Computation Theory

Supervision 1

1. Algorithmically undecidable problems

1. Two important concepts in the theory of computability are *enumerations* and *diagonalisation*. Intuitively, an enumeration of a set S is an ordered, "exhaustive" listing of all elements. While this intuition works for finite sets, we need to be more formal to handle infinite sets. Thus, an *enumeration* of a finite or infinite set S is a surjective function from the natural numbers $\mathbb{N}$ to S, if it exists. If it does, the set S is called *countable*; if it doesn't, it is *uncountable*.

   Prove or disprove the following statements:

   a) The set of natural numbers is countable.

   b) The set of integers is countable.

   c) The Cartesian product of two countable sets is countable.

   d) The set of rational numbers is countable.

   e) The finite n-ary product of countable sets is countable.

   f) The set of polynomials with coefficients from a countable set is countable.

   g) The powerset of a countable set is countable.

   h) The set of real numbers is countable.

   Feel free to do some research if you are not familiar with these results.

2. Rephrase the proof of the undecidability of the Halting Problem (with an abstract definition of an algorithm) as a diagonal argument.

2. Register machines

1. Show that the following arithmetic functions are all register machine computable.

   a) First projection function $p \in \mathbb{N} \to \mathbb{N}$, where $p(x, y) \triangleq x$

   b) Constant function with value $n \in \mathbb{N}$, $c_n \in \mathbb{N} \to \mathbb{N}$ where $c(x) \triangleq n$

   c) Truncated subtraction function, $\_ \dot{-} \_ \in \mathbb{N}^2 \to \mathbb{N}$, where

   $$x \dot{-} y \triangleq \begin{cases} x - y & \text{if } y \leq x \\ 0 & \text{if } y > x \end{cases}$$

   d) Integer division function, $\_\text{div}\_ \in \mathbb{N}^2 \to \mathbb{N}$ where

   $$x \text{ div } y \triangleq \begin{cases} \text{integer part of } x/y & \text{if } y > 0 \\ 0 & \text{if } y = 0 \end{cases}$$

e) Integer remainder function, $\_\mathrm{mod}\_ \in \mathbb{N}^2 \to \mathbb{N}$ with $x \bmod y \triangleq x \dot{-} y \cdot (x \operatorname{div} y)$

f) Exponentiation base 2, $e \in \mathbb{N} \to \mathbb{N}$, where $e(x) = 2^x$

g) Logarithm base 2, $\log_2 \in \mathbb{N} \to \mathbb{N}$, where

$$\log_2(x) \triangleq \begin{cases} \text{greatest } y \text{ such that } 2^y \leq x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

Hint: instead of defining everything from scratch, try implementing these machines with the help of general control flow components.

3. Coding programs as numbers

1. *Gödel numbering* is a general technique for assigning a natural number to some mathematical object (such as a well-formed formula in some formal language). The numbering is often computed by translating every symbol of a formula Φ to a natural number, then combining the codes to create a unique Gödel number $G(\Phi)$. For example, with the assignments $t(\text{`}\forall\text{'}) = 1, t(\text{`}x\text{'}) = 2, t(\text{`.'}) = 3$, and $t(\text{`}=\text{'}) = 4$, the Gödel number of the formula $\forall x.\ x = x$ with a particular combination function could be $G(\text{`}\forall x.\ x = x\text{'}) = 272794772250$.

   a) Is the Gödel numbering of register machines described in the notes a bijection, an injection, a surjection, a total function, a partial function, or a relation? Justify your answer.

   b) In the example of first-order logic above, is a particular Gödel numbering a bijection, an injection, a surjection, a total function, a partial function, or a relation? Justify your answer.

   c) Suggest one or more ways of combining the symbol codes of a formula Φ to generate a *unique* Gödel number for Φ. Demonstrate your methods on the formula $\Phi = \text{`}\forall x.\ x = x\text{'}$ used above.

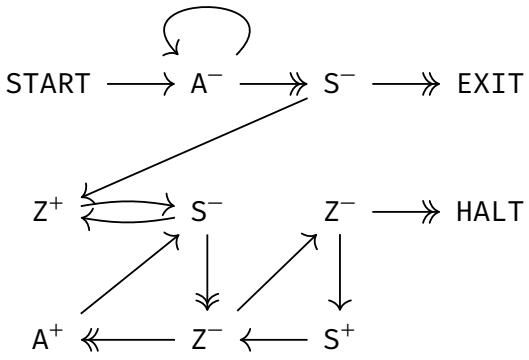
2. Let $\varphi_e \in \mathbb{N} \rightharpoonup \mathbb{N}$ denote the unary partial function from numbers to numbers computed by the register machine with code e. Show that for any given register machine computable unary partial function $f \in \mathbb{N} \rightharpoonup \mathbb{N}$, there are infinitely many numbers e such that $\varphi_e = f$. Two partial functions are equal if they are equal as sets of ordered pairs; equivalently, for all numbers $x \in \mathbb{N}$, $\varphi_e(n)$ is defined if and only if $f(x)$ is, and in that case they are equal numbers.

4. Universal register machine

1. What is the aim of the universal register machine U? How does it work? Annotate the diagram of the register machine with its major components, explaining what they accomplish in the bigger context of the operation of U.

2. Consider the list of register machine instructions whose graphical representation is shown below. Assuming that register Z holds 0 initially, describe what happens when the code is executed (both in terms of the effect on registers A and S and whether the code halts by

jumping to the label EXIT or HALT).



Optional exercise

Write a register machine interpreter in a programming language you prefer (a functional language such as ML or Haskell is recommended). Implement a library of RM building blocks such as the ones appearing in the universal register machine or your answer for Ex. 2.1. You may try implementing the RM U as well, but don't worry if you run into resource constraints. The format of input and output is up to you but the RM representation and computation must conform to the theoretical definition.