Homework 10

Austin Frownfelter Matthew Bialecki

February 5, 2018

1 Problem 14

> "Show by diagonalization that there is a language accepted by a Java program that is not accepted by any mini-Java program."

Assume all languages accepted by Java programs are accepted by mini-Java programs.

Consider a table of all valid mini-Java programs. The rows represent the programs and the columns represent the inputs. Each cell is a 1 if the mini-Java program accepts that input and a 0 if it rejects the input. The table is organized such that the (i, i) cell is the ith program running with itself as input.

Now consider a row which represents a Java program where each i element is a 0 if the ith program accepts the ith program as input, and a 1 if it rejects. This row cannot exist in this table since it is exactly opposite the diagonal of the table. Contradiction. Therefore, there exist languages accepted by Java programs that are not accepted by any mini-Java programs.

2 Problem 15

2.1 Part a

> Show that A can be accepted by a log-space Turing machine.

Consider a TM M which accepts x iff x is a string of properly nested brackets, which is defined as follows:

> Write a 0 on the work tape
>
> For each character a in x starting from the left
>
> > If a is '(', increase the counter on the work tape
> >
> > If a is ')' and the counter on the work tape is 0, reject
> >
> > Otherwise, if a is '(', decrease the counter on the work tape
>
> If the counter on the work tape is 0, accept; otherwise, reject.

M decides the language A. M only writes a counter on the work tape. Since the counter ranges from 0 to the length of x and the length of x can be represented with a number of size $log(|x|)$, M uses only log space. Therefore, A can be accepted by a log-space Turing machine.

2.2 Part b

Show that B can be accepted by a log-space Turing machine.

Consider a TM M which accepts x iff x is a string of properly nested brackets, which is defined as follows:

Write a 0 on the work tape

For each character a in x starting from the left

If a is '(', increase the counter on the work tape

If a is ')' and the counter on the work tape is 0, reject

Otherwise, if a is ')', decrease the counter on the work tape

If the counter on the work tape is not 0, reject

Clear and write a 0 on the work tape

For each character a in x starting from the left

If a is '[', increase the counter on the work tape

If a is ']' and the counter on the work tape is 0, reject

Otherwise, if a is ']', decrease the counter on the work tape

If the counter on the work tape is not 0, reject

Begin at the start of the input tape

while current character c in x:

if x is '(':

while the next(c) is '(': move c to the right
If next(c) is ')', skip to the next loop iteration
If next(c) is ']', reject
Start a counter n
For each character a in x starting from c

If a is '(', increase the counter n on the work tape
If a is ')', decrease the counter n on the work tape
If a is ')' and the counter is 0, set P_right to the position of a

Clear counter n

For each character a in x starting from c+1

If a is '[', increase the counter n on the work tape

If a is ']', decrease the counter n on the work tape

If a is ']' and the counter is 0, set B_right to the position of a

if P_right < B_right, reject

if x is '[':

while the next(c) is '[': move c to the right

If next(c) is ']', skip to the next loop iteration

If next(c) is ')', reject

Start a counter n

For each character a in x starting from c

If a is '[', increase the counter n on the work tape

If a is ']', decrease the counter n on the work tape

If a is ']' and the counter is 0, set B_right to the position of a

Clear counter n

For each character a in x starting from c+1

If a is '(', increase the counter n on the work tape

If a is ')', decrease the counter n on the work tape

If a is ')' and the counter is 0, set P_right to the position of a

if B_right < P_right, reject

move c to c+2

accept

M decides B, and uses at most $4*log(|x|)$ space - 3 pointer variables (P_right, B_right, and c) and the counter n for searching for the matching close. The definition of M is an explicit definition for the following: First, make sure the parentheses and brackets are nested properly individually. Then, make sure there are no cases of ([*)] where * is anything.

Therefore, B is accepted by a log-space Turing machine.