ProjectCSE105F24: Project

CSE105F24

Due December 11, 2024 at 11am

The CSE 105 project is designed for you to go deeper and extend your work on assignments and to see how some of the abstract notions we discuss can be implemented in concrete ways. The project is an individual assignment and has two tasks:

Task 1: Illustrating the decidability of a computational problem, and

Task 2: Illustrating a mapping reduction

What resources can you use? This project must be completed individually, without any help from other people, including the course staff (other than logistics support if you get stuck with screencast). You can use any of this quarter's CSE 105 offering (notes, readings, class videos, homework feedback). Tools for drawing state diagrams (like Flap.js and JFLAP and the PrairieLearn automata library) can be used to help draw the diagrams in the project too.

These resources should be more than enough. If you are struggling to get started and want to look elsewhere online, you must acknowledge this by listing and citing any resources you consult (even if you do not explicitly quote them), including any large-language model style resources (ChatGPT, Bard, Co-Pilot, etc.). Link directly to them and include the name of the author / video creator, any and all search strings or prompts you used, and the reason you consulted this reference. The work you submit for the project needs to be your own. Again, you shouldn't need to look anywhere other than this quarter's material and doing so may result in definitions that conflict with our conventions in this class so think carefully before you go down this path.

If you get stuck on any part of the project, we encourage you to focus on communicating what you think the question might mean, including bringing an example from class or homework you think might be relevant, and include any submission any aspect where you're unsure. Clear communication about these theoretical ideas and their applications is one of the main goals of the project.

Submitting the project You will submit a PDF plus a video file for the first task and a PDF plus a video fiile for the second task. All file submissions will be in Gradescope.

Your video: You may produce screencasts with any software you choose. One option is to record yourself with Zoom; a tutorial on how to use Zoom to record a screencast (courtesy of Prof. Joe Politz) is here:

`https://drive.google.com/open?id=1KROMAQuTCk4OzwrEFotlYSJJQdcG_GUU`.

The video that was produced from that recording session in Zoom is here:

`https://drive.google.com/open?id=1MxJN6CQcXqIbOekDYMxjh7mTt1TyRVMl`

Please send an email to the instructors (minnes@ucsd.edu) if you have concerns about the video / screencast components of this project or cannot complete projects in this style for some reason.

Reference definitions for computational problems from Section 4.1:

Acceptance problem		
... for DFA	A_{DFA}	$\{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$
... for NFA	A_{NFA}	$\{\langle B, w\rangle \mid B$ is a NFA that accepts input string $w\}$
... for regular expressions	A_{REX}	$\{\langle R, w\rangle \mid R$ is a regular expression that generates input string $w\}$
... for CFG	A_{CFG}	$\{\langle G, w\rangle \mid G$ is a context-free grammar that generates input string $w\}$
... for PDA	A_{PDA}	$\{\langle B, w\rangle \mid B$ is a PDA that accepts input string $w\}$
Language emptiness testing		
... for DFA	E_{DFA}	$\{\langle A\rangle \mid A$ is a DFA and $L(A) = \emptyset\}$
... for NFA	E_{NFA}	$\{\langle A\rangle \mid A$ is a NFA and $L(A) = \emptyset\}$
... for regular expressions	E_{REX}	$\{\langle R\rangle \mid R$ is a regular expression and $L(R) = \emptyset\}$
... for CFG	E_{CFG}	$\{\langle G\rangle \mid G$ is a context-free grammar and $L(G) = \emptyset\}$
... for PDA	E_{PDA}	$\{\langle A\rangle \mid A$ is a PDA and $L(A) = \emptyset\}$
Language equality testing		
... for DFA	EQ_{DFA}	$\{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$
... for NFA	EQ_{NFA}	$\{\langle A, B\rangle \mid A$ and B are NFAs and $L(A) = L(B)\}$
... for regular expressions	EQ_{REX}	$\{\langle R, R'\rangle \mid R$ and R' are regular expressions and $L(R) = L(R')\}$
... for CFG	EQ_{CFG}	$\{\langle G, G'\rangle \mid G$ and G' are CFGs and $L(G) = L(G')\}$
... for PDA	EQ_{PDA}	$\{\langle A, B\rangle \mid A$ and B are PDAs and $L(A) = L(B)\}$

Task 1: Illustrating the decidability of a computational problem Many computational problems are decidable, sometimes using beautiful algorithms. In this part of the project, you'll choose a **decidable** computational problem, and demonstrate the proof that it is decidable by building a program in a programming language of your choice (aka a high-level description of a Turing machine) that decides it. You will then demonstrate how your construction works for some test examples.

Specifically:

1. Choose a decidable computational problem from Section 4.1. *Note: if you'd like to consider a different computational problem instead, please check with Prof. Minnes first. You must do so no later than the start of Week 9.*

2. Write a program in Java, Python, JavaScript, C++ , or another programming language of your choosing that decides this computational problem. The function input must be a **string** and part of your work in this program is to design string representations for arbitrary instances of the model of computation the computational problem you picked is about (e.g. DFA, NFA, regular expressions, CFG, or NFA). The function output must be a **boolean** true (if the string is in the set representing the computational problem) or false (if the string is not in the set representing the computational problem).

    - You may use our class notes and the textbook for ideas on the algorithm that your program will implement.
    - If you would like, you may use aids such as co-pilot or ChatGPT to help you write this program. However, you should test the code that is produced and be able to explain what it is doing. Your code needs to be well-organized and well-documented. As a header in your code file, include a comment block describing any resources that were used to help generate your code, including any and all prompts used in interactions with LLM coding tools.

3. To demonstrate your program, select one string that is in the set representing the computational problem, and one string that is not in the set representing the computational problem, explain why these strings are valid examples, and demonstrate running your program on each to get the appropriate output.

Presenting your reasoning and demonstrating it via screenshare are important skills that also show us a lot of your learning. Getting practice with this style of presentation is a good thing for you to learn in general and a rich way for us to assess your skills. To demonstrate your work, you will create a 3-5 minute screencast video explaining your code design and demonstrating its functionality.

Checklist for submission For this task, you will submit a PDF plus a video file.

(PDF) Writeup includes a clear specification of computational problem being decided.

(PDF) Documentation for program deciding this computational problem: include a description of how input strings are parsed to represent instances of the computational problem.

(PDF) Clear specification of two example strings, explaining which is in the set (and why) and which is not in the set (and why not).

(PDF) Project submission includes a printout of code for program implementing algorithm to decide the computational problem, as well as screen shots demonstrating running your program on your example strings.

(PDF) Project writeup is typed or clearly hand drawn with precise language and notation for all terms.

(Video) Start with your face and your student ID visible for a few seconds at the beginning, and introduce yourself audibly while on screen. You don't have to be on camera for the rest of the video, though it's fine if you are. We are looking for a brief confirmation that it's you creating the video and doing the work you submitted.

(Video) Present the computational problem you will be working with, and example strings that you will be using, including explanations of why you chose this problem and these strings (and why one of the strings is in the set and why the other is not).

(Video) Show on the screen and explain the code for your program, including the software design choices you made (e.g. which data structures are you using, etc.) and any resources you used. The video should clearly describe which programming language was chosen for the implementation and gives the reasons why.

(Video) Demonstrate running your code on each of your example inputs. The video should include screencasts of running the code live. Explain why the output of your program is what you would expect, by connecting the output of the the definition of the computational problem and your chosen parsing of input strings.

(Video) Logistics: video needs to load correctly, be between 3 and 5 minutes, show your face and ID, and you introduce yourself audibly while on screen.

Note: Clarity and brevity are both important aspects of your video. In previous years, we've seen students speed up their videos to get below the 5 minute upper bound. This is ok so long as it doesn't compromise clarity. If the graders need to slow your video down to understand it, it may not earn full credit.

Task 2: Illustrating a mapping reduction We can use mapping reductions to prove that interesting computational problems are undecidable, building on the undecidability of other computational problems. In this part of the project, you'll choose a specific **mapping reduction** and implement a computable function that witnesses it using a programming language of your choice (aka a high-level description of a Turing machine that computes it). You will then demonstrate how your construction works for some test examples.

Specifically:

1. Choose a mapping reduction we discussed in class or in the homework or in review quizzes or in the textbook where both sets being compared are undecidable. *Note: if you'd like to consider a mapping reduction we have not discussed instead, please check with Prof. Minnes first. You must do so no later than the start of Week 9.*

2. Write a program in Java, Python, JavaScript, C++ , or another programming language of your choosing that implements a computable function witnessing this mapping reduction. The function input must be a **string** and the function output must be a **string**. Part of your work in this program is to design string representations for arbitrary instances of the model of computation the computational problems being compared in the mapping reduction. Your function will need to be able to process *any* string as input.

    - You may use our class notes and the textbook for ideas on the algorithm that your program will implement.
    - If you would like, you may use aids such as co-pilot or ChatGPT to help you write this program. However, you should test the code that is produced and be able to explain what it is doing. Your code needs to be well-organized and well-documented. As a header in your code file, include a comment block describing any resources that were used to help generate your code, including any and all prompts used in interactions with LLM coding tools.

3. To demonstrate your program, you will need to run it for an example positive and negative instance. That is to say, if you are implementing a computable function witnessing $X \leq_m Y$, you will select one string that is in X and one string that is not in X, and you will demonstrate running your program on each of these strings and explain why the output of the function is good.

Presenting your reasoning and demonstrating it via screenshare are important skills that also show us a lot of your learning. Getting practice with this style of presentation is a good thing for you to learn in general and a rich way for us to assess your skills. To demonstrate your work, you will create a 3-5 minute screencast video explaining your code design and demonstrating its functionality.

Checklist for submission For this task, you will submit a PDF plus a video file.

(PDF) Writeup includes a clear specification of mapping reduction being witnessed, and both sets in the reduction are undecidable.

(PDF) Documentation for program computing the function witnessing this mapping reduction: include a description of how input strings are parsed and how output strings correspond to input strings.

(PDF) Clear specification of two example strings, explaining which is is a positive instance (and why) and which is a negative instance (and why not).

(PDF) Project submission includes a printout of code for program computing the function witnessing the mapping reduction, as well as screen shots demonstrating running your program on your example strings.

(PDF) Project writeup is typed or clearly hand drawn with precise language and notation for all terms.

(Video) Start with your face and your student ID visible for a few seconds at the beginning, and introduce yourself audibly while on screen. You don't have to be on camera for the rest of the video, though it's fine if you are. We are looking for a brief confirmation that it's you creating the video and doing the work you submitted.

(Video) Present the mapping reduction you will be working with, and example strings that you will be using, including explanations of why you chose this reduction and these strings (and why one of the strings is a positive instance and the other is a negative instance).

(Video) Show on the screen and explain the code for your program, including the software design choices you made (e.g. which data structures are you using, etc.) and any resources you used. The video should clearly describe which programming language was chosen for the implementation and gives the reasons why.

(Video) Demonstrate running your code on each of your example inputs. The video should include screencasts of running the code live. Explain why the output of your program is what you would expect, by connecting the output of the program to the definition of the mapping reduction and your chosen parsing of input strings.

(Video) Logistics: video needs to load correctly, be between 3 and 5 minutes, show your face and ID, and you introduce yourself audibly while on screen.

Note: Clarity and brevity are both important aspects of your video. In previous years, we've seen students speed up their videos to get below the 5 minute upper bound. This is ok so long as it doesn't compromise clarity. If the graders need to slow your video down to understand it, it may not earn full credit.