***Definitions, Propositions, Relationships, Shorthand Syntax, and Plans of Approach***

Grady Ward Thesis Spring 2016 Semester

**IN PROGRESS:**

R3

**BACKLOG:**

L1

A1

S2

**Definitions:**

"Equivalence on Paths 1": The sorted matrix that represents the values of paths (the Paths Matrix or Paths Object), of two input graphs, is identical up to P = V, where V is the number of vertices in each one of the graphs.

"Equivalence on Paths 2": Same as Equivalence on Paths, but with P = 2V.

"Agreement on Paths": Same as Equivalence on Paths, but with P defined in terms of any polynomial function of V.

"Probable Equality on Paths": Same as Agreement on Paths, but with P is any calculable value.

"Co-Paths Graphs": The two input graphs have probable equality on paths, but are not isomorphic to one another.

"Subgraphs": If not made more clear, this refers to the set of V graphs generated by copying G V times, and removing one vertex in each one of the copies, so that every vertex is removed from only one graph.

**Shorthand:**

Iso(G, H) = The graphs G and H are isomorphic to one another.

PEquiv1(G, H) = The graphs G and H are Equivalent on Paths 1.

PEquiv2(G, H) = The graphs G and H are Equivalent on Paths 2.

PAgree(G, H) = The graphs G and H Agree on Paths.

PEquals(G, H) = The graphs G and H have Probable Equality on Paths.

CoPaths(G, H) = The graphs G and H are Co-Paths graphs.

SubGs(G) = The set of subgraphs of the graph G.

Connections(a, P) = The set of connections between a vertex a and a disjoint paths object P.

GI = The abstract Graph Isomorphism Problem, or its computational complexity class

P = The Polynomial time complexity class.

PathsGroups(G) = Set of quazi-equivalence classes created by paths with high discrimination.

AutoGroups(G) = Set of automorphism classes of a graph G.

Calc(T) = The complexity class of the calculating the abstract task T.

Embedded(G) = The embedded subgraph of G generated by linking its true equivalence classes.

***Goals (G)***

**G1: GI ⊂ P**

The Graph Isomorphism subspace in algorithmic complexity is within the broader polynomial class.

* Status: Uncertain
* For: GI has not been proven to be NP complete, and being in Quazi-Polynomial is the lowest class that one can be in without yet being in P. Similar problems (originally not in NPC or P) have changed categorization in similar ways over time, as more is understood about them, and all that have followed this trajectory have ended up in P.
* Against: People have been working on GI forever, and there has never been a credible claim to a polynomial time algorithm.

**G2: ¬∃ G, H, CoPaths(G, H)**

There do not exist graphs that are probabilistically equal in Paths, but are non-isomorphic.

* Status: False
* Evidence: There exist graphs with V=10 E=16 such that their paths values are the same for all observed values of P (confirmed via Java runner), yet they are not isomorphic (guaranteed by NAUTY generation).

***Paths Length Power Propositions (L)***

**L1: PEquiv1(G, H) ⇒ PEquiv2(G, H)**

If two graphs are equivalent on paths values P=[1-V], then they are equivalent for values P=[1-2V].

* Status: Uncertain
* Evidence: None yet has been tested
* Testing Plan 1: Go through Co-Paths graphs that have already been generated, test to see if they differ at 2V.
* Testing Plan 2: Go back through original Trie Calculations, and see how many graphs (if any) differentiate at different trie levels. If ANY are in the range 10\*10+, then this proposition is false. If none meet this critera, then we may have a good candidate proposition here.

**L2: PEquiv2(G, H) ⇒ PAgree(G, H)**

If two graphs are equivalent on paths values P=[1-2V], then they are equivalent for P being any polynomial function of V.

* Status: Uncertain
* Evidence: None yet has been tested
* Testing Plan 1: Go through Co-Paths graphs that have already been generated, test to see if they differ at any known values (using the Java Program).
* We may not be able to verify this one.

**L3: PAgree(G, H) ⇒ PEquals(G, H)**

If two graphs are equivalent on paths for any polynomial function of V, then they are equivalent for all observable values of P.

* Status: Uncertain
* Evidence: None yet has been tested
* Attack Plan: This would require a theoretical justification, as opposed to a numerical calculation. I believe this is possible through multivariate equation wrangling.

**L4: PEquals(G, H) ⇒ Iso(G, H)**

If the paths of two graphs are identical for all observable values, then the two graphs are isomorphic.

* Status: False
* Evidence: CoPaths graphs do exist, see G2

***Examining Subgraphs (S)***

**S1: PEquals(G, H) ⇒ (Paths(SubGs(G)) = Paths(SubGs(H)))**

If the paths generated by two graphs G and H are identical, then all of the paths of their sub graphs are the same, meaning we have a one to one correspondence enforced by the equal cardinality of the sets.

* Status: False
* Evidence: CoPaths graphs do exist, and when we examine their subgraphs, none have a significant number of shared subgraphs (usually 2-4 / 10).
* In fact, we can even see that some co-paths graphs (V10E20MN) have NO graphs that are copaths in common in their subgraph sets, which tells us that copaths graphs can be bore out of more than a shared common root graph with automorphisms.

**S2: (Paths(SubGs(G)) = Paths(SubGs(H))) ⇒ PEquals(G, H)**

If all of the subgraphs' paths of two graphs, G and H, are the same, then the paths of the two graphs are the same. This is the inverse of S1.

* Status: Uncertain, Likely True
* Evidence: We have not started to reason about the stringent conditions imposed on a graph by demanding that all of its subgraphs be copaths. I believe that we may be able to prove this by looking at the algebraic constraints that this places on our graphs.

**S3: PEquals(G, H) ⇒ ∃ g ∈ SubGs(G), h ∈ SubGs(H) | Paths(g) = Paths(h)**

If two graphs share a paths object, then there exist a pair of vertices (one from each graph), such that when they are removed, the two resulting subgraphs (one of G, and one of H) also share a paths value (either through an isomorphism or simply through co-paths graphs).

* Status: False
* Evidence: Counterexample: A pair of co-paths graphs (V10E20MN) have NO graphs that are copaths in common in their subgraph-1 sets, which tells us that copaths graphs can be borne out of more than a shared common root graph with automorphisms.
* This is difinitive evidence that copaths graphs can be borne out of more than a -1 graph isomorphism.

**S4: (Paths(SubGs(G)) = Paths(SubGs(H))) ⇒ Iso(G, H)**

If all of the subgraphs' paths of two graphs, G and H, are the same, then the paths of the two graphs are the same, and moreover the two graphs are isomorphic.

* Status: Uncertain Waiting
* Note that this is simply a stronger statement than S2, and implies S2.
* Waiting: We should examine S2 before examining S4, as Not S2 implies Not S4.

***Reconstruction (R)***

**R1: Connections(a, Paths(A)) ∧ Paths(A) ⇒ Paths (A ∪ a)**

If we are given the Paths of a graph 'A', and a new vertex 'a', and know its connection to the rows of the paths object, then we can deduce a new paths object which describes the super graph created by conjoining the new vertex to the existing graph.

* Status: Uncertain
* Plan of Attack: Formalize this question in mathematical terms, and attempt to use known methods to show that only one answer can exist to a system of equations describing the new values of paths. Think about this in terms of the number of paths in A' as paths in A, and paths which involve a.

**R2: Connections(a, Paths(A)) ∧ Paths (A ∪ a) ⇒ Paths (A)**

If we are given a paths object for a graph, and told the connections of one of the vertices to remove, we can deduce the paths object of the modified graph once the vertex has been removed.

* Status: Uncertain
* Plan of Attack: Formalize this question in mathematical terms, and attempt to use known methods to show that only one answer can exist to a system of equations describing the new values of paths. Think about this in terms of the number of paths in A' as paths in A, and paths which involve a.

**R3: Paths (A ∪ a) ∧ Paths(A) ⇒ Connections(a, Paths(A))**

If we are given a paths object for a graph, and a paths object for that graph minus one vertex, we can deduce the connections that the removed vertex had within the context of the paths object.

* Status: Uncertain
* Plan of Attack: Formalize this question in mathematical terms, and attempt to use known methods to show that only one answer can exist to a system of equations describing the new values of paths. Think about this in terms of the number of paths in A' as paths in A, and paths which involve a.

**R4: R1, but limit a to exactly k connections, and see how far we can go (the maximum** guaranteed value of k).

* Status: Uncertain
* Plan of Attack: Utilize recursion and basic logic to figure out N=0, 1, 2, then focus on 3+, and figure out whether or not that can be achieved by the same means.

**R5: R2, but limit a to exactly k connections.**

* Status: Uncertain
* Plan of Attack: Utilize recursion and basic logic to figure out N=0, 1, 2, then focus on 3+, and figure out whether or not that can be achieved by the same means.

**R6: R3, but limit a to exactly k connections.**

* Status: Uncertain
* Plan of Attack: Utilize recursion and basic logic to figure out N=0, 1, 2, then focus on 3+, and figure out whether or not that can be achieved by the same means.

**R7: ∃ ρ | Paths(G, P) 1≤P≤ρ ⇒ Paths(G, P) 1≤P≤∞**

There exists some maximal value of power, such that if we have all the paths values beneath that value, we can use them to construct the paths values above them with determinism.

* Status: Uncertain
* Plan of Attack: Find out whether or not there is evidence to support the L1-L3 claims, and if there is, then attempt to prove this property, which is a much stronger assumption about the way that paths is generated, and the amount of information it can encode.
* Note that this proposition makes the paths invariant LESS descriptive, not more descriptive.

***Automorphism Groups (A)***

**A1: PathsGroups(G) = AutoGroups(G)**

The groups generated through paths object comparison are fundamentally the same as the true automorphism groups of a graph.

* Status: Uncertain
* Plan of Attack: Formalize my understanding of automorphism groups. Decide on a testing procedure which will determine whether a group is a properly formed automorphism group.
  + Remember that the AutoGroups can only be subsets of the paths groups, so we only need to test for hypothetical automorphisms within sets (thankfully)!
* Then, calculate whether in all of our graphs so far (through V=9), the paths function accurately deduces the automorphism groups of a graph.
* Then if that is found to be true, attempt to do the same for a select number of higher order graphs.

**A2: Calc( Iso(G, H) | AutoGroups(G) ∧ AutoGroups(H) ∧ G ∧ H) ∈ P**

We can create an algorithm that operates in polynomial time, which can take as input the automorphism groups of two graphs and calculate the presence of an isomorphism between the two graphs, and reports one such isomorphism, if one exists, with accuracy and precision.

* Status: Uncertain
* Plan of Attack: Research whether or not this question has been solved. My guess is that it has.

**A3: Calc(AutoGroups) ∈ P ⇒ GI ∈ P**

If we can calculate automorphism groups of a graph in polynomial time, then Graph Isomorphism is in polynomial time.

* Status: Uncertain Waiting
* Plan of Attack: This could be proven by A2, but could also be prove through alternative mechanisms.

***Miscellaneous (M)***

**M1: Paths(G) ⇒ Polygon(G)**

If we are given the paths of a graph, for some fixed power, we can find the corresponding values of polygons, through manipulation of the paths values up to a proportion of that fixed power.

* Status: False
* Why? Hamiltonian cycle is in NP-Complete, but Paths calculations are always in a polynomial power of the input graph. Even a polynomial of a polynomial is a polynomial.

**M2: ∃ G v=V, e=E, ∀ pv ∈ Paths(G), pv = k ⇒ ∃=1 such graph**

If all of the vertices of a graph with a set number of edges and vertices are in the same paths group, then there does not exist another graph that satisfies the same constraints: i.e. these constraints demand that there is either 1 or 0 graphs of this type.

* Status: Uncertain, Likely True, Not a Priority
* Evidence: The constraints placed by this kind of condition seem incredibly limiting. I t seems unlikely to me that there are two such co-paths graphs, given that we know what we know about co-paths graph generation and occurrence (usually requiring a shared base graph early in their lineage).
* This would likely require a strand of copaths graphs that would be very long, and it seems unlikely to me that two such strands can reach perfect paths symmetry without having an intersection/merge.
* However, my intuition on this stuff will necessarily be wrong frequently... This is not a priority, as I no longer need it for my inductive proof based on subgraph structure.

**M3: ∀ G, H, CoPaths(G, H) ⇒ !CoPaths(Embedded(G), Embedded(H))**

The embedded subgraphs (formed by analysis of the structure of the Paths Groups) of all co-paths graph pairs must be fundamentally distinct

* Status: False
* Evidence: We have found copaths graphs which possess no non-trivial automorphisms. This means that the embedded graphs formed by their automorphism classes are equivalent to the original graphs. This means that the paths of the embedded graphs are necessarily the same as their originals.
* Example graphs: V10E18AB.

Paths(SubGs(G))=Paths(SubGs(H)) ∧ Paths(G)=Paths(H) ⇒ Iso(G, H)

***Future Thoughts:*** *Not a current priority, but certainly worth thinking about and looking in to.*

***Performance (P)***

*Paths performs worse means that we are more likely (by probability) to find co-paths, non-isomorphic graphs within the given subset.*

**P1: Paths performs worse on medium connected graphs.**

**P2: Paths performs worse on larger numbers of vertices.**

**P3: Paths performs worse on even values of V.**

**P4: Paths performs worse on even values of E.**

***Construction (C)***

C1: We can always construct copaths graphs from a graph with non-trivial automorphisms, in a deterministic way that we can predict. What this looks like is left intentionally vague.

*Symbols, for future use.*

∩→←⇒⇐⇔∞≤≥∪⊃⊇⊄⊂⊆∈∉∧¬∅∨∃∀