Task 4 [Extra Credit]: Modeling Exercise

In order to implement cones into this program we would first have to add a cone(c1) predicate similar to the block but will not be able to have blocks stacked on top of it and can be used with the color and size facts.

color(c1, orange).

size(c1, large).

To add the left and right spaces of the cones we can edit the predicate to left(C1, C2) and right(C2, C1) and this would be also able to keep order of what blocks are next to each other on the table, we can check the order by just checking which blocks are to the left and right of each other.

We might add to the grammar:

noun(n(block)) --> [cone].

adj(a(large)) --> [left].

adj(a(large)) --> [right].

prep(p(of)) → [of].

prep(p(to)) --> [to].

For the reference resolution we would need to incorporate the possibility of a cone while accounting for nothing being able to be stacked on top of it where it then would return false. This would be verPy similar to the original block resolve_ref. Along with this we will need to add the possibility of it being to the left or right.

For hypothetical questions first we will need to add to the grammar verb(v(can)) → [can], then we need to take into account the resolve_ref which will now need to be able to send back true or false without altering the current state, this could possibly be done through the process input where we can preform resolve_ref check if it returns true or false then return to the original state.