COM3412: Logic and Computation

Specimen answers to the 2001 examination

1. (a) i. The Halting Problem: Given an arbitrary Turing machine M and input tape t, to determine whether or not M will halt when run with t.

   ii. The Decision Problem for First-order Logic: Given an arbitrary set of formulae of first-order logic, and an arbitrary formula, to determine whether the latter is a logical consequence of the former. [Alternative version: Given an arbitrary set of first-order formulae, to determine whether or not it is satisfiable, i.e., whether there is an interpretation under which all the formulae in the set come out true.]

   The practical importance of the Halting Problem is that since computer programs are equivalent to Turing machines, the Halting Problem applies to them too. Non-termination is an obviously undesirable feature for a program intended to produce a specific result, and it would be useful to be able to detect it in advance. Non-termination manifests itself in practice when the system "freezes" and will not respond to further input.

   The practical importance of the Decision Problem is that many everyday information processing problems can be formulated as first-order inferences, so if we had a general method of testing such inferences for validity then we would be able to solve such problems, and maybe get computers to do more of our reasoning for us.

   (b) A Turing machine together with its tape, as well as the step-by-step operation of the machine, can be comprehensively described by means of first-order formulae. The machine halts if and only if the formula stating that it halts after some number of steps is a logical consequence of the formulae describing the machine and it operation. Hence if we can solve the Decision Problem, we can solve the Halting Problem as well.

   Conversely, given a first-order inference we could use a method like Truth Trees to test it for validity. If the inference is valid, then this process will always terminate, but if it is invalid it may or may not terminate. If we can solve the Halting Problem then we can test this process for termination first; if it is non-terminating, then we know the inference is invalid, whereas if it terminates we can run it to determine whether or not the inference is invalid. Hence if we can solve the Halting Problem, we can solve the Decision Problem as well.

   Thus the two problems are equivalent.

   (c) We know that we cannot use a Turing Machine to solve the Halting Problem (by the standart *reduction ad absurdum* argument), and so if the Church-Turing thesis is correct, there is no effective procedure for solving it. By the equivalence of the two problems, it follows that there is no effective procedure for solving the Decision Problem either. Only if we were to discover some method of computation which was (a) effective and (b) non-Turing Equivalent, would there be any chance of solving either of these problems.

2. (a) Assume the following non-logical vocabulary:

Constant: 0
Unary function symbol: s
Binary function symbols: $+$, $\times$

The first-order theory of arithmetic consists of all formulae over this vocabulary, in the first-order predicate calculus with identity, which are satisfied under the standard interpretation with domain $\mathbb{N}$ and

0 means zero
s means the successor function
$+, \times$ mean addition and multiplication respectively.

(b) It means that there is no finitely-specifiable set of first-order formulae such that the first-order theory of arithmetic consists of all and only the logical consequences of those formulae.

The key step in the proof is the 'arithmetisation of syntax', by which formulae and sequences of formulae are assigned natural numbers ('Gödel numbers') in such a way that the sequence of symbols in each formula (or sequence of formulae) is recoverable from the sequence of exponents in the unique prime factorisation of its Gödel number. This enables one to construct formulae which may be interpreted as saying things about other formulae (or indeed themselves). It is then possible to write down a formula which says that (i.e., is true if and only if) that very formula cannot be proved in the formal system under consideration. If true, this formula is *true but unprovable*, testifying to the incompleteness of the system, while if it is false, it is *false but provable*, testifying to the unsoundness of the system — either way the system cannot be both sound and complete.

(c) By **M1** we have

(I) $s0 * 0 = 0$.

(This is the base case.) Suppose we have

(IH) $s0 * a = a$.

(This is the induction hypothesis.) Then

$$\begin{array}{rcll} s0 * sa & = & (s0 * a) + s0 & \text{(by \textbf{M2})} \\ & = & a + s0 & \text{(by \textbf{IH})} \\ & = & s(a + 0) & \text{(by \textbf{A2})} \\ & = & sa & \text{(by \textbf{A1})} \end{array}$$

Hence $s0 * a = a \rightarrow s0 * sa = sa$, and since this holds for all values of a we have

(II) $\forall x(s0 * x = x \rightarrow s0 * sx = sx)$

By **Ind**, with $\Phi(x) \equiv s0 * x = x$, we have

(III) $s0 * 0 = 0 \wedge \forall x(s0 * x = x \rightarrow s0 * sx = sx) \rightarrow \forall x(s0 * x = x)$.

Together, **I**, **II**, and **III** imply $\forall x(s0 * x = x)$ as required.

(d) The axioms (including the infinitely many substitution-instances of the induction schema) of Peano Arithmetic constitute a finitely-specifiable set of formulae. By Gödel's theorem, the complete set of logical consequences of these formulae cannot be identical with the first-order theory of arithmetic. It follows that *either* there are formulae of Peano Arithmetic which are not true statements about natural numbers (and hence that at least one of the axioms is false), *or* there are true statements about the natural numbers, expressible in the language of Peano Arithmetic, which cannot be proved within Peano Arithmetic.

3. (a) **OE** says that if x and y both overlap exactly the same things, then they are equal.
PO says that x is part of y if and only if everything which overlaps x also overlaps y.

(b) **PR** From $O(z, x)$ we can infer $O(z, x)$, hence $O(z, x) \rightarrow O(z, x)$ is true, and since this applies to any z we have $\forall x(O(z, x) \rightarrow O(z, x))$. By **PO**, this implies $P(x, x)$, and since x is general we have $\forall x P(x, x)$.

PT Suppose $P(x, y) \wedge P(y, z)$. If $O(u, x)$, then by **PO**, we have $O(u, y)$ from $P(x, y)$, and hence also $O(u, z)$ from $P(y, z)$. Hence we have $O(u, x) \rightarrow O(u, z)$, and since this holds for all u we have $P(x, z)$ by **PO**. Hence $P(x, y) \wedge P(y, z) \rightarrow P(x, z)$. This holds for all x, y, z, giving **PT**.

PA Suppose $P(x, y) \wedge P(y, x)$, and let $O(x, z)$. By **OS** this means that $O(z, x)$, and by **PO**, since we have $P(x, y)$, we obtain $O(z, y)$. By **OS**, this gives $O(y, z)$, and hence we have $O(x, z) \rightarrow O(y, z)$. Similarly, using $P(y, x)$ we get $O(y, z) \rightarrow O(x, z)$. Together, these give $O(x, z) \leftrightarrow O(y, z)$, and since this holds for all z, we get $x = y$ by **OE**. Hence $P(x, y) \wedge P(y, x) \rightarrow x = y$, and this holds for all x, y, giving **PA**.

(c) For any non-empty set X, we have $X \cap X = X \neq \emptyset$. Hence if $I(x) = X$ we have $O(x, x)$, verifying **OR**.

Clearly $X \cap Y \neq \emptyset$ implies $Y \cap X \neq \emptyset$, and hence if $I(x) = X$ and $I(y) = y$, we have $O(x, y) \rightarrow O(y, x)$, verufying **OS**.

Now suppose $\forall z(O(x, z) \leftrightarrow O(y, z))$. Let $X = I(x)$ and $Y = I(y)$. Also, let $I(z) = \{a\}$. If $a \in X$ then $\{a\} \cap X \neq \emptyset$, so $O(x, z)$, hence $O(y, z)$, so $Y \cap \{a\} \neq \emptyset$, so $a \in Y$. Therefore $X \subseteq Y$. Similarly $Y \subseteq X$. Hence $X = Y$, and therefore we have $\forall z(O(x, z) \leftrightarrow O(y, z)) \rightarrow x = y$, verifying **OE**.

$P(x, y)$ means, if **PO** is to be true, that for any set Z, if $Z \cap X \neq \emptyset$, then $Z \cap Y \neq \emptyset$. In particular, for each $a \in X$ we have $\{a\} \cap X \neq \emptyset$, giving $\{a\} \cap Y \neq \emptyset$, so $a \in Y$. Hence $X \subseteq Y$. Thus $P(x, y)$ should be interpreted to mean $X \subseteq Y$.

(d) In the suggested interpretation we can have, for example,

$$X = \{0, 1\},\ Y = \{1, 2\},\ Z = \{2, 3\}.$$

Then $X \cap Y \neq \emptyset$, $Y \cap Z \neq \emptyset$, $X \cap Z = \emptyset$, so the formula

$$O(x, y) \wedge O(y, z) \wedge \neg O(x, z)$$

is satisfied, contradicting **OT**. Thus **OT** is not satisfied by every model for $\{$**OR**, **OS**, **SE**$\}$, and is therefore not a logical consequence of them.

4. (a) If the premisses are contradictory, they have no model, and therefore *any* interpretation satisfying them must satisfy the conclusion (since any counter-example to this statement would be a model for the premisses which falsifies the conclusion — and there are no models for the premisses).

Any inference with a logically true conclusion (i.e., satisfied in every model) is valid.

(b) Suppose $\Sigma \models C$ and $\Sigma \subseteq \Sigma'$. Suppose that interpretation I satisfies Σ'. This means that it satisfies every formula in Σ'. Since $\Sigma \subseteq \Sigma'$, these formulae include all the formulae in Σ, so I satisfies Σ. Since $\Sigma \models C$, this means that I satisfies C. We have now shown that any interpretation satisfying Σ' also satisfies C. Hence $\Sigma' \models C$. This means that $\models$ is monotonic.

(c) If we have $P_1, \ldots, P_n \models C$, then any interpretation satisfying $\{P_1, \ldots, P_n\}$ satisfies C, and therefore falsifies $\neg C$. This means that *no* interpretation satisfies $\{P_1, \ldots, P_n, \neg C\}$. It therefore follows, vacuously, that any interpretation satisfying $\{P_1, \ldots, P_n, \neg C\}$ also satisfies C (or indeed anything else), and so $P_1, \ldots, P_n, \neg C \models C$, in conformity with nonmonotonicity.

(d) There are many possible examples, e.g., no-one answers the doorbell so I "infer" that no-one is at home; then I learn that everyone in the house is fast asleep, and rescind the inference. [Discussion needed.]

5. (a) Important points are that 'algorithm' and 'heuristic' are *task-relative*, e.g., a heuristic for solving the TSP is in fact an algorithm for solving a related problem. The distinction between 'procedure' and 'algorithm is not always clear-cut, but I have suggested that the latter ought to imply termination (so algorithm = effective procedure).

   (b) A chance to display an understanding of the theory of NP-completeness, with at least some mastery of the details.