Practical Limits of Computation
CS142 Unit 3 Assignment

Fall Semester 2022

1 Warm Up (#Complexity, #Logic)

On the previous assignment you showed that combining decidable languages with regular operations preserved decidability. How about complexiety class?

1.1 Show that P is closed under union, concatenation, and complement.

Let's first define the class P. From the formal definition in Sipser we have that: **P class** The class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine.

Union

To prove this we can use proof by construction.

Let's assume we have two languages L_1 and L_2 which belong to the class P so $L_1 \in P$ and $L_2 \in P$. Since they are in P, then it means that they are decided in polynomial time by the two TMs T_1 and T_2 respectively. Now we can construct a new TM T' that will decide the language $L' = L_1 \cup L_2$.

T' = "On input w:

1 Run T_1 on w (check if $w \in L_1$). If T_1 accepts, accept. Otherwise, move to next step.

2 Run T_2 on w (check if $w \in L_2$). If T_2 accepts, accept. Otherwise, move to next step.

3 Reject since both T_1 and T_2 rejected w."

This construction works because we are using T_1 and T_2 as subroutines of T' and we know that both of them terminate (decide) in polynomial time, which means that T' also decides and terminates in polynomial time. We can express it as a sum of two polynomial times: $O(T_1 + T_2) = O(n^k) + O(m^p)$, which is still a polynomial time.

Concatenation

Let's follow a similar logic. Let's assume we have two decidable languages L_1 and L_2, both $\in P$, that are decided by the two TMs T_1 and T_2 respectively. Now we can construct a new TM T' that will decide the language $L' = L_1 \circ L_2$.

T' = "On input w:

1 For $i = 0, 1, 2 \ldots$

2 Split w into $w_{part1} = w_1 \ldots w_i$ and $w_{part2} = w_{i+1} \ldots w_n$, where n is the length of the input string.

3 Run T_1 on w_{part1}. If T_1 rejects, reject. If it accepts, go to next step.

4 Run T_2 on w_{part2}. If T_2 rejects, reject. If it accepts, go to next step.

5 Otherwise, accept."

The logic behind this is that we check all possible splits of the input string w. Then we check if the first sub-string is part of L_1 and check if the second sub-string is part of L_2. If both of those conditions hold true, then we accept. The overall time is still polynomial since it is the sum of the two polynomial times of T_1 and T_2 for all possible splits of the string w. We

can express it as $n[O(m^k) + O(r^q)]$, where n is the number of all possible splits of w and the brackets represent the sum of the polynomial time that T_1 and T_2 take.

Complement

Let's assume a language $L \in P$. This means that there exists a TM T that decides L in polynomial time. The complement of the language L, is $\overline{L}$ - the set of all strings that do not belong in L. Now let's construct a TM T' for the complement

T' = "On input w, where w is some string

1 Run T on the string w.

2 If T accepts, reject. If T rejects, accept.

It will reject everything accepted by T, while also accepting everything rejected by T. Step 1 uses T as a subroutine and we know that T decides the language L in polynomial time $O(n^k)$. Step 2 takes constant time $O(1)$. Therefore, T' takes $O(n^k) + O(1) = O(n^k)$, which is still polynomial time.

1.2 Show that NP is closed under union and concatenation.

Again let's first find the definition of the class NP from Sipser. There are two possible definitions.

NP class The class of languages that have polynomial time verifiers.

NP class A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

Union

We follow a similar strategy as before.

Let's assume we have two languages L_1 and L_2 which belong to the class NP so $L_1 \in NP$ and $L_2 \in NP$. Since they are in NP, then it means that they are decided in polynomial time by the two NTMs T_1 and T_2 respectively. Now we can construct a new NTM T' that will decide the language $L' = L_1 \cup L_2$.

T' = "On input w:

1 Run T_1 on w (check if $w \in L_1$). If T_1 accepts, accept. Otherwise, move to next step.

2 Run T_2 on w (check if $w \in L_2$). If T_2 accepts, accept. Otherwise, move to next step.

3 Reject (since both T_1 and T_2 rejected w)."

Similar as before, the new NTM simply uses the existing NTMs T_1 and T_2 as subroutines by running them once in steps 1 and 2 respectively. So the it runs in $O(T_1 + T_2)$.

Concatenation

Let's follow a similar logic as before. Let's assume we have two decidable languages L_1 and L_2, both $\in NP$, that are decided by the two NTMs T_1 and T_2 respectively. Now we can construct a new NTM T' that will decide the language $L' = L_1 \circ L_2$.

T' = "On input w:

1 Nondeterministically split w into two parts $w = w_{part1}w_{part2}$

2 Run T_1 on w_{part1}. If T_1 rejects, reject. If it accepts, go to next step.

3 Run T_2 on w_{part2}. If T_2 rejects, reject. If it accepts, go to next step.

4 Otherwise, accept."

All the possible splits of an input string w are finite. Let's say their number is n. For each of them we simply run the two NTMs we know of T_1 and T_2. This means that T' is also a polynomial NTM.

1.3 Do we know whether NP is closed under complement? Why or why not?

The short answer is that we do not know.

From the textbook we know that there is another set of languages $coNP$ with the languages that are complements of languages in NP. We are also told that we do not know if the two languages are the same. If $P = NP$, then the answer would have to be that $coNP = NP$ since P is closed under complement.

The intuition behind why we do not know whether NP is closed under complement is that it is much easier to check if something is present than having to verify that it is absent.

Let's show this with an example. From the textbook we know the language $CLIQUE$ is in NP. We can verify that there exists some k-clique in a graph by trying out different k-clique combinations. But how do we verify that there are no k-cliques that exist ($\overline{CLIQUE}$)? We could check ALL combinations of nodes for a k-clique but that number could grow exponentially large.

For $CLIQUE$ the certificate (hint) we give the verifier is a k-clique to check. But what would the (polynomial size) certificate be for $\overline{CLIQUE}$ be? As already said one option is a list of ALL possible combinations but we are not guaranteed that its size is polynomial.

1.4 (Optional) Show that P is closed under the star operation. (Hint: Use dynamic programming. On input $y = y_1 \dots y_n$ with $y_i \in \Sigma$, build a table that checks whether the substring $y_i \dots y_j \in A^*$ for $i \leq j$.)

2 Building Puzzles #Complexity, #Computability, #Machines

This problem is an analysis of a single-player puzzle game played on an undirected graph G called *Water-Seeker*. Each node of G either contains a pure, untapped water source or is empty. The player must open or mark each node as a water source. If the player opens an empty node, the player learns the number of neighboring nodes that are sources. If the player opens a water source node it is contaminated. The player wins if and when all the sources have been identified without contaminating any of the water.

In the puzzle building problem, you are given a graph G along with numbers labeling some of its nodes. You must determine whether G can be expanded into a consistent Water-Seeker game

board. In other words, can the remaining nodes of G be labled so that the result is a valid, playable game?

2.1 Formulate the puzzle building problem as a language WS.

$$WS = \Big\{ \langle G, F \rangle | G \text{ is a graph with nodes and edges } (N, E)$$
$$\text{and there exists a combination of labels that makes it a valid configuration, meaning}$$
$$\text{there is a placement of water sources on the nodes such that all node labels are consistent, and}$$
$$F \text{ is a function used for labeling the nodes } N \to \{X, 0, 1, 2, \dots\} \Big\}$$

I assumed that we need to also pass a function that knows how to produce the labels, marking water sources with X and the remaining nodes with integers depending on their neighbors. It also makes it easier to convert WS to SAT when labeling the nodes in the following parts.

2.2 Determine whether the following graphs are in WS.

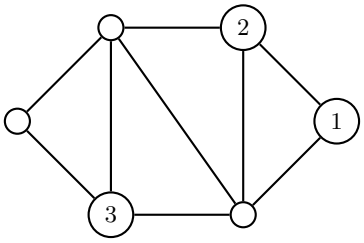


Figure 1: G_1

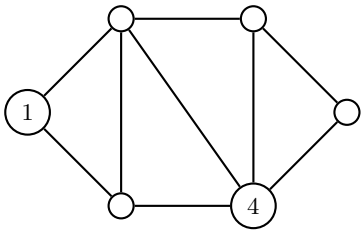


Figure 2: G_2

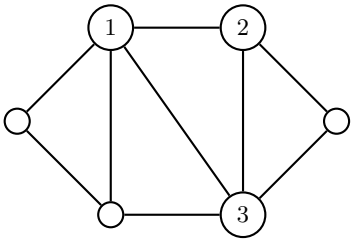
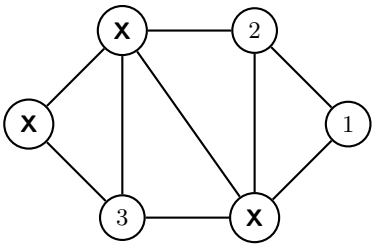


Figure 3: G_3

To show that they are in WS we just need to show that there is a configuration of labels that works so that all labels are consitent. For graph G_1 we have:



This is a consistent combination of labels.

If we now take a look at G_2 we can deduce that there is no concistent combination of labels. The node with label 4 has exactly four neighbors meaning that all of them need to be water sources. However, that leads two two water source neighboring nodes to the node with label 1, so a combination that works is impossible. To make this proof stronger we should consider all possible iterations of label combinations and show that not a single one of them works.

But for the purposes of this assignment we can use our reasoning and logically show that this is the case.

Let's take a look at graph G_3. Following a similar logic, let's look at the node with label 3. It has four neighbors in total. Two of them are not water sources as given by the graph. So it can have a maximum of two water source neighboring nodes. Therefore, a consistent combination of labels does not exist. Again a better approach is to show all possible combinations of labels and that not a single one is consistent.

2.3 Show that WS is decidable.

To show that WS is decidable we need to construct a Turing machine that decides it. The language WS is in NP and to show that we can construct an NTM that non-deterministically decides it in polynomial time.

To determine if an incompletely labeled graph G can be valid, we can make guess random positions for the water sources and then verify that after completing the labels the numbers we want were produced.

T' = "On input $G, F\rangle$:

1 Make a copy of the graph G and call it G'.

2 Nondeterministically assign water sources to the nodes of graph G' and use the labeling function F to complete the rest of the labels.

3 Iterate over the nodes of G and compare them with the labels on G'. If all existing labels on G match with the ones produced on G', accept. Otherwise, reject."

This NTM clearly runs in polynomial time. I assume that it is much smarter than the NTMs we have seen so far. In step 1 it needs to copy the graph which should take polynomial time since it is iterating over a list of vertexes and edges and copying them (at the end of the tape or on another tape for example). Then, step 2 runs in polynomial time too since there is a finite list of nodes that we go through once. The same is valid for step 3 and step 4 takes constant time.

2.4 Show that WS is NP-complete.

The general strategy to show that language B is NP-complete is to show that it is in NP and that all NP-problems are polynomial time reducible to it.

Theorem 7.36 from Sipser: If B is NP-complete and $B <=_p C$ (polynomial time reduction) for C in NP, then C is NP-complete.

Usually we show a polynomial time reduction from 3SAT to the other language (WS in this case).

$$3SAT = \{\langle\phi\rangle | \phi \text{ is a satisfiable 3-CNF Boolean formula}\}$$

I will follow the gadget-style proofs from the textbook to translate a Boolean formula into a Water source problem graph.

First, for every variable (literal) x_i that appears in the formula create nodes for x_i and its negation $\neg x_i$. Since they are mutually exclusive, either one or the other is true, we can connect them with a third node x_i^c whose label will always be a 1 denoting that either x_i or $\neg x_i$ is a water source. Let's use the notation with the labeling function from earlier we have that $F(x_i^c) = 1$.

Second, for every clause c_i in the formula we will make an individual node. Let's label it with a 3 $F(c_i) = 3$. We will connect it with edges to the nodes of the three literals the clause contains. We know that for a clause to be true it means that at least one of the literals needs to be true since a clause is composed of three literals connected by disjunctions. So to be able to satisfy the label of 3 (even if only one or two of the literals are true) we will create two nodes c_i^1 and c_i^2 that are empty but can be labeled with a water source if needed. So for example, if no literal is true then the node of the clause is never satisfiable If all literals are true, then the condition is satisfied. If a single literal is true, then we make $F(c_i^1) = X$. If two literals are true, then we add $F(c_i^2) = X$.

This concludes our proof by reduction. It is a valid because for a formula to be satisfiable it means that there needs to be a combination of variables such that each clause is true. For a clause to be true it means that at least one of its three literals is true. We have expressed that structure of the formula in the WS graph with the variable and clause gadgets we created above.

Let's demonstrate how this works with an example. I took a formula we used in Session 22:

$$\phi = (x \vee x \vee z) \wedge (x \vee y \vee y) \wedge (y \vee y \vee z)$$

After following the construction this is how the graph looks like:

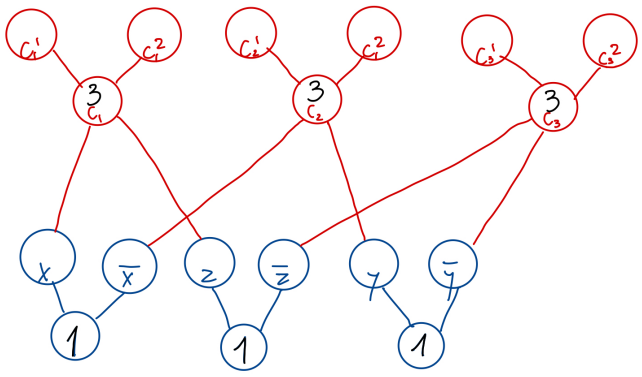


Figure 4: Construction of graph

It is a valid graph in WS which means it is a satisfiable formula. These are the two ways to label the nodes to make sure all labels are consistent.

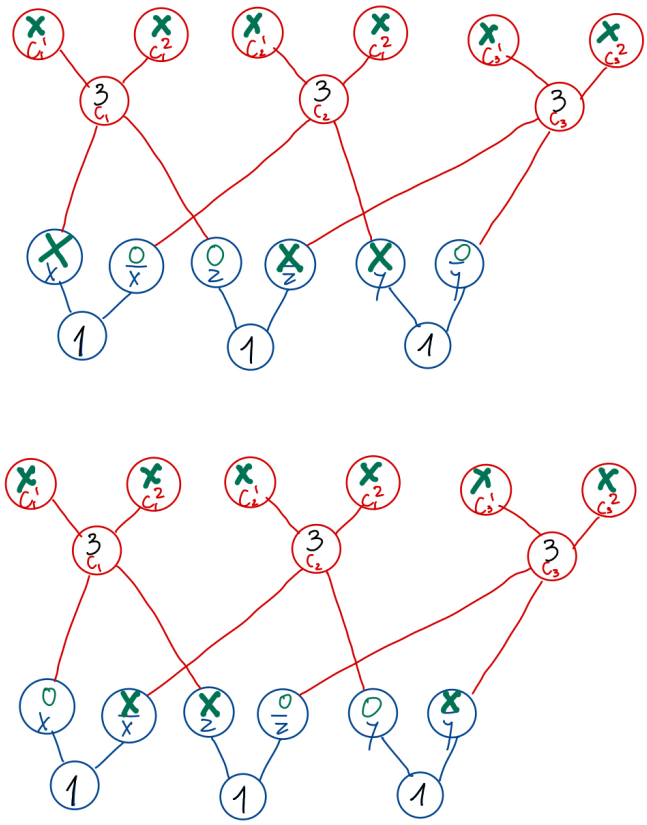


Figure 5: Consistent graph labels

We got the same conclusion as in session 22. The truth value of the variables can be either $x = 1, y = 1, z = 0$ or $x = 0, z = 1, y = 0$. As we can see from the graph the truth value is established depending on which nodes of literals are water sources.

We also need to show that the procedure translating a formula ϕ into a graph defines a polynomial time reduction.

Let's say there are n literals in the formula (size of input)

First we define the nodes of the graph based on the literals in each clause plus defining a meta clause structure for the graph, which takes $O(n)$. Then for each node, check each other node and draw an edge if the following rules are satisfied .... $O(n^2)$:

1 no edge is present b/n nodes in the same triple (clause)

2 no edge is present b/n two nodes with contradictory labels

3 connect each clause node to its three literals and two supporting nodes

4 connect each pair for a variable to its joint middle node with label 1

In the end we have $O(n) + O(n^2)$, which is polynomial, indicating that the procedure is a polynomial time reduction.

3 Location-based Analysis (#Complexity)

How might a resolution to P vs NP change the world? Computer Scientists have theorized multiple possible "worlds" corresponding to different potential outcomes [Fortnow 2022].

- *Algorithmica* ($P = NP$) NP problems can be solved in polynomial time.
- *Heuristica* ($P \neq NP$) While some cases remain hard, the average time needed to solve an NP problem is polynomial.
- *Pessiland* ($P \neq NP$) Everything is hard. The average time needed to solve NP hard problems is astronomical. In this world we are no longer able to use public-key cryptography.
- *Cryptomania* ($P \neq NP$) In this world NP problems remain hard to solve, but we have the ability to use public-key cryptography.

Our current world seems to closely resemble Cryptomania, but ¯_(ツ)_/¯? If we live in one of the other worlds suddenly, how might that affect the people living around you?

3.1 Choose a technology that is relevant to your current city. It could be an web app used by many local residents (e.g., Moovit in San Francisco), or a tech start-up head quartered in the city. Write a 200 summary of the technology you chose.

Outside of the what I think is the most widespread NP problem in Buenos Aires - public transport scheduling, I decided to dig deeper into the security that is implemented by the government. One of the most popular technologies is the Federal Biometric Identification System for Security (SIBIOS). Its puprpose is to establish public safety and help identify people after criminal acts with the help of biometric systems. Its main use is in Trace processing, specifically when an expert obtains images of fingerprints left at a crime scene. All information is stored in a system called AFIS. So the expert can upload these images and identify possible matches that can be used during the investigation. It is important to note that often the images of fingerprints found at a crime scene are only partial and not of perfect quality.

Usually in fingerprint matching, pattern matching is used where an algorithm looks for patterns in data. In this case it looks for duplicates of the initial fingerprints (King 2022). Pattern matching is one of the applications of Graph coloring, a special case of graph labeling ("Graph Coloring", 2022), which is an NP-complete problem.

3.2 Chose two of the theorized worlds *other than Cryptomania* to explore. How would suddenly transitioning into the other world affect the technology you chose? Write an $\tilde{2}$00-world analysis for each.

The first world I chose was Algorithmica where $P = NP$. If that was the case, then this problem would be solvable in polynomial time. This world has huge implications for everything in general. Strictly speaking about the SIBIOS technology it would that from this unreliable

technology it would become much more accurate in finding existing or potential fingerprint matches. As a result public safety would increase as it will be easier for experts and the police to find potential suspects and solve crimes faster and with higher success rate. Right now the technology is heavily criticized that it is not reliable in protecting citizes but those critiques will be minimal in this world. The world Heuristica would have similar implications where we would be able to use that technology more successfully but not always.

The second world I chose was Pessiland. Since the average time to solve this problem of finding potential fingerprint duplicates is astronomical, then it means that the technology will not be useful in identifying potential suspect and therefore will not aid in crime solving much. It simply will not be able to do so in short enough time for us to be realistic. The Argentinian government (and the rest of the world) will have to rely on other methods to solve crimes or other methods to compare and use fingerprints as evidence.

4 Reflection

4.1 Consider one idea from the course that you have found beautiful, and explain why it is beautiful to you. Your answer should: (1) explain the idea in a way that could be understood by a classmate who has taken classes X and Y but has not yet taken this class and (2) address how this beauty is similar to or different from other kinds of beauty that human beings encounter.

I find the problem of P vs NP absolutely fascinating! I have already shared this with a bunch of my friends who have not taken CS142 and how it is connected to their fields if interest, like biology and protein synthesis to prevent cancer.

This is an explanation to someone who has not taken any class in particular. P is a group of problems that are relatively easy and we know how to solve them efficiently. On the other hand, NP is a group of what we currently think of as very hard problems. Most of them we do not know how to solve. We can only efficiently check if some solution exists by trying out different answers (backwards) but not actually find that solution forward.

The question is if these two groups of problems are the same or not. If they are $P = NP$, then it would mean that the very hard problems have very easy solutions and that every problem in NP that can be quickly verified can be quickly solved too.

I think at first it is so hard to comprehend your mind around this well that it is fascinating and it seems unreal, like something hidden that you just uncovered a new part of your reality. I would compare it to falling in love with your childhood friend who you never saw in that way. You do not want to admit it at first when it happens but it makes sense and it is fascinating.

4.2 Give two examples HCs you applied while solving this assignment. How did you apply the HC? Was the application sucessful? If yes, why? If not, how could you improve your application?

#networks

This application is not focused on a complex system but rather on Boolean formulas and how they can also be transformed and analyzed as graphs. I applied it in problem 2.4 where we to show that WS is NP-complete I created a reduction from $3SAT$. To construct the reduction

I analyzed how the different components (literals and clauses) of a boolean formula work and interact with each other as well as what it means for the formula to be satisfiable. I modeled this with gadgets for the two components and used edges to connect each clause with its literals, based on the disjunctive connectors of the formula. I analyzed how the WS graphs work and when the network of nodes would work in such a way so that all labels actually work with the number and location of water sources. I made a similar application on the last assignment with a PDA. What I pointed out then was that usually when we talk about #networks we reference aspects like connectivity and dispersity. The beauty of this reduction is in the fact that it is generalizable. No matter how many literals, clauses and subsequently nodes there are it will always be a valid way to convert from $3SAT$ to WS in polynomial time.

#heuristics

I applied this HC while solving the whole problem set. My rule of thumb is reading the problem, thinking about an approach for 5 to 10 minutes, and then referencing the relevant parts of the textbook to identify theorems or existing proofs that can help me find the right direction to the solution. It always saves me time and keeps me on track because I do not need to reinvent the wheel. I build up all of my proofs from prior existing knowledge from a text that I have already read for class. For example, when solving the reducability problem in 2.2 I immediately used this heuristic to find the relevant theorem. I quoted it and it gave me a hint to use $3SAT$. I referenced the existing proof in the textbook for $CLIQUE$ and saw the gadget-style proof, which I knew I could adapt and use in this scenario. The main heuristic that plays a role in this initial process before writing the proof is the recognition heuristic. I skim the text from the textbook and identify potential hints based on information I have seen before (in memory), which allows me navigate the solutions better.

5 Sources

Fortnow, L. (2022) "Fifty Years of P vs. NP and the Possibility of the Impossible," Communications of the ACM, 65(1), pp. 76–85. Available at: https://doi.org/10.1145/3460351.

"Graph Coloring." Wikipedia, 7 Dec. 2022, https://en.wikipedia.org/wiki/Graph_coloring#Applications. Accessed 10 Dec. 2022.

King, Rawlson. "Explainer: Fingerprint Matching — Biometric Update." Www.biometricupdate.com, 5 Dec. 2015, www.biometricupdate.com/201512/ . Accessed 10 Dec. 2022.