Foundations B-Exam
Spring 1995

1. Let I_1 and I_2 be propositional interpretations (i.e., truth assignments) on the same set AP of atomic propositions (i.e., sentence symbols). We say that $I_1 \leq I_2$ if $I_1(P) = \mathbf{true}$ implies $I_2(P) = \mathbf{true}$ for every $P \in AP$ (i.e., I_2 is "more true than or equal to" I_1).

   A Horn formula is a propositional formula in conjunctive normal form such that each conjunct contains at most one positive literal (a literal is an atomic proposition or a negated atomic proposition). Show that a Horn formula φ has a $\leq$-minimal satisfying interpretation. (Hint: Let φ be a Horn formula and let I_1 and I_2 be two interpretations that satisfy φ, i.e., $I_1 \models \varphi$ and $I_2 \models \varphi$. Let I be an interpretation such that $I(P) = \mathbf{true}$ iff both $I_1(P) = \mathbf{true}$ and $I_2(P) = \mathbf{true}$, for every atomic proposition P. Prove that $I \models \varphi$.)

2. Fix a vocabulary $\mathcal{L}$ that consists of a single binary predicate symbol E. A sentence φ is *finitely valid* if it holds in all finite structures. The *finite-validity problem* is to determine whether a given a sentence φ is finitely valid.

   Trakhtenbrot's Theorem says that the validity problem for sentences is undecidable. Show that Trakhtenbrot's Theorem implies that finite validity is not recursively enumerable. What can you deduce from this about the possibility of finding a sound and complete formal system for finite validity?

3. Let Σ_1 and Σ_2 be sets of sentence such that there is no structure $\mathcal{A}$ that is a model of both Σ_1 and Σ_2. Use the Compactness Theorem to show that there is sentence α such that $\Sigma_1 \models \alpha$ and $\Sigma_2 \models \neg\alpha$.

4. Let R be a regular language and let $w_1, w_2, \cdots, w_i, \cdots$ be an enumeration of all elements of R such that $|w_i| \leq |w_{i+1}|$ for all $i \geq 1$. Prove that there exists a constant c such that $|w_{i+1}| - |w_i| \leq c$ for all $i \geq 1$.

5. Let $L = \{w \in \{0,1\}^* \mid w = xx',\ |x| = |x'|, \text{ and } x' \neq \overline{x}\}$, where $\overline{x}$ is the string obtained from x be reversing all bits. For instance, if $x = 01000$ then $\overline{x} = 10111$. Prove that L is a context-free language.

6. Which of the following problems about Turing machines are decidable, and which are undecidable? Justify your answers carefully.

   (a) To determine, given a Turing machine M, whether there exists a string w such that M halts on input w.

   (b) To determine, given a Turing machine M and an alphabet symbol a, whether M ever writes a on its tape when started on an empty tape.

Solutions

1. Let φ be a Horn formula and let I_1 and I_2 be two interpretations that satisfy φ, i.e., $I_1 \models \varphi$ and $I_2 \models \varphi$. Let I be an interpretation, denoted $I_1 \sqcap I_2$, such that $I(P) = \mathbf{true}$ iff both $I_1(P) = \mathbf{true}$ and $I_2(P) = \mathbf{true}$, for every atomic proposition P. Note that $I \leq I_1$ and $I \leq I_2$.

   Claim: $I \models \varphi$.

   Proof: Consider a conjunct α of φ. There are 3 cases to consider:

   (a) α is a positive literal P. In this case, we have that $I_1(P) = I_2(P) = \mathbf{true}$, so $I(P) = \mathbf{true}$, and $I \models \alpha$.

   (b) α is a disjunction $\neg P_1 \vee \cdots \vee \neg P_k$ of negative literals, $k \geq 1$. Since $I_1 \models \alpha$, we must have $I_1(P_j) = \mathbf{false}$ for some $1 \leq j \leq k$. Thus, $I(P_j) = \mathbf{false}$, and $I \models \alpha$.

   (c) α is a disjunction $\neg P_1 \vee \cdots \vee \neg P_k \vee Q$ with precisely one positive literal, $k \geq 1$. Since $I_1 \models \alpha$ and $I_2 \models \alpha$, we either have that $I_1(Q) = I_2(Q) = \mathbf{true}$ or $I_1(P_j) = \mathbf{false}$ for some $1 \leq j \leq k$. In the former case we have that $I(Q) = \mathbf{true}$ and in the latter case we have that $I(P_j) = \mathbf{false}$. In either case $I \models \alpha$.

   Take $M = \{I : I \models \varphi\}$ to be the set of interpretations of φ and let $J = \sqcap M$. By the claim above we have that $J \models \varphi$ and by the observation above J is $\leq$-minimal.

2. Let us say that a sentence φ is *finitely satisfiable* if it holds in some finite structures. The *finite-satisfiability problem* is to determine whether a given sentence φ is finitely satisfiable.

   The finite-satisfiability problem is recursively enumerable. To find if a given sentence is finitely satisfiable enumerate all finite structures and check whether they satisfy the given sentence.

   A sentence φ is finitely valid iff $\neg\varphi$ is not finitely satisfiable. If finite validity were recursively enumerable, then it would be decidable, contradicting Trakhtenbrot's Theorem. To check whether φ is finitely valid or not, we will enumerate in parallel all finitely valid sentences and all finitely satisfiable sentence. Ultimately either φ will show up in the enumeration of finitely valid sentences or $\neg\varphi$ will show up in the enumeration of finitely satisfiable sentences.

   If there is a sound and complete formal system for finite validity, then a sentence φ is finitely valid precisely when it is provable in the system. But then finite validity is recursively enumerable, since one can simply enumerate all proofs and see whether any of them constitute a proof of φ.

3. By assumption $\Sigma_1 \cup \Sigma_2$ is unsatisfiable. By the Compactness Theorem, there are finite subsets $\Sigma_1' \subseteq \Sigma_1$ and $\Sigma_2' \subseteq \Sigma_2$, such $\Sigma_1' \cup \Sigma_2'$ is unsatisfiable. Clearly, $\Sigma_1' \cup \Sigma_2$

is also unsatisfiable. Let α be $\bigwedge \Sigma_1'$. Since $\Sigma_1' \subseteq \Sigma_1$, clearly $\Sigma_1 \models \alpha$. Also, since $\Sigma_1' \cup \Sigma_2$ is unsatisfiable, $\Sigma_2 \cup \{\alpha\}$ is also unsatisfiable, and consequently $\Sigma_2 \models \neg\alpha$.

4. We will use the Pumping Lemma. Let F be a finite automaton that accepts R and let n be the number of states of F. If no string in L has length greater than n, then we can choose $c = n$. Otherwise let x be any string in R of length greater than n. According to the Pumping Lemma there exist strings u, v, and w such that $x = uvw$, $1 \leq |v| \leq n$ and $uv^k w$ is in R for any $k \geq 0$. Thus we can choose in this case $c = |v|$ (or $c = n$ since $|v| \leq n$).

5. We will construct a non-deterministic pushdown automaton P that accepts L. Note that L consists of all strings of even length that are not of the form $x \mid x$.

   Let w be any string of length $2n$. To check whether $w \in L$ P will, essentially, try to "guess" a pair of symbols α and β from w such that (i) if k and l are the positions of α and β in w then $l = n + k$, and (ii) $\alpha = \beta$. This will imply that w is not of the form xx'.

   On input w, P reads the symbols of w and pushes them onto the stack for some (non-deterministically chosen) number of steps. Then P stores the next, say k-th, symbol of w in its finite memory. Next P reads k more symbols of w popping at each step an element from its stack until the stack is emptied. Once again, P pushes a number of symbols of w (say $l - 1$) onto its stack and compares the l-th symbol with the one stored in its finite memory. If the two symbols are different then P has no next move and dies. If the symbols are the same, then P checks if there are exactly l more symbols left in w by popping off symbols from its stack. If there are more or less than l symbols left, then P has no next move. If the remaining symbols are exactly l, then P accepts w.

   It is straightforward to check that, if P accepts w, then w has $2(k + l)$ symbols and that the two symbols that P has compared are on positions k and $2k + l = n + k$.

6. (a) **Undecidable.** We will show this by a reduction from the halting problem.

   Assume that the problem were decidable and let A be an algorithm that determines whether a given arbitrary TM halts on any input. Let M be an arbitrary TM and w be any string over the alphabet of M. Then we can check if M halts on input w by the following algorithm. We construct a TM M' that ignores its own input and simulates M on w. By definition, M' halts on any input, iff M halts on w, and M' halts on no input iff M does not halt on w. Then we can use A to determine whether M' halts on any input: if yes, then M halts on w, if not, then M does not halt on w. This is a contradiction since the halting problem is undecidable.

   (b) **Undecidable.** We will prove this by a reduction from the membership problem. (We can use the halting problem as well.)

Assume that the problem were decidable and let B be an algorithm that determines whether a given arbitrary TM ever writes a specified alphabet symbol on its tape. Then, given any string w, we can check whether M accepts w as follows. Define a new TM M' by extending M's alphabet by a new symbol z and changing M's moves so that each time M accepts and halts, M' first writes z and then accepts and halts. Then we can determine if M accepts w by checking (using B) if M' writes z on its tape.