```
/ Match the 3 major structures
// 0. Ring structure
MATCH ringPath=(beginMol:Molecule)-
[:FORMS*{{MIN_RING_SIZE}}..{{MAX_RING_SIZE}}]->(beginMol:Molecule)
//{{MIN_RING_SIZE}}..{{MAX_RING_SIZE}}
UNWIND nodes(ringPath) as ringMol // Create the iterator ringMol to
iterate over all the molecules in the ringPath, assuring that only the
distinct smiles strings are counted in ringMols (otherwise results will
duplicate the beginMol because the ring query starts and ends at beginMol)
WITH collect(distinct ringMol.smiles_str) AS ringMols, ringPath,
relationships(ringPath) as ringRels, beginMol
// Filter the query by several conditions
WHERE size(ringMols) > 2 // controls number of molecules in the ring
AND size(ringRels) = size(ringMols) // asserts that the number of
molecules must equal the number of relationships in the ring (so the
relationships don't hop molecules more than once and create a collapsed
ring figure 8 looking structure


// 1. Molecule in ring splitting to form the beginMol (autocatalytic
structure)
MATCH autocatPath=(catMolInRing:Molecule)-[autocatRel:FORMS]-
>(beginMol:Molecule)
WHERE catMolInRing.smiles_str IN ringMols
AND NOT autocatRel IN ringRels // prevent the autocatPath from being the
molecule directly related to the beginMol already through the ringPath--
needs to be new relationship
WITH ringMols, ringPath, ringRels, beginMol, catMolInRing, autocatPath,
relationships(autocatPath) as autocatPathRels, nodes(autocatPath) as
autocatPathNodes

// 2. Ring consumer structure
MATCH branchedBeginMolPath=(beginMol)-[:FORMS]-
>(beginMolConsumer:Molecule)
WHERE beginMol <> beginMolConsumer
WITH branchedBeginMolPath, beginMol, beginMolConsumer, ringMols as
ringMols, ringPath as ringPath, ringRels as ringRels, catMolInRing,
autocatPath, autocatPathRels, autocatPathNodes



// 3. Feeder structure
MATCH attachedPath=(feederMol:Molecule)-[:FORMS]-
>(intermediateMol:Molecule)-[:FORMS]->(consumerMol:Molecule)
WITH feederMol, intermediateMol, consumerMol, ringMols as ringMols,
ringPath as ringPath, ringRels as ringRels, beginMol as beginMol,
beginMolConsumer as beginMolConsumer, branchedBeginMolPath as
branchedBeginMolPath, attachedPath as attachedPath, catMolInRing,
autocatPath, autocatPathRels, autocatPathNodes
WHERE NOT beginMolConsumer.smiles_str IN ringMols
AND beginMolConsumer <> feederMol
AND beginMol <> feederMol
AND feederMol <> consumerMol
AND beginMol <> consumerMol
```

```
AND NOT feederMol.smiles_str IN ringMols
AND NOT consumerMol.smiles_str IN ringMols
AND intermediateMol.smiles_str IN ringMols
AND beginMolConsumer <> consumerMol
AND intermediateMol <> beginMol
AND NOT (beginMol:Molecule)-[:FORMS]->(beginMol:Molecule)<-[:FORMS]-
(beginMol:Molecule) // Assert that the relationships in the ringPath must
travel all in the same direction
// control the generation range at which the cycle is formed by assuming
it can't exist until after the feederMol is formed
{{COMMENT_OUT_FEEDER_GEN_LOGIC}}AND feederMol.generation_formed >=
{{MIN_FEEDER_GENERATION}}
{{COMMENT_OUT_FEEDER_GEN_LOGIC}}AND feederMol.generation_formed <=
{{MAX_FEEDER_GENERATION}}


// Finally, return 1 matching structure to look at! (Tabulate results of
many using a different query)
RETURN ringMols, size(ringMols) as countMolsInRing, ringRels,
relationships(branchedBeginMolPath) as branchedBeginMolPathRels,
relationships(attachedPath) as attachedPathRels, beginMol,
beginMolConsumer, feederMol, intermediateMol, consumerMol, nodes(ringPath)
as ringPathNodes, nodes(branchedBeginMolPath) as
branchedBeginMolPathNodes, nodes(attachedPath) as attachedPathNodes,
catMolInRing, autocatPathRels, autocatPathNodes // ringPath,
branchedBeginMolPath, attachedPath, autocatPath
LIMIT {{NUM_STRUCTURES_LIMIT}}
```