2 + y)$

(Here $s < t$ is short for $\exists z(s + z' = t)$, where z is the first variable not present in s or in t, and $s \leq t$ is short for $s < t'$.)

What *functions* do each of these formulas represent in Q?

KEY CONCEPTS AND SKILLS

☐ You should understand the key properties of theories: *consistency*, *completeness*, being *deductively defined*, and having the *finite model property*, and be able to check, for a given simple theory, whether it has each property or fails to have it.

☐ You should understand what it is for a theory to be *decidable*, and understand, and be able to explain the proofs that deductively defined theories with the finite model property (in a finite vocabulary) are decidable, and that complete deductively defined theories are also decidable.

- ☐ You should be familiar with the axioms of Robinson's arithmetic (Q), and be able to deduce simple theorems from these axioms.
- ☐ You should understand how we can use formulas in the language of Q to represent recursive functions.

NON-RECURSIVE FUNCTIONS 10

QUESTIONS

1. The diagonalisation proof shows that there is a non-recursive function by constructing the diagonal function f^*, from an enumeration $f_0, f_1, f_2, \ldots$ of all (total) recursive functions, by setting $f^*(n) = f_n(n) + 1$, cannot be on the list of all recursive functions. First, explain why this argument cannot be used to show that there is a non-recursive *partial* function.

   Next, try to explain why there is a *universal* two-place partial recursive function $F : \omega \times \omega \rightharpoonup \omega$ that has the following interesting (incredible) property: for any (possibly partial) one-place recursive function f, there is some n where for all m, $f(m) = F(n, m)$. (HINT: to do this, try to explain how you could compute a function like F.)

2. Recall, the Ackermann function A, is a two-place function defined like this:

   - $A(0, n) = n + 1$
   - $A(m + 1, 0) = A(m, 1)$
   - $A(m + 1, n + 1) = A(m, A(m + 1, n))$

   This is a recursive function. (Althought it is not straightforward to make a register machine that calculates A, it can be done.) However, it is *not* primitive recursive.

   First task: Try calculating the value of $A(3, 1)$, and see how the recursion goes through *many* more steps than just 3. This raises the question of whether the calculation for A terminates, or whether the calculation might go on forever.

   Second task: Discuss in your groups how you might prove that the function A is total and not partial, that for each possible input n and m, $A(n, m)$ is number.

3. It's important to understand the difference between *recursive* sets and *recursively enumerable* sets. A set $S \subseteq \omega$ is RECURSIVE if and only if its *characteristic function* (the function f_S defined by the condition that $f_S(x) = 1$ if x is in S and $f_S(x) = 0$ otherwise) is recursive. A set $S \subseteq \omega$ is RECURSIVELY ENUMERABLE if and only if there is some recursive *enumeration* $e : \omega \to S$.

   First task: explain why every (non-empty) recursive set is recursively enumerable. (Explain how you could find an enumeration of your set S given a recursive characteristic function f_S.)

Second task: Here is another example that will help you get a feel for difference between recursive and recursively enumerable sets. Remember that a natural number n is *prime* if the only numbers that evenly divide into n are 1 and n itself. So 2, 3 and 5 are prime, but 4, 6 and 8 are not. Let's call a number natural m a *prime gap* if it is the distance between two successive prime numbers. So, 1 is a prime gap, since $1 = 3 - 2$, and so is 2, since $2 = 5 - 3$. Explain why the set of prime numbers is recursive, and why the set of prime gaps is recursively enumerable.

KEY CONCEPTS AND SKILLS

- You need to understand why there are non-recursive functions—since there are countably many recursive functions, and uncountably many functions, most functions are non-recursive.
- You need to know how *diagonalisation*, the *halting problem* and the *busy beaver* function give rise to non-recursive functions.
- You should understand the behaviour of the Ackermann function, as an example of a fast-growing recursive (but not primitive recursive) function.
- You should understand the notion of a characteristic function, and the difference between recursive and recursively enumerable sets.

11 GÖDEL NUMBERING, DIAGONALISATION AND ITS CONSEQUENCES

QUESTIONS

1. For each item in the list below, say as much as you can about what *kind of thing* it is.

   - 5
   - $\underline{0}''''$
   - $\exists y(y + \underline{0}' = x)$
   - The Gödel number of $\exists y(y + \underline{0}' = x)$
   - $\ulcorner \exists y(y + \underline{0}' = x) \urcorner$
   - The diagonalisation of $\exists y(y + \underline{0}' = x)$
   - The function *diag*
   - A formula that represents the function *diag* in Q.

2. Take the list of facts presented here, and explain the connections between them (by drawing a tree, if you can), and put in your own words how each fact has been proved.

   a) Any consistent extension of Q can define all recursive sets.

   b) Any consistent, deductively defined extension of Q is incomplete.

   c) The set of theorems of any consistent extension of Q is not recursive.

   d) Any complete and consistent deductively defined theory has a recursive set of theorems.

   e) Predicate logic is undecidable.

   f) Truth in the standard model of arithmetic is not recursively enumerable.

   g) Any consistent extension of Q cannot define the set of its own theorems.

   h) True arithmetic is not deductively defined.

   i) Truth in the standard model of arithmetic is not recursive.

   j) If the set of theorems of predicate logic is recursive, so is Q.

   (Don't look up the notes or the video/slides for the answer to this question before attempting it.)

KEY CONCEPTS AND SKILLS

- ☐ You should understand the techniques of Gödel numbering, and how the diagonalisation of a formula is defined.
- ☐ You should be able to state the Diagonalisation Lemma, and follow (and reproduce) its proof.
- ☐ You should understand the consequences of the Diagonalisation Lemma, such as the undecidability of Q, the incompleteness of Q and any of its consistent deductively defined extensions, the undecidability of first order predicate logic and the indefinability of arithmetic truth in the language of arithmetic.

12 PROVABILITY PREDICATES AND BEYOND

QUESTIONS

1. The relation "x is the Gödel number of a PA-proof of a formula with Gödel number y" is a recursively definable relation. It is represented in PA by some formula $\mathrm{Pr}(x, y)$ with two free variables. Why does the formula $\exists x \mathrm{Pr}(x, y)$ *not* represent, in PA, the set of Gödel numbers of formulas that are provable in PA? (Remember, *that* set is not recursive, and so, it cannot be represented in PA.)

2. Compare and contrast these three theories:

   - PA
   - $\mathrm{PA} + \mathrm{Con}(\mathrm{PA})$
   - $\mathrm{PA} + \neg\mathrm{Con}(\mathrm{PA})$

   What can you say about each theory?

3. What makes a logical theory a *good* theory? In the light of these limitative results, what is logic good *for*? Where are its limits?

KEY CONCEPTS AND SKILLS

☐ You should understand the notion of a provability predicate, how this differs from an internal representation of the set of theorems of a logic.

☐ You should understand and be able to reproduce the proof of Löb's Theorem, and you should be able to grasp some of its significance.

☐ You should understand the kind of incompleteness which is essential to deductively defined arithmetic theories.

☐ You should be able to reflect on the power and limits of logic, and its connections to allied fields, in philosophy, mathematics, linguistics, computer science, etc.