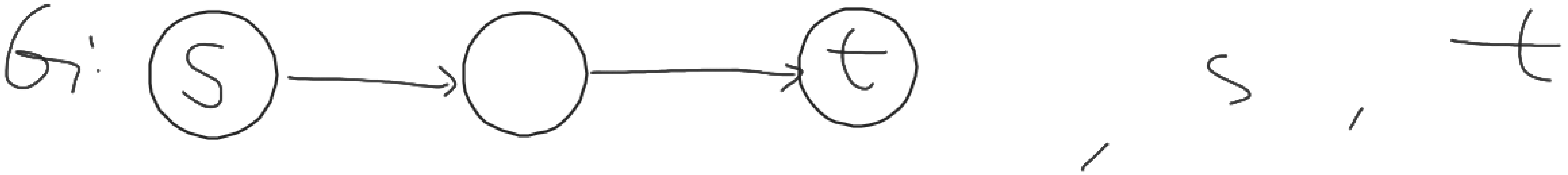
CSCC63 TUT 0002
Tutorial 8

Hamiltonian Path Reductions

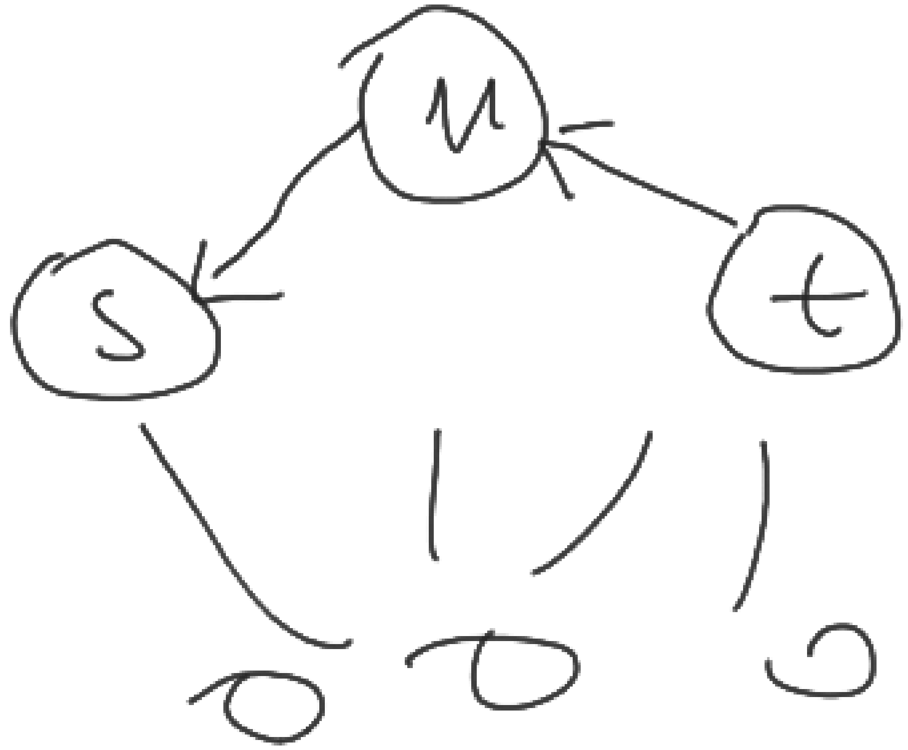
Ham-Path = { <G = (V, E), s, t> | G is a graph, and has a hamiltonian path from s to t }



a path from s to t consisting of |V| nodes, |V| - 1 edges without repeating any nodes.

Ham-Cycle = { <G = (V, E)> | G is a graph, and has a hamiltonian a cycle }

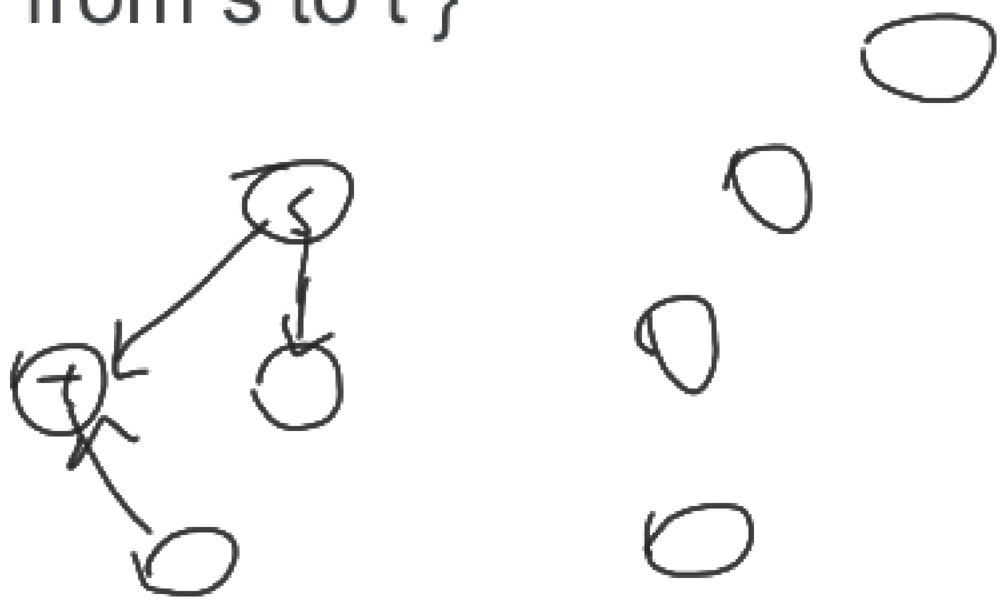
P on input <G = (V, E), s, t>:
  let G' = (V', E') be a copy of G
  add a new node u to V'
  add (t, u), (u, s) to E'
  return G'



<G, s, t> is in HAM-PATH iff G' in HAM-CYCLE
Suppose that <G, s, t> is in HAM-PATH then there is a path from s to t that covers all nodes, connecting them via a new node u creates a cycle in G'

Suppose that <G, s, t> is not in HAM-PATH then there is no path from s to t such that it covers all nodes, connecting them will not make it a HAM-CYCLE because there will be some node that the cycle cannot reach.

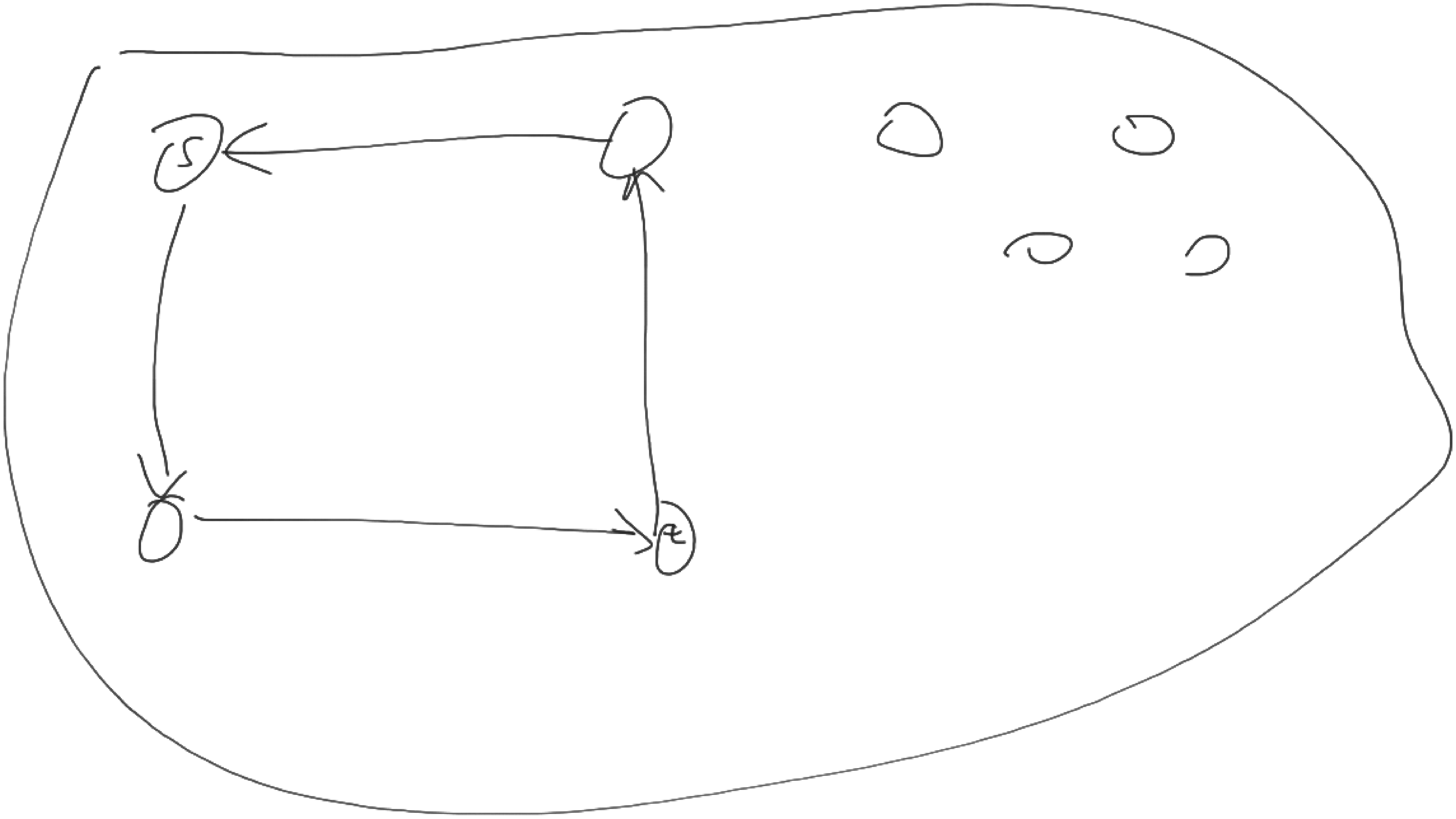
Half-PATH = { <G=(V, E), s, t> | G has a path of size |V|/2 from s to t }



P on input <G = (V, E), s, t>:
  let V' be |V| disconnected nodes
  return <G' = (V u V', E), s, t> # G' has 2|V| nodes

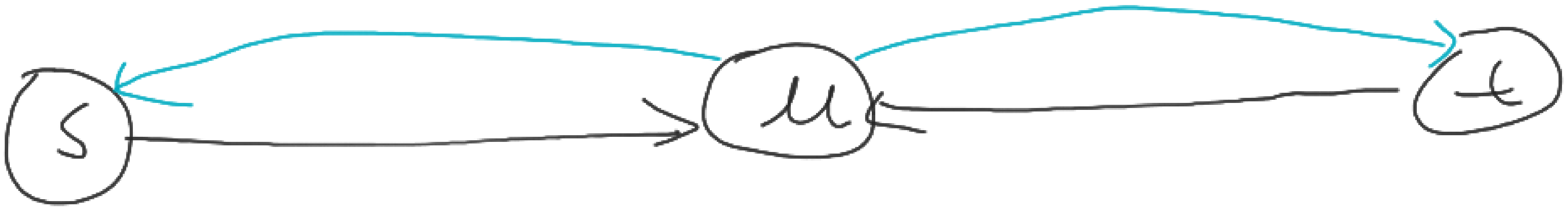
suppose <G = (V, E), s, t> is in HAM PATH
then there is a path of size |V| from s to t, G' will still have this size |V| path, however G' has an additional |V| number of disconnected nodes. This means that the path in G' from s to t will be made up of half its nodes.

suppose <G = (V, E), s, t> is not in HAM PATH
then there is no path size |V| from s to t, G' is simply the same graph with an additional |V| number of nodes. thus G' will not have a path from s to t that is half its nodes.


s
t

UHAM-PATH = { <G, s, t> | G is an undirected graph with a hamiltonian path from s to t }

naive approach: add reverse edge for every edge



doesn't work, we cannot make every directed edge an edge

Q3 Help:

To say that L(M) is context free, means there exists cfg G such that L(M) = L(G)
let G_i be enumerations over CFGs
say we try to prove its recognizable:

```
R on input <M>:
  for i = 0 to infinity:
    if L(M) = L(G_i):
      accept
```

try to show that its corecognizable

```
R-c on input <M>:
  for w in L(M):
    for i in 0 to infinity:
      assert w not in L(G_i)
```

do these work?

Q6 Help:
Think about what number to put for m given that G has a k sized clique.

Edge-Dense: <G = (V, E), m, n>

there is an n sized subset such that within this subset there are at least m edges.

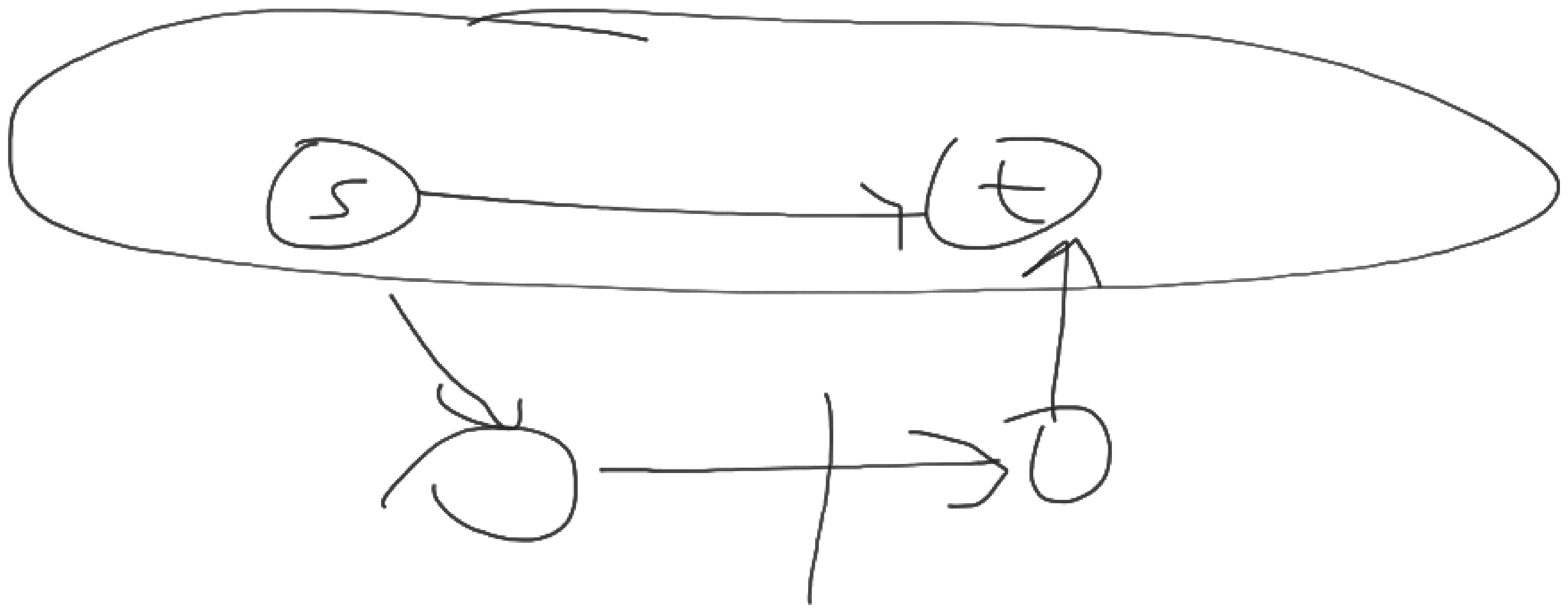
Q1 Help:
For recognizers/co-recognizers

you can check if a string w can be generated by a CFG (G)

L1 = { <G> | exists w in Sigma* where w and 0w in L(G) }
L1-c = { <G> | forall w in Sigma* where not (w and 0w in L(G)) }

Q5 Help:
Ham-Path yes instance properties:
G = (V, E) has a path from s to t of size |V|

Power-Path yes instance properties:
G = (V, E) has a path from s to t of size 2^n



path certificate

```
C = [s, u1, u2, ..., t]
for i = 0 to |C| - 1:
  assert that (C[i], C[i + 1]) in E # O(1) time given E is an adjacency matrix
```