Take-home test 3 answers

1. Somehow need to represent the start state, the accept state, the reject state, and the transitions. Details of the states, input alphabet and tape alphabet could be left implicit. (This answer is incomplete but the question was discussed in tutorials.)

2. Easiest to use a multitape Turing machine: one tape for the simulated machine’s description, mainly its transitions; one tape for the simulated machine’s tape, with head position marked; and one tape for the simulated machine’s state. Each step of the simulated machine requires the Universal machine to scan the transitions for one matching the current state and tape symbol, then perform the transition. (This answer is incomplete but the question was discussed in tutorials.)

3. First, here’s a solution to a simpler problem: the complement of the language generated by the following context-free grammar:

   \[ A \rightarrow a_1 c_1 | \ldots | a_n c_n | a_1 A c_1 | \ldots | a_n A c_n \]

   (We use the shorthand \( a \rightarrow \gamma \) to denote \( a, \varepsilon \rightarrow \gamma \). We also use shorthands \( a_i, c_i, a_i \) and \( c_i, a_j \) in what we hope will be obvious ways.)

   \[ \varepsilon \rightarrow \$ \]

   \[ a_i \rightarrow a_i \]

   \[ c_i, a_i \rightarrow \varepsilon \]

   \[ \varepsilon, \$ \rightarrow \varepsilon \]

   \[ c_i \rightarrow \varepsilon \]

   \[ a_i \rightarrow \varepsilon \]

   \[ c_i \rightarrow \varepsilon \]

   \[ c_i \rightarrow \varepsilon \]

   \[ a_i \rightarrow \varepsilon \]

   This solution then needs to be extended by replacing the single symbols \( a_i \) by the (nonempty) symbol sequences \( \alpha_i \) and taking the end-of-input symbol into account.

4. (a) No solution, by straightforward case analysis.

   (b) The sequence 1-4-2 is a solution.

5. If each element of any pair has the same length, the sequence consisting of just that pair is a solution.

   If all pairs have the first element shorter than the second element, there is clearly no solution. Similarly if all pairs have the first element longer than the second element.

   So, suppose the \( i \)'th pair \( (a_i, b_i) \) has \( a_i < b_i \) and the \( j \)'th pair \( (a_j, b_j) \) has \( a_j > b_j \). Let \( m_i = b_i - a_i \) and \( m_j = a_j - b_j \). Then the PCP instance has a solution consisting of \( m_j \)
occurrences of pair $i$ and $m_i$ occurrences of pair $j$, since

$$m_j a_i + m_i a_j = (a_j - b_j)a_i + (b_i - a_i)a_j$$

$$= a_j a_i - b_j a_i + b_i a_j - a_i a_j$$

$$= b_i a_j - b_j a_i$$

$$= (a_j - b_j)b_i + (b_i - a_i)b_j$$

$$= m_j b_i + m_i b_j$$

6. (a) First, by Rice’s Theorem, it is undecidable whether a Turing machine recognises the empty language. Now, given a Turing machine encoding $\langle M \rangle$, transform $M$ so that its alphabet omits 1, then transform it again so that immediately before accepting it writes 1 (and then accepts). Call this machine $M'$. Now $L(M) \neq \emptyset$ if and only if $M'$ writes 1 on its tape. But $L(M) = \emptyset$ is undecidable and hence so is the required property.

(b) This Turing machine property is semantic (it’s a property of the language) and nontrivial (it’s true of some Turing machines and not others), so by Rice’s Theorem the property is undecidable.

7. (a) It is straightforward to construct a Turing machine $S$ that simulates $M$ for $n$ steps on each of the first $n$ strings in $M$’s input alphabet, for increasing $n$, and accepts if and only if $M$ accepts its second string. Hence, the given property is Turing-recognisable.

(b) This property appears to be not Turing-recognisable, because there is no obvious Turing machine that recognises (Turing machines with) the property. However, I am still looking for a proof of this claim.

Rodney Topor