Mohammed Haque (62655407)
Jose Alberto Padilla (91945869)
EECS 118

# TP Geo Peer Review - I137

**Test Plan**

Team 37 were assigned to solve a geometry problem consisting of 2 triangles with 5 intersection points. We will test their solver by using test cases consisting of 3 predicates each. The test cases are meant to test that each function works correctly and that certain boundary cases have been taken into account.

**Test 1:**

In this test case, we set all possible valid sides to be parallel. This will test that the set_parallel function works, and that Team 37 accounted for this possible case.

set_parallel("sc2", "sb1")
set_parallel("sa4", "sb2")
set_parallel("sa3", "sa2")

**Results:**

parallel [('sc7', 'sa6'), ('sc6', 'sb7')]
equal [('a3', 'c4'), ('c1', 'a5'), ('a1', 'b5'), ('b2', 'b3'), ('d1', 'b4'), ('c2', 'c5'), ('c5', 'b1'), ('c3', 'b4'), ('a2', 'c1'), ('a1', 'a4'), ('d1', 'c3'), ('a2', 'a5'), ('c2', 'b1'), ('b2', 'a1'), ('b5', 'a4'), ('b3', 'a4'), ('a1', 'b3'), ('b3', 'b5')]
sum_value [('d3', 'b4', 180), ('a2', 'd4', 180), ('sc5', 'sb1', 'sc3'), ('a1', 'd5', 180), ('d1', 'a3', 'd5'), ('a5', 'd4', 180), ('b5', 'd5', 180), ('sa3', 'sc4', 'sc7'), ('d5', 'a4', 180), ('c1', 'd4', 180), ('d2', 'c5', 180), ('d1', 'd3', 180), ('c2', 'd2', 180), ('d3', 'c3', 180), ('b1', 'd2', 180), ('b3', 'd5', 180)]
similar [('ar2', 'ar6'), ('ar3', 'ar4'), ('ar6', 'ar5'), ('ar1', 'ar5'), ('ar2', 'ar1'), ('ar7', 'ar4')]

**Verdict**

It appears that Team 37 only accounted for up to two sets of parallel sides instead of 3. Nevertheless, the function seems to work properly and the solver was able to determine that this set of predicates would result in sets of similar triangles. The contents of sum_value are also correct. Team 37 even accounted for the parallelograms that would be formed as per the sets of equal angles and equal sides in the equal and sum_value results.

**Test 2:**

This test case focuses on the set_perpendicular function. We will attempt to create 3 valid sets of perpendicular lines. We expect the solver to be able to determine many sum_values and equal angles based on the fact that the inner angles of a triangle add up to 180 degrees.

```
set_perpendicular("sa2", "sb1")
set_perpendicular("sc3", "sa3")
set_perpendicular("sa3", "sa4")
```

**Results:**

```
perpendicular [('sa3', 'sa4'), ('sc3', 'sa3'), ('sa2', 'sb1')]
equal [('c3', 'b4'), ('c1', 'a5'), ('b5', 'a4'), ('c2', 'b1')]
sum_value [('b1', 'd2', 180), ('b5', 'd5', 180), ('d3', 'c3', 180), ('c1', 'd4', 180), ('a5', 'd4', 180), ('a3',
'c3', 90), ('d5', 'a4', 180), ('c2', 'd2', 180), ('d3', 'b4', 180)]
```

**Verdict**

The solver successfully set all 3 sets of line perpendicular to each other. However, it was not able to solve much else. The equal and sum_value results are all the ones that are inherently true to the diagram. The solver was not able to deduce that since one angle of a triangle is 90 degrees, the other two angles should add up to 90 degrees. It also was not able to set the angles on either side of the perpendicular intersection equal to each other.

**Test 3:**

This test case is meant to test that the remaining functions work. We expect to see some perpendicular values due to the set_sum_value function as well as the equality results that derive from that.

```
set_sum_value("a1", "c4", 90)
set_equal("sc2", "sc3")
set_fraction("sb2", "sa3", 2)
```

**Results:**

```
perpendicular [('sa7', 'sc7')]
equal [('b5', 'a4'), ('sc2', 'sc3'), ('c2', 'b1'), ('c3', 'b4'), ('c1', 'a5')]
fraction [('sb2', 'sa3', 2)]
sum_value [('b2', 'a3', 90), ('d5', 'a4', 180), ('c2', 'd2', 180), ('b1', 'd2', 180), ('a5', 'd4', 180), ('b5',
'd5', 180), ('a1', 'c4', 90), ('d3', 'c3', 180), ('d3', 'b4', 180), ('c1', 'd4', 180)]
```

**Verdict**

The solver was able to correctly deduce the perpendicular lines derived from the set_sum_value predicate. The equal and  fraction values also work as expected. It was not able to deduce the sum_values of 90 degrees that derive from the perpendicular results, but this was already noted in the previous test case.

**Test 4:**

In this test case, we attempt to create an invalid triangle by making two of its angles 90 degrees. This should result in a null value being returned.

```
set_perpendicular("sa2", "sb1")
set_perpendicular("sb1", "sa4")
set_sum_value("a1", "c4", 180)
```

**Results:**

```
perpendicular [('sb1', 'sa4'), ('sa2', 'sb1')]
equal [('b5', 'a4'), ('c1', 'a5'), ('c3', 'b4'), ('c2', 'b1')]
sum_value [('d5', 'a4', 180), ('c2', 'd2', 180), ('a5', 'd4', 180), ('d3', 'b4', 180), ('b5', 'd5', 180), ('a1',
'c4', 180), ('c1', 'd4', 180), ('d3', 'c3', 180), ('b1', 'd2', 180)]
```

**Verdict**

It appears that the solver does not deduce that the problem in unsolvable and instead solves what it can. After testing some other cases, it seems like the solver is not able to return a null value at all, or perhaps only in specific cases. Nevertheless, the solver is able to solve what it can from the given predicates. Thus assuming that valid inputs are given, the solver performs quite well.

**Conclusion**

Team 37 has done an overall good job with their solver. Its strengths are in setting parallel lines as it can solve even the parallelograms created by these. Its biggest weakness is not returning null values when appropriate, but this only applies to invaild input. Overall, the solver performs as expected.