**Solution:**

The first thing we need to consider while parsing the SPARQL query is to find out from which point to start. Let’s consider the two following queries:

Q1.

SELECT ?x, ?y, ?z

WHERE ?x <name> “Jack”.

?x <friend> ?y

?y <brother> ?z

?x <home> “Kansas City”

Q2.

SELECT ?X ?Y ?Z

WHERE ?X ?Y ?Z

?X <name> “Jack”

?Z <name> “Adam”

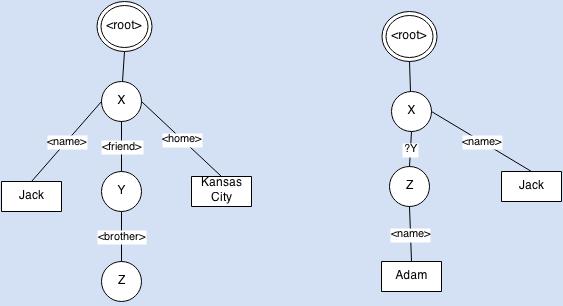


Figure 1: Guide Graph

**Steps:**

1. Create a guide graph (see details below)
2. Sort the query conditions (grey colored lines). To compare the two query statements the following comparison rules will be applied:
   1. Ancestor in guide graph should be processed first
   2. Condition with less variables should be processed first
3. Execute the sorted query conditions
4. For each condition (e.g. Y <brother> Z) we assume that we already have a set of values for Y and replace Y with these values. If Z isn’t found for any Y, then we track back until we reach root and then discard that combination path from root. This operation will guarantee that all the paths from <root> to Z are valid eventually. Suppose in Q1 we found an X who has a friend Y but Y don’t have a brother Z. So we shouldn’t consider that (X,Y) pair at all, hence we remove that combination.
5. When all conditions are executed we just traverse from root to leaf for each (x,y,z) combination.

**Create guide graph:**

1. Create a root node.
2. Read the first condition and create a edge from root to subject and another from subject to object. The value of the edge is the predicate. Also keep track of the variables that have been visited. To uniquely mark each variable in the guide, we can use Dewey numbering scheme.
3. While processing a condition, if we encounter a variable in the object position which has been used previously as an subject, then we transfer that sub-tree under current subject. For example, in Q1, let’s say, Y <brother> Z is processed before X <friend> Y and that there was a edge from root to Y. Now when we find X <friend> Y, we create an edge from X to Y and transfer the Y-edge-Z sub-tree under X-edge-Y.