

Solutions for Problem Set 6
cs172

Allison Coates

1. Write out the program SELF in pseudo pascal: Take care to get the quotes printed correctly.

We reduced the requirement that the program be written in pseudo pascal, since people seemed to find it easier to write the program in some actual language. This makes details really hard, instead of easier, but perhaps it is less confusing if you are a good programmer (unlike me.) But you weren't supposed to write unreadable programs! That was the point of the pseudocode. Here's a reasonable idea, where I take only one liberty with how my code interprets quotes: it prints all but the outer matching pair.

```
A = "print"A = "" ; printA; print"" ; printA;";  
print"A = "";  
printA;  
print"" ;  
printA;  
-----
```

For ease, I reprint the above with the lines numbered:

```
A = "print"A = "" ; printA; print"" ; printA;"; (1)  
print"A = ""; (2)  
printA; (3)  
print"" ; (4)  
printA; (5)  
-----
```

Below, I include the line numbers to show the result of each line:

```
(1)  
A = "  
(2)  
print"A = "" ; printA; print"" ; printA; (3)  
"; (4)  
print"A = "" ; printA; print"" ; printA; (5)
```

Neat, ain't it? So how did i come up with that? Well, I really started by coming up with a string of commands, and then I quoted that string. I started by saying I'd like

```
A = "printA...";  
printA;
```

Then I saw that this didn't allow me to include the "A =" part, so I revised it:

```
A = "A = "print"A...";  
print"A = "";  
printA;
```

But this didn't get me passed printing the first line. So i'd have to print it again.

```
A = "A = "print"A = "" ; printA; printA;";  
print"A = "";  
printA;  
printA;
```

This is close...and I fiddled some more. Included below are a couple of short solutions in other programming languages:

```
main() char* s = "main char* s = %c%s%c; printf(s,34,s,34);";printf(s,34,s,34);
( (lambda (x) (list x (list quote (quote x)))) (quote (lambda (x) (list x (list quote (quote x))))))
```

5.6 Show that \leq_m is a transitive relation.

We wish to show that if $A \leq_m B$ and $B \leq_m C$ then $A \leq_m C$.

Proof by construction. If $A \leq_m B$, then there is a function ϕ such that if $x \in A$ then $f(x) \in B$ and if $x \notin A$ then $f(x) \notin B$. That is, there is a bijection f from A to B . If $B \leq_m C$, then there is a bijection g from B to C . Assume $x \in A$. Then $f(x) \in B$. If $f(x) \in B$, then $g(f(x)) \in C$. If $x \notin A$, then $f(x) \notin B$. If $f(x) \notin B$, then $g(f(x)) \notin C$. Hence, if $x \in A$, then $h(x) = g(f(x)) \in C$, and if $x \notin A$, then $h(x) = g(f(x)) \notin C$. Hence, $A \leq_m C$.

5.23 Show that both conditions of Rice's Theorem are necessary for proving that P is undecidable.

Rice's Theorem: If P is any problem about Turing machines that satisfies the following two properties:

- For any TMs, M_1 and M_2 , where $L(M_1) = L(M_2)$, we have $\langle M_1 \rangle \in P \iff \langle M_2 \rangle \in P$. In other words, the membership of a TM M in P depends only on the language of M .
- There exist TMs M_1 and M_2 where $M_1 \in P$ and $M_2 \notin P$. In other words, P is nontrivial: it holds for some, but not all TMs.

then P is undecidable.

Proof by contradiction.

- Assume the second condition is unnecessary. Then, P could be trivial. Assume P is trivial. In that case, either P holds for no TMs or P holds for all TMs. Assume P holds for no TMs. Then we can decide P : we reject all machines. Likewise, if P holds for all TMs, we can accept all machines. Hence, P is decidable.
- Assume the first condition is unnecessary. Then, membership of M in P does not depend solely on the language of M . That is, given M_1 and M_2 where $L(M_1) = L(M_2)$, either $\langle M_1 \rangle \in P$ does not imply $\langle M_2 \rangle \in P$ or either $\langle M_2 \rangle \in P$ does not imply $\langle M_1 \rangle \in P$.

I will give an example for which the relaxation of this condition leads to decidability.

Consider two machines M_1 and M_2 as follows: M_1 = "on input w ,

Read first character of w . If it is a 1, accept and halt. If not, reject and halt.

M_2 = "on input w ,

Read the entire input.

Compute the first 100 digits of π .

Read the first character of w . If it is a 1, accept and halt. If not, reject and halt.

These two machines have the same language: $L(M_1) = L(M_2)$. Now, consider the following property P : A machine accepts its language and halts in 3 steps on all those strings. In this case, $L(M_1) = L(M_2)$, but $\langle M_1 \rangle \in P$ does not imply $\langle M_2 \rangle \in P$. Now, $L_P = \{ \langle M \rangle \mid M \text{ is a TM that has property } P \} = \{ \langle M \rangle \mid M \text{ is a TM that halts on all inputs in 3 steps} \}$. We simply run each machine as input for 3 steps. If that machine accepts, accept. Else reject. In both cases, we halt. Hence, P is decidable.

It is important to note that $L(M_1) = L(M_2)$. We are not saying that M_2 accepts only strings beginning with 1 that are accepted in 3 steps. Such a statement would be inconsistent. The property P , that the machine halts on its input in 3 steps, is Not Part of the language specification

for M_2 . It is simply a property of M_2 . The language L_P doesn't discuss the particular strings that a machine accepts, it only cares about the descriptions of the machines. An incorrect example or proof that the first condition is required would begin with describing two machines whose languages are different. The first condition, and Rice's theorem in general, says nothing about that case.

6.5 Describe Turing machines M and N where M outputs $\langle N \rangle$ and N outputs $\langle M \rangle$. Here's Wim's answer:

Define the TM N as follows:

```
Print("I am N!")
Erase_printed_text;
Obtain, via recursion theorem, own description
Replace in the first command line of the description the capital M by N, and/or the capital N
by M.
Print out this changed description.
```

Conversely, define M to be:

```
Print("I am M!")
Erase_printed_text;
Obtain, via recursion theorem, own description
Replace in the first command line of the description the capital M by N, and/or the capital N
by M.
Print out this changed description.
```

Note that M and N only differ in the first line, and that by the instruction in 4), M will print $\langle N \rangle$, and N will print the description $\langle M \rangle$.

Here's another answer, slightly more complicated:

N = "on input w ,

Erase the input

Using the recursion theorem to get $\langle N \rangle$.

Output $q(\langle N \rangle)$ where q is the computable function that on its input $\langle N \rangle$, computes the description of a Turing machine $P_{\langle N \rangle}$ that outputs that input $\langle N \rangle$ and then halts.

M = "on input w ,

ignore input.

Read the output of N and run it on w .

Clearly M outputs N , since the output of N is $q(\langle N \rangle)$, which is the description of $P_{\langle N \rangle}$. Running $P_{\langle N \rangle}$ outputs N ! Likewise, N outputs M , since $P_{\langle N \rangle} = M$ as defined above.

Note that an answer similar to the following two are incorrect:

M :

Print "I am M!";

Print $\langle N \rangle$

....

There is the same problem with this machine:

M :

Print "la la la!"

Construct $P_{\langle N \rangle}$, the machine that writes $\langle N \rangle$ on the tape and halts.

Both of these are wrong, because in both cases, M has not received a description of N , and yet is somehow generating $\langle N \rangle$. How can it do this? $\langle N \rangle$ can't be in the memory of M unless the source code for M says so. M can't compute $\langle N \rangle$ for the same reasons. Likewise, you can use the self-reference theorem for M to get its own description, but you can't use the self-reference theorem to generate N 's description inside M .

Dates Tell us when else you can take the final. If you didn't do this problem, Do so Immediately! Email Allison!