EECS 118
Knowledge Engineering and Software Engineering
Fall 2019

Term Project - Part 1
Symbolic Geometry Problem Solver
Assigned on 10/23

In this project, you will create a geometry problem solver and complete a web interface for the TAs' programs to call. You are required to identify symbolic problems (detailed below) for each geometry diagram and use Python to create a solver program to solve the problems. You also need to test your program and test a program of someone else (to be assigned) with a set of test cases. This is a team project of two people; only one team member should submit the work. The due date is Thursday, December 5, 11:59 PM.

Deliverables:

Labeled diagrams, test report including test plan (cases) and test results, peer review report including test plan (cases) and test results, and the source code of your project.

Diagram Assignment:

Your will be randomly assigned one or more diagrams based on their difficulties.

Requirements:

1. Label all existing angles, edges, areas, line segments, etc. with alphanumeric labels. e.g., a1, e3, ar5
2. Identify all problems that can be described with at most three geometric predicates (detailed below) for each geometry diagram assigned to you. This means all problems having one, two, or three different geometric predicates. Predicates with the same predicate symbol but different parameters are considered different. You do not need to add auxiliary lines and create new variables.
3. Write a solver program for each problem (diagram) assigned to you. The solver program should solve the problems identified in (2).
4. The program should, for each possible problem (described in terms of the predicates explained in 2), return the output as another set of predicates. If the problem cannot be solved by the program, your program should return NULL as the output.
5. You will complete separate tasks assigned by the TAs in order to provide an API.

- Input & output:
  - The user inputs and outputs should include, but are not limited to, any valid combination of three of the following predicates:

Geometric Predicates:

- set_parallel(name1, name2): true if two line segments name1 and name2 are parallel.
- set_ perpendicular(name1, name2): true if two line segments name1 and name2 are perpendicular.
- set_equal(name1, name2): true if two angles, line segments, or areas name1 and name2 are equal.
- set_fraction(name1, name2, fraction): true if two angles, line segments, or areas name1 and name2 satisfy the relationship name1 = fraction * name2
- set_sum_value(name1, name2, sum): true if two angles, line segments, or areas name1 and name2 satisfy the relationship name1 + name2 = sum.
- set_similar(name1, name2): true if two shapes are similar.
- set_congruent(name1, name2): true if two shapes are congruent.
- set_tan(name1, name2): true if for a line segment name1 tangent to a circle name2.

Output methods for the solver:

- get_all(): returns a dictionary containing all resulted relations in the following format: {“parallel”: [name1, name2], “equal”: [name1, name2], “fraction”: [name1, name2, fraction]…}. The terms “parallel”, “equal”, “fraction”… are from the second part of the name of each input method. The list after each colon is similar to the parameters of each input method. It should return the string “null” if the requested term is not solvable.

The solver program should have a basic text-based UI (Python console) that asks the user for input and displays the output. Email the TAs if you need additional input functions to complete the solver.

Peer Review and Test Plans:

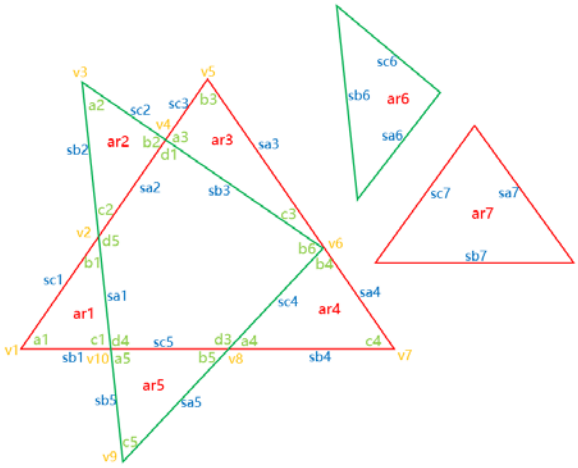
After turning in your work, you will be randomly assigned to review another student’s project. During this process, you should determine the completeness, strengths, and weaknesses of the reviewee’s program. You need to develop a thorough test plan for your peer’s problem solver and include the test results in your peer review report. You also need to write a test report on your own project.

Grading:

1. You will receive full credit if your solver program can handle all combinations of up to three predicates and give the correct output.
2. Points will be deducted if a problem cannot be solved correctly.
3. Up to 20% extra credit will be given if your solver can handle combinations of more than three different predicates.

Example:

The picture on the right shows one way to label the diagram (You do not need to label the vertices).



If the inputs are:

set_parallel(sb6, sa7)
set_equal(b1, d5)

Since b1 = d5, and b1 + d5 = 180 degrees, we know b1 = 90 degrees. As sb6 and sa7 are parallel, we know b1 = b3. Therefore, sc7 and sa7 are perpendicular, and a1 + c4 = 90 degrees.

Your program should at least provide the output:
{“perpendicular”: [“sc7”, “sa7”], “sum_value”: [“a1”, “c4”, 90]}

This is only a demonstration of format, your solver should find all possible results. Also, notice that this is a dictionary with string type keys and list type values.