Name: Firas Aboushamalah

Student Number: 250920750

Assignment 4

For each of the following languages, prove, without using Rice's Theorem, whether it is (i) in D, (ii) in SD but not in D, or (iii) not in SD.

1. **L1 = {<*M*> | {*ε*, ab, abab} ⊆ *L*(*M*)}**

SD

1. Use dovetailing to run M on L(M), in particular {*ε,* ab, abab}
2. Check each step of M run on each of the input strings to see if either accepts
3. If it accepts, halt and accept

¬D

H = {<M, w> | TM M halts on input w}. Let H ≤ L1 = true.

Reduction R(<M, w>) =

1. Constructing <M’> where M’ operates like so:
   1. Erase tape
   2. Write w on the tape
   3. Run M’ on w
   4. Accept w
2. Return M’

If Oracle exists, then C = Oracle(R(<M, w>)) decides H.

* R can be implemented as a TM
* <M, w> ∈ H: M halts on w, so M’ accepts everything, namely {*ε,* ab, abab}. Oracle accepts
* <M, w> ∉ H: M does not halt on w, so M’ does not accept anything, including {*ε,* ab, abab}. Oracle does not accept.

However, there is no way to decide H, therefore L1 is undecidable as well.

1. **L2 = {<*M*> | L(*M*) ∩ (ab)\* is infinite}**

¬SD

The proof can be made in the form of a reduction from ¬H where

¬H = {<M, w> | TM M does not halt on input string w}. Let H ≤ L2 = True.

Reduction R(<M, w>) =

1. Construct description of <M’> for M’(x):
   1. Save x
   2. Erase the tape
   3. Write w on the tape
   4. Run M’ on w for |x| steps
      1. If M halts within |w| steps then loop
   5. Otherwise accept
2. Return M’

Assume C = Oracle(R(<M, w>)) semidecides ¬H.

* <M, w> ∈ ¬H: M does not halt on w, so M’ accepts everything, therefore M’ accepts infinitely many strings x from L(M) ∩ (ab)\* 🡪 Oracle accepts
* <M, w> ∉ ¬H: M halts on w within |w| steps, so M’ only accepts strings x that within a finite set 🡪 Oracle does not accept.

However, this is a contradiction because ¬H is not semidecidable, therefore L2 is not semidecidable.

1. **L3 = {<M> | L(M) ∩ (ab)\* is finite}**

¬SD

We can prove this similar to L2 from ¬H where

¬H = {<M, w> | TM M does not halt on input string w}. Let H ≤ L3 = True.

Reduction R(<M, w>) =

1. Construct description of <M’> for M’(x):
   1. Save x
   2. Erase the tape
   3. Write w on the tape
   4. Run M’ on w
   5. Accept
2. Return M’

Assume C = Oracle(R(<M, w>)) semidecides ¬H.

* <M, w> ∈ ¬H: M does not halt on w: M’ does not accept any string x, therefore L(M) ∩ (ab)\* is finite. Oracle accepts.
* <M, w> ∉ ¬H: M halts on w: M’ accepts every string x, therefore M is infinite. Oracle does not accept.

However, this is a contradiction because ¬H is not semidecidable, therefore L3 is not semidecidable.

1. **L4 = {<M> | L(M) ∩ (ab)∗ = ∅}**

¬SD: Proof with a reduction from ¬H = {<M, w> | TM M does not halt on input string w}.

R(<M, w>) =

1. Construction of M’:
   1. Save input x on tape
   2. Erase tape
   3. Write w on tape
   4. Run M on w
2. Return M’.

Oracle semidecides ¬H :

* <M, w> ∈ ¬H: M does not halt on w, so M’ does not accept anything, namely M’ does not accept any string x. Therefore, M’ accepts the empty set. Oracle accepts.
* <M, w> ∉ ¬H: M halts on w, so M’ accepts everything. Therefore, every string is accepted meaning it is infinite. Therefore, Oracle does not accept.

However, this is a contradiction because ¬H is not semidecidable, therefore L4 is not semidecidable.

1. **L5 = {<M> | L(M) ∩ (ab)∗ ≠ ∅}**

SD

1. Run M on all the strings in the union of L(M) and (ab)\* which is also just (ab)\*
2. If M accepts any of the strings, accept

¬D

H = {<M, w> | TM M halts on input w}. Let H ≤ L5 = true.

R(<M, w>): =

1. Construct M’:
   1. Erase tape
   2. Write w on tape
   3. Run M on w
   4. Accept
2. Return M’

If Oracle exists, then C = Oracle(R(<M, w>)) decides H.

* <M, w> ∈ H: M halts on w, so M’ accepts everything, specifically the strings in (ab)\*, therefore the language is not empty. Oracle accepts.
* <M, w> ∉ H: M does not halt on w, so M’ does not accept anything which means the language is empty. Oracle rejects.

However, there is no way to decide H, therefore L5 is undecidable as well.

1. **L(6) = {< M > | L(M) ≠ L(M’) for any other TM M’}**

This language is decidable because it is the language of the empty set. The reason is because instead of checking infinitely many turing machine encodings, we only have to check one to decide whether or not it is accepted in the language.

1. **L(7) = {< M > | ¬L(M) ∈ D}.**

¬SD: Proof with a reduction from ¬H = {<M, w> | TM M does not halt on input string w}.

1. Construct M’:
   1. Save x
   2. Erase tape
   3. Write w on tape
   4. Run M on w
   5. accept
2. Return M’

* <M, w> ∈ ¬H:does not halt on w, so M’ does not accept anything, namely M’ does not accept any string other than the empty set. However, the complement of the empty set is sigma star which is in D. Oracle accepts.
* <M, w> ∉ ¬H: M halts on w, so M’ accepts everything which means L(M’) is in H. Oracle does not accept.

However, this is a contradiction because ¬H is not semidecidable, therefore L7 is not semidecidable.

1. **L(8) = {< M > | L(M) ∈ SD}.**

This is decidable because the language L(M) is in SD is true for any Turing Machine (if it is undecidable there will not be a TM for it). Therefore, the descriptions (encodings) of the Turing Machines are all correct so they can all be decidable.

2) Prove for each of the languages whether you can use rice’s theorem or not (20 poitns)

1. We can use rice’s theorem because the language has the nontrivial property where we accept always on epsilon, ab, abab. So the language is not in D.
2. We can use rice’s theorem because the language’s non trivial property is to accept whenever the intersection is infinite. So the language is not in D.
3. We can use rice’s theorem because the language’s non trivial property is to accept whenever the intersection is finite. So the language is not in D.
4. We can use rice’s theorem because the language’s non trivial property is to accept ONLY when the set is empty and reject otherwise. So the language is not in D.
5. We can use rice’s theorem because the language’s non trivial property is to accept ONLY when the set is NOT EMPTY and reject if it IS EMPTY. So the language is not in D.
6. We cannot use Rice’s Theorem here because the language has a trivial property (always true) where the language is always empty. Therefore it is in D.
7. We can use rice’s theorem because the language’s non trivial property is to accept ONLY when the language is in D. Therefore it is not in D.
8. We cannot use Rice’s theorem here because the language has a trivial property (always true) where every TM accepts a language in SD (and thereby in D because D is a subset of SD). So it is always true and in D.