

# Assignment 2: Solutions and Comments

## CS 2MJ3 Fall 2009

Jick Names

November 6, 2009

Gah! I didn't realize you guys had a test TOMORROW! I hope this isn't too late. Sorry about that, Folks! I thought I should finish marking the assignment so I could write detailed comments, but no time for that now! Comments to follow later (I hope).

Ehr... I mean...

It is pleasing to meet on you. I am the Jick Names from um... Haha! Well, it's country you never hearing about nothing. Other guy, he striking out in picket fence. So ok. Make answers, ai?

### Question 1

The first two questions are about Turing machines and decidability. Let  $L$  be a language consisting of a single string  $w$ , i.e.,  $L = \{w\}$ . The string  $w$  is defined as follows:

$$w = \begin{cases} 111 & \text{if there is life on Mars} \\ 000 & \text{if there is no life on Mars} \end{cases}.$$

Is  $L$  a *decidable* language? Justify your answer (assuming that the question "is there life on Mars?" has a definitive 'yes' or 'no' answer).

### Answer making 1

I no this, but hard to word-making. So beating picket fence man until he say these things...

Every finite language is decidable so both  $\{000\}$  and  $\{111\}$  are decidable languages. Regardless of whether *we* know the answer, the stated assumption is that a definitive answer *exists*, and hence there also exists a Turing machine to recognize  $L$  whether  $L = \{000\}$  or  $L = \{111\}$ . Thus  $L$  is decidable.

### Question 2

Show that decidable languages are closed under the following operations:

- (a) union

- (b) concatenation
- (c) star
- (d) complementation
- (e) intersection

## Answer making 2

More beatings... Picket man say there be many ways to answer... So I hit more. We is assuming  $A$  and  $B$  be decidable languages with Turing machines  $M_A$  and  $M_B$  to recognizing them, ya? Then he make some noises like this...

- (a) Let  $M_C$  be a Turing machine that simulates both  $M_A$  and  $M_B$  processing the same input tape in parallel (this could be imagined to be a two-tape machine, for convenience).  $M_C$  accepts the input string,  $x$ , iff either  $M_A$  or  $M_B$  accept it. Then  $M_C$  recognizes the language  $A \cup B$ .
- (b) Let  $M_C$  be a Turing machine that first converts the input string,  $x = x_1x_2 \cdots x_n$ , into  $n + 1$  pairs of strings:

$$\{(\varepsilon, x), (x_1, x_2x_3 \cdots x_n), (x_1x_2, x_3x_4 \cdots x_n), \dots, (x, \varepsilon)\}$$

These could be written in sequence on the input tape with one delimiter to separate each consecutive pair and another to separate the two elements in each pair.  $M_C$  would then process each pair, simulating  $M_A$  operating on the first element and  $M_B$  on the second.  $M_C$  would accept  $x$  iff a pair were encountered that caused both  $M_A$  and  $M_B$  to accept. Then  $M_C$  recognizes the language  $AB$ .

- (c) Given a nonempty input string,  $x = x_1x_2 \cdots x_n$ , there are only finitely many ways to choose  $k$  strings,  $w_1, w_2, \dots, w_k$ , such that  $x = w_1w_2 \cdots w_k$ , for each value of  $k = 1, 2, \dots, n$ . Hence, we can create a Turing machine,  $M_C$ , that enumerates all such decompositions and simulates  $M_A$  running on each of them.  $M_C$  accepts  $x$  iff there is a decomposition,  $x = w_1w_2 \cdots w_k$  such that  $M_A$  accepts  $w_1, w_2, \dots$ , and  $w_k$ . If  $x$  happens to be empty,  $M_C$  simply accepts immediately. Then  $M_C$  recognizes the language  $A^*$ .
- (d) Deterministic Turing machines are as powerful nondeterministic ones. Hence, we can use the same trick we used to show that the set of regular languages over a given alphabet is closed under complement: namely by assuming, without loss of generality, that  $M_A$  is deterministic and then making a new deterministic Turing machine,  $M_C$ , that is the same as  $M_A$  in all respects except its set of accepting states—which we take to be the complement of  $M_A$ 's set of accepting states. Then  $M_C$  recognizes the complement of  $A$ .
- (e) As in part, (a), we can make a Turing machine,  $M_C$ , that accepts  $x$  iff both  $M_A$  and  $M_B$  accept it.

### Question 3

Happy Talky-man only *see* beating and write all this in very fastly! Maybe he like. So maybe while write test, make a wrong in person beside you head with bat for ball of base. Talky-man like and probably give answer fastly!

This is a  $\lambda$ -calculus question. Apply  $\beta$ -reductions to each of the following  $\lambda$ -terms until no more  $\beta$ -reductions can be applied:

1.  $(\lambda x.xx)(\lambda y.yx)z$

$$\begin{aligned}(\lambda x.xx)(\lambda y.yx)z &= ((\lambda x.xx)(\lambda y.yx))z && \text{[application left associates]} \\ &\rightarrow_{\beta} ((xx)\{x \mapsto (\lambda y.yx)\})z && \text{[substitution]} \\ &= ((\lambda y.yx)(\lambda y.yx))z \\ &\rightarrow_{\beta} ((yx)\{y \mapsto (\lambda y.yx)\})z && \text{[substitution]} \\ &= ((\lambda y.yx)x)z \\ &\rightarrow_{\beta} ((yx)\{y \mapsto x\})z && \text{[substitution]} \\ &= (xx)z = xxz && \text{[application left associates]}\end{aligned}$$

2.  $(\lambda x.(\lambda y.(xy))y)z$

$$\begin{aligned}(\lambda x.(\lambda y.(xy))y)z &\rightarrow_{\beta} (\lambda x.((xy)\{y \mapsto y\}))z \\ &= (\lambda x.(xy))z \\ &\rightarrow_{\beta} (xy)\{x \mapsto z\} = zy\end{aligned}$$

3.  $((\lambda x.xx)(\lambda y.y))(\lambda y.y)$

$$\begin{aligned}((\lambda x.xx)(\lambda y.y))(\lambda y.y) &\rightarrow_{\beta} ((xx)\{x \mapsto (\lambda y.y)\})(\lambda y.y) \\ &= ((\lambda y.y)(\lambda y.y))(\lambda y.y) \\ &\rightarrow_{\beta} (y\{y \mapsto (\lambda y.y)\})(\lambda y.y) \\ &= (\lambda y.y)(\lambda y.y) \\ &= (\lambda y.y) && \text{[just repeating previous line]}\end{aligned}$$

4.  $((\lambda x.\lambda y(xy))(\lambda y.y))w$

$$\begin{aligned}((\lambda x.\lambda y(xy))(\lambda y.y))w &= (((\lambda x.\lambda v.(xv))(\lambda y.y))w) \\ & && \text{[use } =_{\alpha} \text{ so } y \text{ not "caught" by } \lambda y\text{]} \\ &\rightarrow_{\beta} ((\lambda v.(xv))\{x \mapsto (\lambda y.y)\})w \\ &= (\lambda v.((\lambda y.y)v))w \\ &\rightarrow_{\beta} (\lambda v.v)w \\ &\rightarrow_{\beta} w\end{aligned}$$

### Question 4

This is a question about relations. Show that if  $R_1$  and  $R_2$  are relations such that  $R_1 \subseteq R_2$ , then  $\text{index}(R_1) \geq \text{index}(R_2)$ .

## Making of proving 4

Now funny picket man writing very fastly just because see bat for ball of base.  
Even no beating! Land of opportunity from knocks, just likely saying!

*Proof.* Define  $f : X/R_2 \rightarrow X/R_1$  as  $f([x]_{R_2}) = [x]_{R_1}$ . If  $X = \emptyset$  the result follows trivially, so assume  $X \neq \emptyset$  and let  $x, y \in X$ . If  $f([x]_{R_2}) = [x]_{R_1} = [y]_{R_1} = f([y]_{R_2})$  then  $(x, y) \in R_1 \subseteq R_2$ . Therefore,  $[x]_{R_2} = [y]_{R_2}$ , which shows that  $f$  is injective. Hence, by the theorem covered during the tutorial on Friday, October 30,  $\text{index}(R_2) = |X/R_2| \leq |X/R_1| = \text{index}(R_1)$ .  $\square$