cpsc 313 assn6

Ali Akbari

TOTAL POINTS

25 / 30

QUESTION 1

1 1a 8 / 10

- **0 pts** Correct
- **1 pts** minor errors

✓ - **2 pts** **extra move at start without explaning how not to move tape**

- **3 pts** only "extra move" or similar is stated, not clearly defined
- **4 pts** writes add extra state but without any explanation of how it is used
- **6 pts** assume that opposite of accept-in-even is accept-in-odd
- **7 pts** preprocessing state is not implementable
- **7 pts** There is no reduction to an unsolvable problem
- **6 pts** Reduction is in the wrong direction
- **9 pts** incorrect
- **10 pts** Missing
- **8 pts** Includes an undecidable TM on the inside of the reduction
- **6 pts** works only if machine accepts in even/odd but not both
- **6 pts** Reduction doesn't cover all cases
- **7 pts** No Reduction to a KNOWN unsolvable problem

QUESTION 2

2 1b 7 / 10

- **0 pts** Correct
- **0 pts** transitions for new state not given (design will still work though)
- **1 pts** Minor errors

✓ - **3 pts** **construction is missing details**

- **3 pts** assume ha and hr are meaningful in internal decision problem
- **3 pts** state q is not a new state, and may be visited by original machine resulting in flaw in reduction
- **4 pts** Reduction doesn't cover all cases
- **6 pts** Reduction in the wrong direction
- **6 pts** There is no reduction to an unsolvable problem
- **7 pts** major errors
- **7 pts** trivial proof based on unproven decision problem
- **8 pts** incorrect
- **10 pts** missing
- **7 pts** Includes an undecidable TM on the inside of the reduction

1 all TMs need ha and hr

QUESTION 3

3 1c 10 / 10

✓ - **0 pts** **Correct**

- **7 pts** assume an unsolvable problem that was not covered and trivialized the proof
- **7 pts** major error in use of unsolvable problem
- **8 pts** there is no reduction to an unsolvable problem
- **4 pts** reduction is done in the reverse direction but otherwise is valid and non-trivial
- **10 pts** Missing
- **3 pts** error in the reduction, not if and only if
- **7 pts** reduction in wrong direction
- **5 pts** assumed a problem that is unsolvable that was not covered but did not trivialize the proof
- **1 pts** minor error
- **5 pts** solution does not reflect a clarity of thought regarding reductions; unclear; details missing
- **9 pts** incorrect
- **1 pts** minor errors

- 2 pts minor errors

gradescope

CPSC 313 Fall 2020
Assignment 6
Ali Akbari
30010402

1. Prove that the following decision problem is undecidable by reducing it to a known unsolvable problem. Be sure to clearly define your reduction. It should be written like a pseudocode program that is straightfoward to implement. Two different TAs reading it should have the same understanding of how your reduction works.

(a) Given a TM T and a word w, does T accept w in an even number of moves?

D = Does T accept w in an even number of moves?
By contradiction we first assume that D is solvable. Therefore there exist an always halting turing machine T_D that decides this problem. We will use it to solve the problem of:
E = "given T and w, does T halt on w."

Our algorithm is the following for T_E:

```
bool does_T_halt_on_w(TM T, WORD w) {

T ' = T with ha and hr swapped.
Ti = T with a state q in front of the machine that makes no difference other than
increasing the number of transitions by one, i.e even number moves become odd.
Ti ' = T with ha and hr swapped.

if (T_accepts_w_in_even_number_of_moves(T, w) == True){
return true
}
else if (T_accepts_w_in_even_number_of_moves(T ' , w) == True){
return true
}

if (T_accepts_w_in_even_number_of_moves(Ti, w) == True){
return true
}
else if (T_accepts_w_in_even_number_of_moves(Ti ' , w) == True){
return true
}

return false
}
```

The subroutines represents our T_D.
The first if statement returns true if and only if T accepts w in an even number of moves.
The second statement returns true if and only if T rejects w in an even number of moves.
The third if statement returns true if and only if T accepts w in an odd number of moves.
The second statement returns true if and only if T rejects w in an odd number of moves. The inner if statements suggest that T or T ‘ halts on w, the return false statement suggests that T does not halt on w. However the algorithm halts if we assume that T_D halts. In the lecture notes we are shown that if T accepts w it is also undecidble therefore it cannot halt. Therefore such a Turing machine T_D cannot exist.

1 1a **8 / 10**

- **- 0 pts** Correct
- **- 1 pts** minor errors
- ✓ **- 2 pts extra move at start without explaning how not to move tape**
- **- 3 pts** only "extra move" or similar is stated, not clearly defined
- **- 4 pts** writes add extra state but without any explanation of how it is used
- **- 6 pts** assume that opposite of accept-in-even is accept-in-odd
- **- 7 pts** preprocessing state is not implementable
- **- 7 pts** There is no reduction to an unsolvable problem
- **- 6 pts** Reduction is in the wrong direction
- **- 9 pts** incorrect
- **- 10 pts** Missing
- **- 8 pts** Includes an undecidable TM on the inside of the reduction
- **- 6 pts** works only if machine accepts in even/odd but not both
- **- 6 pts** Reduction doesn't cover all cases
- **- 7 pts** No Reduction to a KNOWN unsolvable problem

gradescope

(b) Given a TM T, a word w, and a state q such that $q \neq ha$ and $q \neq hr$, does T ever enter q when processing w.

D = Does T ever enter q when processing w?
By contradiction we first assume that D is solvable. Therefore there exist an always halting turing machine T_D that decides this problem. We will use it to solve the problem of
E ="given T and w, does T halt on w."

Our algorithm is the following for T_E:

```
bool does_T_halt_on_w(TM T, WORD w) {

T ' = T but with a new state q, where all states in T that pointed to ha, now point
the new q.

T " = T but with a new state q, where all states in T that pointed to hr, now point
the new q.

if (does_enter_q_while_processing_w(T ', w) == True){
return true
}
if (does_enter_q_while_processing_w(T " , w) == True){
return true
}

return false
}
```

The subroutines represents our T_D .The first if statement returns true if and only if T ' accepts w if it enters q(ha of T) while processing w. The second statement returns true if and only if T " accepts w if it enters q(hr of T) while processing w. The inner if statements suggest that T ' or T " halts on w, the return false statement suggests that T does not halt on w. However the algorithm halts if we assume that T_D halts. In the lecture notes we are shown that if T accepts w it is also undecidble therefore it cannot halt. Therefore such a Turing machine T_D cannot exist.

2 1b **7 / 10**

- **- 0 pts** Correct
- **- 0 pts** transitions for new state not given (design will still work though)
- **- 1 pts** Minor errors
- ✓ **- 3 pts construction is missing details**
- **- 3 pts** assume ha and hr are meaningful in internal decision problem
- **- 3 pts** state q is not a new state, and may be visited by original machine resulting in flaw in reduction
- **- 4 pts** Reduction doesn't cover all cases
- **- 6 pts** Reduction in the wrong direction
- **- 6 pts** There is no reduction to an unsolvable problem
- **- 7 pts** major errors
- **- 7 pts** trivial proof based on unproven decision problem
- **- 8 pts** incorrect
- **- 10 pts** missing
- **- 7 pts** Includes an undecidable TM on the inside of the reduction

1 all TMs need ha and hr

gradescope

(c) Given a TM T, and two words w and x, does T accept either wx or xw?

D = Does T accept w either wx or xw?
By contradiction we first assume that D is solvable. Therefore there exist an always halting turing machine T_D that decides this problem. We will use it to solve the problem of E = "given T, does T accept ε." Observe that if w, x = ε then wx, and xw = $\varepsilon * \varepsilon = \varepsilon$.

Our algorithm is the following for T_E:

```
bool T_accepts_epsilon(T) {
return T_accepts_wx_or_xw(T, epsilon, epsilon)
}
```

The subroutines represents our T_D.
The return statement returns true if and only if T_D accepts w and x.The return statement suggests that T_D does halt on w, and x when equal to epsilon. However the algorithm T_E halts if we assume that T_D halts. In the lecture notes we are shown that if T accepts ε it is also undecidble therefore it cannot halt. Therefore such a Turing machine T_D cannot exist.

3 1c 10 / 10

- ✓ **- 0 pts** Correct
- **- 7 pts** assume an unsolvable problem that was not covered and trivialized the proof
- **- 7 pts** major error in use of unsolvable problem
- **- 8 pts** there is no reduction to an unsolvable problem
- **- 4 pts** reduction is done in the reverse direction but otherwise is valid and non-trivial
- **- 10 pts** Missing
- **- 3 pts** error in the reduction, not if and only if
- **- 7 pts** reduction in wrong direction
- **- 5 pts** assumed a problem that is unsolvable that was not covered but did not trivialize the proof
- **- 1 pts** minor error
- **- 5 pts** solution does not reflect a clarity of thought regarding reductions; unclear; details missing
- **- 9 pts** incorrect
- **- 1 pts** minor errors
- **- 2 pts** minor errors

gradescope