PHIL3110 - Assignment 2

Maxwell Bo

June 18, 2018

Problem 1

Definition 1 *$f : \mathcal{X} \rightarrowtail \mathcal{Y}$ is injective $\Leftrightarrow \forall x_1, x_2 \in \mathcal{X}$ if $F(x_1) = F(x_2)$ then $x_1 = x_2$.*

Theorem 1 *If $f : \mathcal{X} \rightarrowtail \mathcal{Y}$ is injective and $g : \mathcal{Y} \rightarrowtail \mathcal{Z}$ is injective, then $g \circ f$ is injective.*

Proof 1 *Suppose $f : \mathcal{X} \rightarrowtail \mathcal{Y}$ is injective and $g : \mathcal{Y} \rightarrowtail \mathcal{Z}$ is injective. We must show that $g \circ f$ is injective. Suppose x_1 and x_2 are elements of $\mathcal{X}$ such that*

$$(g \circ f)(x_1) = (g \circ f)(x_2)$$

By definition of composition of functions,

$$g(f(x_1)) = g(f(x_2))$$

Since g is injective

$$f(x_1) = f(x_2)$$

And since f is injective

$$x_1 = x_2$$

Theorem 2 *There is some injection $f : A \rightarrowtail B \Leftrightarrow A \precsim B$.*

If A, B and C are sets such that $A \precsim B$ and $B \precsim C$, there exists an injective $f : A \rightarrowtail B$ and injective $g : B \rightarrowtail C$. Per theorem 1, $g \circ f : A \rightarrowtail C$ is injective. Per thereom 2, $g \circ f$ entails $A \precsim C$.

Problem 2

1. $$\varphi_1 := (\exists x)(x)$$

   says that there is at least 1 object, and

   $$\varphi_2 := (\exists x)(\exists y)(x \neq y)$$

   says that there at least 2 objects.

   Thus

   $$\varphi_n := (\exists x_1)(\exists x_2) \ldots (\exists x_n)(x_1 \neq x_2 \neq \ldots \neq x_n)$$

   says that for any natural number n, there are at least n objects.

2. We need to prove that every finite subset Δ of S has a model, e.g. there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \Delta$.

   First, we'll define a stage based defintion of S, where:

$$s(0) = T$$
$$s(n+1) = s(n) \cap \{\varphi_{n+1}\}$$

such that $\{s(n) \mid n \in \omega\} = S$.

At each step, we need to show that

- The model that satisfies $s(n)$ defines $s(n)$ objects
- The model is finitely satisfiable

Base

$0 \in A$. We can fix some $\mathcal{M}$ where $\mathcal{T} \subseteq \mathcal{M}$, where $\mathcal{T} \models T$ for every finite subset Δ of $s(0)$.

The only finite subset of $s(0)$ is T.

By our fix of $\mathcal{M}$, $\mathcal{M} \models T$. Therefore, $s(0)$ is finitely satisfiable - $\mathcal{M} \models s(0)$.

$\mathcal{M}$ defines at least 0 objects.

Induction Step

Suppose $n \in A$.

Let $\mathcal{N}$ be the model that satisfied $s(n)$. It defined at least n objects.

$\mathcal{N} \not\models \Delta$ for each finite subset Δ of $s(n+1)$.

Why?

The only finite subset Δ of $s(n+1) \setminus s(n)$ is $\{\varphi_{n+1}\}$.

Consider the only sentence in Δ, φ_{n+1}.

In order for $\mathcal{N} \models \varphi_{n+1}$, $\mathcal{N}$ would need to define $n+1$ objects.

Consider some $\mathcal{M}$ that defines a new object, so that $\mathcal{N} \subseteq \mathcal{M}$. Now $\mathcal{M}$ defines at least $n+1$ objects.

By theorem 139, as γ is of the form Σ_1, and $\mathcal{N} \models {\varphi_n}$[1], $M \models \varphi_{n+1}$.

As $\mathcal{M} \models s(n)$, and $\mathcal{M} \models s(n+1) \setminus s(n)$, every finite subset Δ of $s(n) \cup (s(n+1) \setminus s(n))$ is satisfiable.

Thus, every finite subset Δ of $s(n+1)$ is satisfiable.

Thus, $n+1 \in A$. Then by induction, we see that every $n \in A$.

S is finitely satisfiable.

3. As every finite subset Δ of S has a model, by the compactness theorem, there is a model $\mathcal{M}$ such that $\mathcal{M} \models S$. By the previous step, we know that $\mathcal{M}$ defines ω objects to satisfy S. ω is infinite. Therefore, $\mathcal{M}$ is infinite, and we are thus able to say that S has, or requires, an infinite model.

[1] $\varphi_n \in s(n) \setminus s(n-1)$

Problem 3

```
; This program accepts a block of n-many 1s and outputs a block of 2n-many 1s,
; after the original block with a single blank space separating them;

; Henceforth:
; - the n-many 1's will be referred to as the "parameter array"
; - the 2n-many 1's will be referred to as the "accumulator array"
; - the single blank space seperating them will be referred to as "the divider"

; ALGORITHM SUMMARY: Move a loop pointer through the parameter array,
; terminating the loop when the pointer reaches the end of the parameter array.
; On each loop, append two 1s to the end of the accumulator array.

; ### State 0: as per the assignment sheet, the head should start under the 1th cell
0 -  -R 1

; ### State 1 deals with placing our loop pointer, and halting the loop
; Our loop pointer is a blank, that shifts through our parameter array.
; where [_ 1 1 1] starts the loop, and [1 1 1 1] halts the loop

; This instruction puts down the new loop pointer.
; This either initializes it (if we came from State 0),
; or increments it (if we came from State 6)
1 1 -R 2

; ### State 2 deals with getting to the start of the accumulator array
; Glide over the parameter array
2 1 1 R 2

; Jump over the divider
2 --R 3

; ### State 4 and 5 deal with getting to the end of the accumulator,
; and appending two 1s.
; Glide over the accumulator...
3 1 1 R 3

; ... until we pop out the end of the accumulator. Put down a 1...
3 -1 R 4

; ... and another one. Now we've gotta turn back around and increment the loop variable.
4 -1 L 5

; ### State 5 deals with trying to get back to the end of the parameter array
; Glide back over the accumulator array
5 1 1 L 5
; Jump over the divider
5 --L 6

; ### State 6 deals with incrementing our loop variable
; Glide over our parameter array...
6 1 1 L 6

; ...until we hit our loop pointer. We clear the current loop pointer, shift the
; head right, and loop back to state 1, so that it may deal with the next loop iteration.
```

```
6 -1 R 1

; This is our loop halting condition. State 6 has cleared the loop pointer,
; and pushed the head onto the divider. We still have to move the head to the 1th cell.
1 --L 7

; #### State 7 deals with getting to the 1th cell, and halting
; Glide back over our parameter array...
7 1  1 L 7

; ... and when we pop out past the head of the parameter array, shift right to
; the 1th cell, and halt
7 --R halt
```

Problem 4

We will attempt to demonstrate that A is recursive. The same process may be repeated to show that B and C are also recurisve.

By definition, A is recursively enumerable, and that $A \subseteq \omega$. To prove that A is recurisive, we must show that $\omega \setminus A$ is recursively enumerable.

Suppose $\omega \setminus A$ was not recursively enumerable.

There exists an $n \in \omega \setminus A$, such that no φ_e exists, that has a domain $\omega \setminus A$, and halts when applied to n.

As B and C are recursively enumerable there are partial recursive functions φ_B and φ_C which have the domains of B and C respectively. We can define a function

$$\varphi_{B\cup C}(n) = \begin{cases} \varphi_b(n), & \text{if } \varphi_b(n) \text{ halts.} \\ \varphi_c(n), & \text{if } \varphi_c(n) \text{ halts.} \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

which has the domain $B \cup C$. By definition $A \cap B \cap C = \omega$, $B \cap C = \omega \setminus A$.

This means we have found a partially recursive function, $\varphi_{B\cup C}$, that has the domain $n \in \omega \setminus A$ and halts on every $n \in \omega \setminus A$. This is a contradiction.

Thus, $\omega \setminus A$ is recursively enumerable.

Thus, A is recursive.