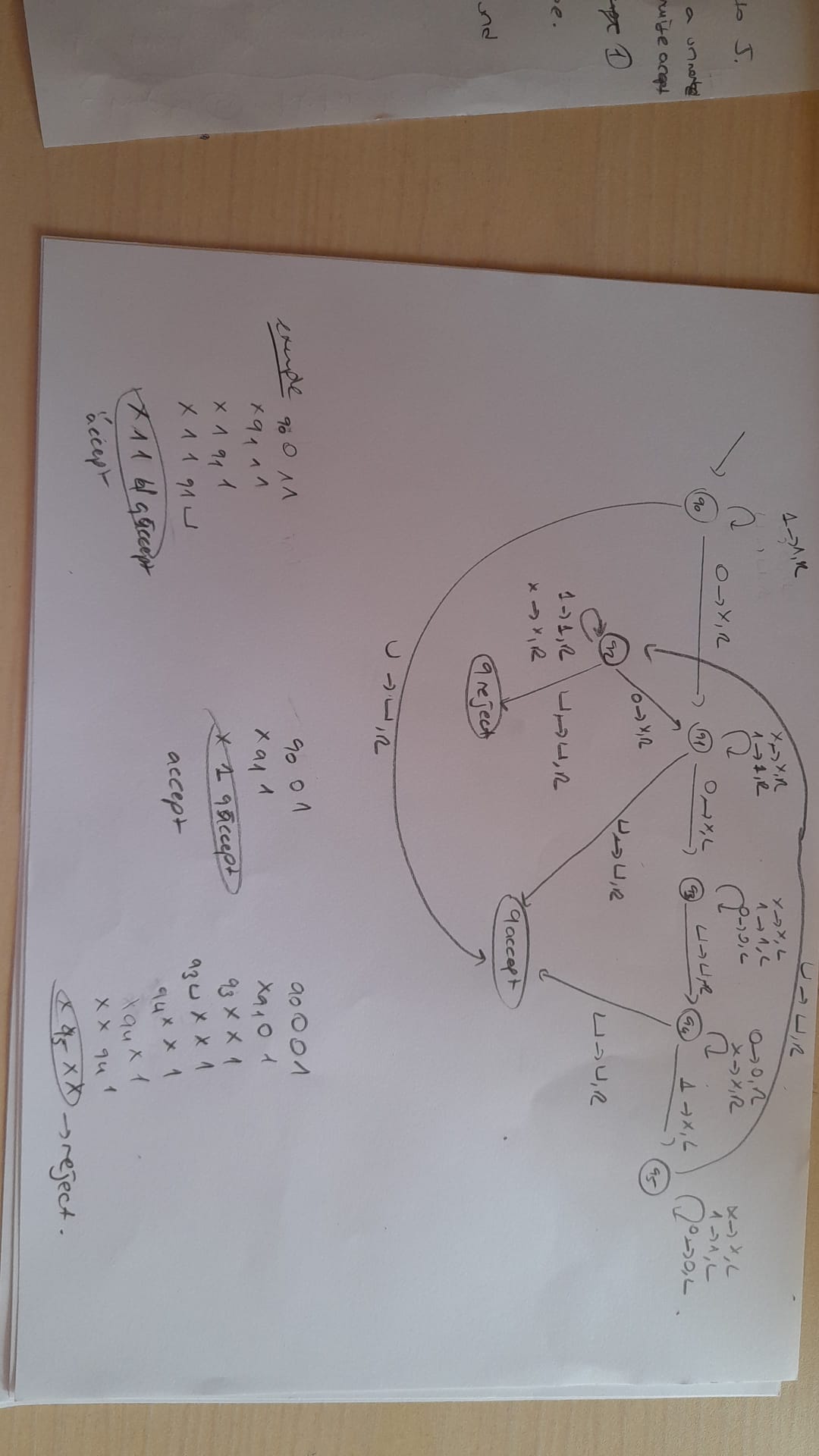
A piece of paper with writing on it

Description automatically generated



1. To prove the language is decidable, we need to construct a Turing Machine that decides the given language A:

The following Turing Machine T decides language A:

“On input <M>:

1. Construct a DFA B that accepts every string containing an odd number of 1s.
2. Construct another DFA C such that L(C) = L(M) ∩ L(B)
3. Control if L(C) is ∅ by using EDFA decider T with Theorem 4.4
4. If T accepts, accept; otherwise, T will reject so, reject.

Theorem 4.4

* This theorem for EDFA decider T

T is simulating with input <A> where A is DFA

1. Mark the initial state
2. Repeat there are no new states to be marked.
3. Mark any state which has transition with another marked states
4. If no accept state is marked, accept; else reject it (since we marked a accept state which means the DFA is not empty.)
5. For this question, we can use emptiness problems as guidelines:

On input <G>, where G is CFG

1. Mark all the terminal symbols
2. Repeat till there are nothing to mark
3. Mark any variable that has connections with marked nodes/ones
4. If the start variable is not marked accept; else reject

So we can find a Turing machine for this. So CFG is decidable.

TM should be halting on all inputs restricting the kinds of languages that can be recognized. Turing machine U recognizes Atm:

U = “On input <M,w>, where M is Turing Machine and w is string.

1. Simulate M on input w.
2. If M ever enters any accepts, accept. Otherwise, reject it.

This machine loops on input <M,w> if also M loops on w, the machine will be undecidable. If the algorithm could have some way to not halting it would be decidable. But it is not the case for this.