Decidability and Computability: Problems for Week 9

Exercise 1 The following program uses recursion to compute a binary function B on natural numbers.

```
B(0,n)  = n+7
B(1,n)  = n+5
B(m+2,0)  = m+17
B(m+2,1)  = m+B(m+1,9)
B(m+2,n+2)  = B(m,n+7) + B(m+2,n)
```

Show that it terminates for all m and n.

Solution For $m, n \in \mathbb{N}$, let $Q(m, n)$ be the statement that the evaluation of $B(m, n)$ terminates. We prove $\forall m \in \mathbb{N}. \forall n \in \mathbb{N}. Q(m, n)$ by course-of-values induction.

- To treat the case $m = 0$, we obtain $\forall n \in \mathbb{N}. Q(0, n)$ since $B(0, n)$ returns $n + 7$.
- To treat the case $m = 1$, we obtain $\forall n \in \mathbb{N}. Q(1, n)$ since $B(0, n)$ returns $n + 5$.
- To treat the case $m = m' + 2$, we prove $\forall n \in \mathbb{N}. Q(m' + 2, n)$ by course-of-values induction on n.
  - To treat the case $n = 0$, we obtain $Q(m' + 2, 0)$ since $B(m' + 2, 0)$ returns $m' + 17$.
  - To treat the case $n = 1$, we obtain $Q(m'+2, 1)$ since $B(m'+1, 9)$ returns a value p (by the outer inductive hypothesis applied to $m' + 1 < m$), and so $B(m' + 2, 1)$ returns $m' + p$.
  - To treat the case $n = n' + 2$, we obtain $Q(m' + 2, n' + 2)$ since $B(m', n' + 7)$ returns a value p (by the outer inductive hypothesis applied to $m' < m$) and $B(m' + 2, n')$ returns a value q (by the inner inductive hypothesis applied to $n' < n$), and so $B(m' + 2, n' + 2)$ returns $p + q$.

Exercise 2 Write `nat k = max(j-i,0)` in Primitive Java. You may use all the encodings listed in the handout.
Solution

```
nat k = j;
repeat i times {k--;}
```

Exercise 3
Here is a unary program in Basic Java (using the encodings given in the handout).

```
nat i = 0;
nat j = 0;
while i != input0 {
  i++;
  i++;
  j++;
}
output = j;
```

What partial function from $\mathbb{N}$ to $\mathbb{N}$ does it compute? (Your answer should be 1–2 lines long.) **Solution** The one that halves every even number, and is undefined on odd numbers.

Exercise 4 Complete the following sentences. Let's say that the alphabet Σ is $\{\mathtt{a}, \mathtt{b}\}$.

- A function $f : \mathbb{N} \to \mathbb{N}$ is computable when ... If it is not computable, then by Church's thesis ...
- A subset $A \subseteq \mathbb{N}$ is decidable when ... If it is not decidable, then by Church's thesis ...
- A language $A \subseteq \Sigma^*$ is decidable when ... If it is not decidable, then by Church's thesis ...

- Ambiguity of a context free grammar over Σ is an undecidable property. This means ... By Church's thesis, this implies ...

Solution

- A function $f: \mathbb{N} \to \mathbb{N}$ is computable when there is a Turing machine that, when executed on a tape containing just a number n written in binary with the head on the leftmost character, terminates when the tape contains just $f(n)$ written in binary. If it is not computable, then by Church's thesis there is no algorithm that takes a number n and returns $f(n)$.

- A subset $A \subseteq \mathbb{N}$ is decidable when there is a Turing machine that, when executed on a tape containing just a number n written in binary with the head on the leftmost character, terminates by returning True if $n \in A$ and False otherwise. If it is not decidable, then by Church's thesis there is no algorithm that takes a number n and returns True if $n \in A$ and False otherwise.

- A language $A \subseteq \Sigma^*$ is decidable when there is a Turing machine that, when executed on a tape containing just a word w with the head on the leftmost character, terminates by returning True if $w \in A$ and False otherwise. If it is not decidable, then by Church's thesis there is no algorithm that takes a word w and returns True if $w \in A$ and False otherwise.

- Ambiguity of a context-free grammar over Σ is an undecidable property. This means that there is no Turing machine that, when executed on a tape containing just context-free grammar L encoded as a word, terminates by returning True if L is ambiguous and False otherwise. By Church's thesis, this implies that there is no algorithm that takes a context-free grammar L and returns True if L is ambiguous and False otherwise.

Exercise 5 Is ambiguity of a context free grammar a semidecidable property? What about non-ambiguity? Explain your answers. You may use facts that we have seen previously.

Solution Ambiguity is semidecidable. Firstly, any triple (w, D, D') consisting of a word w and two distinct leftmost derivations can be encoded as a string. Here is a program that semidecides ambiguity. Given a grammar, go through all strings in order until you find one that encodes such a triple, then return True. Thus True is returned if there is such a triple (i.e. the grammar is ambiguous), and if not, then the program runs forever.

Non-ambiguity is not semidecidable. For if it were semidecidable, then ambiguity would be decidable, contradicting what was stated in the previous exercise.