CS3133 Foundations of Computer Science HW5

Keith DeSantis

10/11/21

Symbols for Convenience: $\emptyset \neq \Sigma \lambda \subseteq \not\subset \epsilon \delta \equiv \not\equiv \notin \exists$
Collaboration with Kush Shah and Samuel Parks

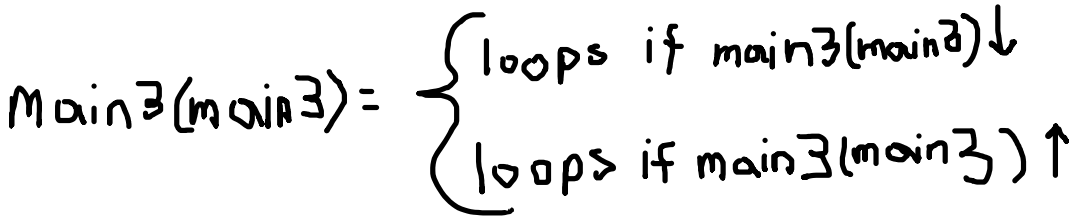
NOTE: I use "L'" to represent L's complement because I don't know how to put the little bar thingy. The only place I don't use L' to mean that is in the last problem, SDSplitting.

258 HaltVariation 3)

```
int main3 (p) {
        if (selfHaltTest(p))
                while (1) do ;
        else
                while (1) do ;
}
```

This will not support the proof by contradiction.

This is because in this new version of the program:



Here, only the upper option of "loops if main3(main3) halts" is a contradiction, while the lower option of "loops if main3(main3) doesn't halt is a perfectly valid statement. Because of this the overall statement cannot be labeled a contradiction, while in the original proof it could because both statements were contradictions ("loops if main3(main3) halts" and "halts if main3(main3) doesn't halt").

275 ApplyRice2)

a) { p | p has even length }

L is not a functional set of programs. This is because two programs, p1 and p2, could compute the same function but p1 could have even length while p2 has an odd length.

L is decidable since a program to determine a program's membership would be counting the size of the program and checking evenness.

b) { p | the domain of pfn(p) has at least 10 strings }

L is a functional set of programs because for any p and q that compute the same function, $p \in L$ iff $q \in L$.

This is because given $p \in L$, that means pfn(p) has at least 10 strings in its domain. Since pfn(p) = pfn(q) this means pfn(q) has at least 10 strings in its domain, therefore $q \in L$.

The same logic can be applied to show that if $q \in L$ then $p \in L$.

L is undecidable by Rice's theorem.

c) { p | the domain of pfn(p) $\neq \emptyset$ }

L is a functional set of programs. Given p and q that compute the same function, $p \in L$ iff $q \in L$ since if pfn(p) $\neq \emptyset$ then pfn(q) $\neq \emptyset$.

L is undecidable by Rice's Theorem.

d) { p | pfn(p) is the identity function }

L is a functional set of programs since for any p and q that compute the same function, $p \in L$ iff $q \in L$. This is because if pfn(p) is the identity function, then pfn(q) is the identity function and therefore $q \in L$.

L is undecidable by Rice's Theorem.

e) { p | pfn(p) is a constant function }

L is a functional set of programs since for any p and q that compute the same function, $p \in L$ iff $q \in L$. This is because if pfn(p) is a constant function, then pfn(q) is a constant function and therefore $q \in L$.

L is undecidable by Rice's Theorem.

f) { p | pfn(p) = pfn(q) } where q is a fixed program that returns 0 on odd length strings and loops forever on even length strings.

L is a functional set of programs since for any p and k that compute the same function, p ϵ L iff k ϵ L. This is because if pfn(p) = pfn(q), then pfn(k) = pfn(q) and therefore k ϵ L.

L is undecidable by Rice's Theorem.

g) { p | the domain of pfn(p) = ∅ }

L is a functional set of programs since for any p and q that compute the same function, p ϵ L iff q ϵ L. This is because if the domain of pfn(p) = ∅ then the domain of pfn(q) = ∅ and therefore q ϵ L.

L is undecidable by Rice's Theorem.

h) { p | the domain of pfn(p) is finite }

L is a functional set of programs since for any p and q that compute the same function, p ϵ L iff q ϵ L. This is because if the domain of pfn(p) is finite then the domain of pfn(q) is finite and therefore q ϵ L.

L is undecidable by Rice's Theorem.

i) { p | there is some shorter program q computing the same function as p }

L is not a functional set of programs. Given two programs, m and k that compute the same function, let m be the shortest possible program that computes said function. k ϵ L because m is a shorter program that computes the same function, however, m ∉ L since there is not program shorter than it that computes the same function.

L is undecidable.

j) { p | the domain of pfn(p) is decidable }

L is a functional set of programs since for any p and q that compute the same function, p ϵ L iff q ϵ L. This is because if pfn(p) has a decidable domain then pfn(q) must have a decidable domain (since they have the same domain) and therefore q ϵ L.

L is undecidable by Rice's Theorem.

280 DecClosure)

D = decidable language
N = undecidable language

1. X = D n N: X is neither necessarily decidable nor undecidable.

   Case where X is decidable:
   X = ∅
   N = odd-lengthed programs that self halt
   D = even-lengthed programs

   Here, N n D will return ∅ (which is decidable) since N ⊆ the complement of D.

   Case where X is undecidable:
   X = odd-lengthed programs that self halt
   N = odd-lengthed programs that self halt
   D = odd-lengthed programs

   Here, N n D will return N since N ⊆ D.

2. N = D n X: X is necessarily undecidable.

   This is because the intersection of two decidable languages is decidable (DnD = D). Since the intersection of D and X is not decidable, X cannot be decidable.

3. D = N n X: X is neither necessarily decidable nor undecidable.

   Case where X is decidable:
   D = ∅
   N = odd-lengthed programs that self halt
   X = even-lengthed programs

   Here, N n X will return ∅ since part of N ⊆ of the complement of X.

   Case where X is undecidable:
   D = ∅
   N = odd-lengthed programs that self halt
   X = even-lengthed programs that self halt

   Here, N n X will return ∅ since part of N ⊆ of the complement of X.

4. D = X': X is necessarily decidable.

   This is because the complement of a decidable language is decidable, so if D = X' then X = D', and D is decidable.

5. N = X': X is necessarily undecidable

If X were decidable then its complement N would be decidable.

288Taxonomy)

For this question, L' will be used to denote the complement of L.

1. L is decidable and L' is decidable

   L = Even-length strings
   L' = Odd-length strings

2. L is decidable and L' is semi-decidable but not decidable

   This cannot happen because the complement of a decidable language is decidable.

3. L is decidable L' is not semi-decidable

   This cannot happen because the complement of a decidable language is decidable and by definition a decidable language is semi-decidable.

4. L is semi-decidable but not decidable and L' is decidable

   This cannot happen because the complement of a decidable language is decidable. Given that L' is decidable, then its complement must be decidable.
   The complement of L' is L, so L must be decidable and we have reached a contradiction.

5. L is semi-decidable but not decidable and L' is semi-decidable but not decidable

   This cannot happen because of the theorem: "L is decidable iff L is semidecidable and L' is semidecidable."

   If the statement from this problem were true, then L and L' would both be semidecidable, meaning L would be decidable. But the problem statement claims L is not decidable, and there is the contradiction.

6. L is semi-decidable but not decidable and L' is not semi-decidable

   L is the set of programs that halt on themselves, and L' is the set of programs that don't halt on themselves.

   We know SelfHalt is semi-decidable but not decidable, so it's complement must be non-semi-decidable due to "L is decidable iff L is semidecidable and L' is semidecidable."

7. L is not semi-decidable and L' is decidable

   This cannot happen because the complement of a decidable language is decidable.
   Given that L' is decidable, then its complement must be decidable.
   The complement of L' is L, so L must be decidable.
   By definition, if a language is decidable it is also semi-decidable, so L must be semi-decidable and we've reached a contradiction.

8. L is not semi-decidable and L' is semi-decidable but not decidable

   L is the set of all programs that don't halt on themselves
   L' is the set of all programs that halt on themselves.

290 SDSplitting)

L is semi-decidable but not decidable. (So L-comp is not semi-decidable).
Consider the language

$L'_{def} = \{ 0x \mid x \text{ is in } L \} \cup \{ 1x \mid x \text{ is not in } L \}$

Can you say for certain whether L' is decidable, semi-decidable, or non-semi-decidable?

L' is non-semi-decidable.

This is because:

L is semi-decidable, meaning we have a program that will always answer 1 if an input is in L. By slightly modifying that program to take in only strings that start with 0 and then running on the rest of the string (the x part of 0x), we can tell if something is in the { 0x | x is in L } part of L'.

However, since L-complement is non-semi-decidable, that means that there is no program that will always answer 1 if an input is in L-complement. Since the { 1x | x is not in L } section of the definition of L' relies directly on L-complement, this means there is no program that will always answer 1 if an input is in { 1x | x is not in L }.

Since there doesn't exist a program to determine affirmative membership for part of L', L' cannot be semi-decidable.