Exercise 34

Samantha Hafner

10/13/19

I outline a proof that SIP on directed graphs is NP-complete which does not rely on an assumption of NP-completeness of another language.

All languages L in NP can solved in polynomial time by a nondeterministic Turing machine M.

I construct a graph G which represents possible paths of a nondeterministic Turing machine, and a graph H which represents an accepting execution of that machine such that H will be isomorphic to a subgraph of G if and only if the nondeterministic Turing machine accepts its input.

Let $P(n)$ be a polynomial function with even integral outputs which is at least the runtime of M.

The graph G, with an example pictured in the figure, is as follows:

G's nodes are arranged into $P(n)+1$ data chunks, with $P(n)$ transition chunks connecting them. Each data chunk contains four columns of nodes representing the state of the machine and tape:

- The first contains a node for each state of M
- The second contains $P(n)$ nodes, each representing a possible head position on the tape
- The third represents the contents of the tape, split into $P(n)$ groups each of which has $|\Gamma|$ nodes in it. All possible edges are present within each group representing a single tape position.
- The fourth is an exact copy of the third.

Each transition chunk contains a group of nodes representing possible transitions:

- M's transition function δ can be represented as a list of transitions with finite length constant with respect to n in the from $(state,\ character) \rightarrow (state,\ character,\ motion)$.
- In order to accept nondeterministic Turing machines that take fewer than $P(n)$ transitions, the $|\Gamma|$ transitions of the form $(Q_accept,\ \cdot\ ) \rightarrow (Q_accept,\ \cdot\ ,\ R)$ should be modeled in addition.
- Each of these transitions is represented in G $P(n)$ times, once for each possible head position.
- Each representation consists of a single node with three directed edges headed to it:
  - One from the node in the first column of the previous chunk, representing the machine state required by the transition
  - One from the node in the second column of the previous chunk, representing the head position the transition represents
  - And one from the node in the fourth column of the previous chunk, representing the tape character required by the transition at the head position the transition represents.
- And three directed edges headed out of it:
  - One to the node in the first column of the next chunk,
  - representing the machine state produced by the transition
  - One to the node in the second column of the previous chunk, representing the head position the transition reaches
  - And one to the node in the third column of the previous chunk, representing the new tape character the transition produces.
- There are $k * P(n)$ nodes of this type where k is constant with respect to n.
- Transitions which cause the machine to travel beyond the $P(n)^{th}$ tape position can be modeled as holding head position constant instead as they cannot be reached in any event.

There are four additional constructs in G:

- Each node representing a tape character at a specific position has an edge to the corresponding node in the subsequent data chunk
- In order to disentangle the motional path of the Turing machines head from the eventual subgraph H which will have to be found, there is a group of scrambling nodes and edges which serves to scramble the order of the tape with respect to a subgraph H, but not with respect to the transitions. In each data chink, for every pair of tape positions n1, n2 and characters c1, c2, there is a node with two edges from the third column of that data chunk coming from c1 at n1 and from c2 at n2, and corresponding edges to the nodes representing c1 at n1 and c2 at n2 in the fourth column of that chunk.
- Instead of the first data chunk's third column and scrambling nodes and edges, there is a single node with one edge to the start state, tape head, and starting character at each tape position.
- Instead of the last data chunk's fourth column and scrambling nodes and edges, there is a single node with one edge from the accept state.

The graph H, with an example pictured in the figure, is arranged into chunks similarly. Each data chunk contains three columns:

- The first is a single node representing the current state
- The first is a single node representing the current head position
- The third contains $P(n)$ nodes representing the character on each tape position,
- The fourth contains $P(n)$ nodes representing the character on each tape position after scrambling

Each transition chunk consists of a single transition node with edges from the previous chunk's state, position, and one tape character node, and to the same nodes on the next chunk. There are four additional constructs in G:

- Each node representing a tape character has an edge to a corresponding node in the subsequent data chunk

- $P(n)/2$ nodes corresponding to the scrambling nodes in H, each receiving an edge from two nodes from the first group and sending an edge to two nodes in the second group.

- Instead of the first data chunk's third column and scrambling nodes and edges, there is a single node with one edge to the state node, tape position node, and each tape node in fourth column of the first chunk.

- Instead of the last data chunk's fourth column and scrambling nodes and edges, there is a single node with one edge from the state node.

I now claim that H will be isomorphic to a subgraph of G if and only if M accepts the input represented by G.

If M accepts the input represented by G, then there exists some list of states of the Turing machine, its head position and its tape contents $((q_0, n_0, (c_{0,0}, c_{0,1}, ..., c_{0,p(n)}))..., (q_{p(n)}, n_{p(n)}, (c_{p(n),0}, c_{p(n),1}, ..., c_{p(n),p(n)})))$ such that the first state represents the machine state being the start state, the head position at the far left, and the tape contents equal to the input, and the last state representing the machine at the accept state.

The subgraph of H containing:

- One node from the first and second column of each data chunk representing the states and head positions $q_0...q_p(n)$ and $n_0...n_p(n)$,

- One node from each group in the third and fourth column of each data chunk representing the characters $c_{i,j}$

- A scrambling node with edges from and to the nodes representing tape character c_{i,n_i} and $c_{i,n_{i+1}}$ in the third and fourth columns respectively of each data chunk

- The single node representing the transition taken at each step of the computation from each transition chunk

- And the original root and final leaf node described as additional constructions,

is isomorphic to G. Therefore, if H will be isomorphic to a subgraph of G if M accepts the input represented by G.

A subgraph g of H which is isomorphic to G must have the following properties:

- No node in g can represent one type of construct (scrambling node, data node, transition node, original root node, or final leaf node) in G an another in H because they are each distinguishable by the number of edges headed to/from them.

- g must not contain any two nodes from corresponding to the same tape position at the same time, for then it would contain a cycle, and G does not contain a cycle.

- if a node representing a character at a position which is not affected by the subsequent transition is present in g, then the same character at the same position in the following chunk must also be present in g, in other words, the represented tape contents cannot change outside of the transition function

- each transition node must correspond to a valid transition of M.

One can now construct a list $((q_0, n_0, (c_{0,0}, c_{0,1}, ..., c_{0,p(n)}))..., (q_{p(n)}, n_{p(n)}, (c_{p(n),0}, c_{p(n),1}, ..., c_{p(n),p(n)})))$ according the constructs which the nodes in g represent which will constitute a valid progression of machine, head, and tape states which leads M to accept. Therefore, H will be isomorphic to a subgraph of G only if M accepts the input represented by G.

H will be isomorphic to a subgraph of G if and only if M accepts the input represented by G. Because H and G can be constructed for a given input in polynomial time for any nondeterministic polynomial time decider which is to be modeled, this conversion serves as a polynomial time reduction from any language in NP to SIP. Therefore, SIP is NP-hard.

SIP is NP-easy because there exists a nondeterministic polynomial time algorithm which can verify the language:

- nondeterministically assign each node in G to a node in H

- for each edge e in G, reject if e does not correspond to an edge in H

- for each edge e in H, reject if e does not correspond to an edge in G accept

SIP is NP-complete because it is NP-easy and NP-hard.