Chapter 1

Finite Functions and Graphs

1.1 Abstract

Since infinite objects cannot be evaluated in a traditional sense, we admit only finite objects and organize them according to length. Using a special type of bijection we can simulate any finite object while also showing that organizing theorems by size can reveal how finite objects cannot express objects larger than themselvs. With some recursion we can show that there is no general finite algorithm that can solve a specific subgraph isomorphism problem.

1.2 The Ω function

1.2.1 For Functions

Let f be a finite function. Then f is the set of pairs $\{(a, f(a)) : a \in dom(f)\}$. Define $\chi_f := \{n \in \mathbb{N} : n \leq |f|\}$. Define the basis of f as $\beta_f := \{dom(f) \cup ran(f) \cup \chi_f \cup f\}$. Then define the extension of a function as the powerset of the Cartesian product of f's basis: $EX_f := \mathcal{P}(\beta_f \times \beta_f)$. Then $f \in \mathcal{P}(f) \subset EX_f$, and $f \in EX_f$.
Then define the Ω function on a finite function as a bijection identified by some bijective index $I_{\beta_f} : \beta_f \to \chi_{\beta_f}$, such that $\Omega(I_{\beta_f}, f) : EX_f \to \chi_{EX_f}$. If $x \in EX_f$, we say $\Omega(I_{\beta_f}, x)$ is the *address* of x in EX_f.

1.2.2 For Graphs

Let G be a finite arbitrary directed graph, $G = (V_G, E_G)$, where V_G are the vertices of G and E_G are the edges of G. Then G is the set of pairs $\{(V_1, V_2) : E(V_1, V_2)\}$ where $E(V_1, V_2)$ is an edge that maps V_1 to V_2. Define $\chi_G := \{n \in \mathbb{N} : n \leq E_G\}$. Define the

basis of G to be $\beta_G := \{V_G \cup E_G \cup \chi_G\}$. Similarly, define the extension of G to be $EX_G := \mathcal{P}(\beta_G \times \beta_G)$. Then $G \in \mathcal{P}(G) \subset EX_G$ and $G \in EX_G$.
Then define the Ω function on a finite graph G as a bijection identified by some bijective index $I_{\beta_G} : \beta_G \to \chi_{\beta_G}$, such that $\Omega(I_{\beta_G}, G) : EX_G \to \chi_{EX_G}$. If $h \in EX_G$, we say $\Omega(I_{\beta_G}, h)$ is the *address* of h in EX_G.

1.2.3 One Construction of $\Omega(I_f, f)$

Let f be a finite directed graph and $f = \{p_n\} \exists n \in \mathbb{N}$ where $p_n = (V_1, V_2)$ if f has an edge from V_1 to V_2. Since β_f is finite, pick an arbitrary indexing $I_f : \beta_f \to \chi_{\beta_f}$. Then for any pair $p_i \in f$, $p_i = (a, b) \in \beta_f \times \beta_f$. Define the *order* of the pair as the cantor pairing function of (a, b) using $x = I_{\beta_f}(a)$ and $y = I_{\beta_f}(b)$, where the cantor pairing function is defined as $C(p_n) = C(x, y) = (1/2)(x+y)(x+y+1)+y$. Let S be a binary representation with a 1 at each digit place for each $C(p_n)$ for all p_n in f. Then the address of f with respect to I_{β_f} is $\Omega(I_{\beta_f}, f) = int(S)$, where int is the integer representation of a binary representation.

1.2.4 Cantor Pairs and Appending to a Basis

Consider EX_f for some finite function f. Then consider a finite g such that $\beta_f \subset \beta_g$. Then if $I_{\beta_f} = I_{\beta_g}\big|_{I_{\beta_f}}$ and $x \in EX_f$, $\Omega(I_{\beta_f}, x) = \Omega(I_{\beta_g}, x)$.
Proof
Let $x \in EX_f$. Then $\Omega(I_{\beta_f}, x) = int(S)$ for some appropriate S. Then $\Omega(I_{\beta_g}, x) = int(S')$ for some appropriate S'. It suffices to say $S = S'$. Since $I_{\beta_f} = I_{\beta_g}\big|_{I_{\beta_f}}$, we know that for all $(a, b) \in x$, $I_{\beta_f}(a) = I_{\beta_g}(a)$ and $I_{\beta_f}(b) = I_{\beta_g}(b)$ which means $C(x, y)$ under I_{β_f} is the same as $C(x, y)$ under I_{β_g}. Then $S = S'$ and $\Omega(I_{\beta_f}, x) = \Omega(I_{\beta_g}, x)$. Thus appending extra elements to any basis preserves old addresses if we preserve the structure of the smaller index function.
Corollary
Given finite f, g, and $|f| < |g|$, $\beta_f \subset \beta_g$, $f \in EX_g$.
Proof
Since we know $x \in EX_f \implies \Omega(I_{\beta_f}, x) = \Omega(I_{\beta_g}, x)$ and $f \in EX_f$, set $x = f$ and $f \in EX_g$.

1.2.5 Graph Isomorphisms and Addresses

Given two graphs $G = (V_G, E_G)$, $H = (V_H, E_H)$, I_{β_G}, I_{β_H}, if there exists a $\phi : V_G \cup V_H \to V_G \cup V_H$ such that $\Omega(I_{\beta_G} \circ \phi, H) = \Omega(I_{\beta_H} \circ \phi, G)$, then we say that G is isomorphic to H.

Proof

Let $\Omega(I_{\beta_G} \circ \phi, H) = \Omega(I_{\beta_H} \circ \phi, G) = Z$ for some integer Z. Then let $bin(Z)$ be the binary representation of Z, and $list(Z)$ be a list of indices for each 1 starting from the right and counting from zero in the decimal description of $bin(Z)$. For example the $list(5) = 0, 2$. Then each integer i in the $list(Z)$ is a pair under the inverse cantor pairing function $C^{-1}(i)$ such that $\exists a, b \in \mathbb{N}$, $C^{-1}(i) = (a, b)$. Then by hypothesis, for all i in $list(Z)$, $C^{-1}(i) = (a, b) = ((I_{\beta_H} \circ \phi)^{-1}(a), (I_{\beta_H} \circ \phi)^{-1}(b)) \in E_G$ and so $list(Z)$ under $I_{\beta_H} \circ \phi$ encodes E_G. Similarly, for all i in $list(Z)$, $C^{-1}(i) = (a, b) = ((I_{\beta_G} \circ \phi)^{-1}(a), (I_{\beta_G} \circ \phi)^{-1}(b)) \in E_H$ and $list(Z)$ under I_{β_G} also encodes E_H. Since E_G and E_H are both encoded in $list(Z)$ with some vertex relabeling, $G \cong H$. Then $\phi|_{V_G}$ and $\phi|_{V_H}$ are maps such that $G \cong H$.

Corollary

Let $\alpha = \Omega(I_{\beta_G} \circ \phi, G)$ and $\beta = \Omega(I_{\beta_H} \circ \phi, H)$. Then if $bin(\alpha) \leq_c bin(\beta)$ componentwise (where $bin(\alpha)$ is just an initial segment of $bin(\beta)$), we say G is isomorphic to some J subgraph of H.

Proof

Since $bin(\alpha) \leq_c bin(\beta)$, just look at where $bin(\alpha) = bin(\beta') = Z$. Then by the previous proof G is isomorphic to some graph J corresponding to β'. Since $bin(\beta')$ is an initial portion of $bin(\beta)$, graph H is graph J with extra elements, or that graph J is a subgraph of graph H. Then $G \cong J \subset H$.

Then define $SI(G, H) = \{\phi : bin(\alpha) \leq_c bin(\beta)\}$ and exclude the ϕ such that ϕ maps all the elements in its domain to one element in the range. Then nonempty $SI(G, H)$ gives a yes answer if G is isomorphic to some J in H and an empty $SI(G, H)$ gives a no answer if G is not isomorphic to some J in H. If ϕ identifies similar vertices, then note that empty $SI(G, H)$ means ϕ is the identity function.

1.2.6 The Definition of an Algorithm

Let S be a set. Then define $\chi_S := \{n \in \mathbb{N} : n \leq |S|\}$. Define an *algorithm* M as a bijection that maps finite sets in a sequence $M : \chi_S \to S$ where $S := \{S_0, S_1, \ldots S_n\}$ $\exists n \in \mathbb{N}$ and $\forall i, |S_i| \leq m$ $\exists m \in \mathbb{N}$.

Then given a finite function f, $M_f \in EX_f$.

Proof

Let $(a, b) \in M_f$. Then $a \in \chi_f \subset \beta_f$ and $b \in \chi_f \subset \beta_f$. Then $(a, b) \in \beta_f \times \beta_f$ and $M_f \in \mathcal{P}(\beta_f \times \beta_f) = EX_f$.

1.3 Lemma 1

Given a finite function f, let any algorithm of f be called M_f. Then M_f is a graph transversal of some graph. Define a graph transversal as an algorithm $R : \chi_S \to S$ such that $S \subset V_G$ for some graph G.

Proof

Let f be a finite function. Then the $dom(f)$ can be indexed by $I : \chi_{dom(f)} \to dom(f)$. Then define the bijection $I_f = \{(a, b) : a \in dom(f), b = (a, f(a))\}$. Then the algorithm of f is the bijection $M_f = I_f \circ I$. Then it suffices to say that all algorithms are graph transversals. Given an algorithm $M : \chi_S \to S$, M is a graph transversal of the graph $G = (S, E_G)$, where G is a complete graph (for any V_1,V_2 in S, $E_G(V_1, V_2)$ and $E(V_2, V_1)$). Then M is a graph transversal by definition. Similarly, a finite function f can be viewed as a graph transversal of graph $G = (V_G, E_G)$, where $V_G = dom(f) \cup ran(f)$, $E_G = f$, and subsequently as subgraphs of a graph.

Then for any function f, call its graph representation as $R_f = (S, f)$ where $S = dom(f) \cup ran(f)$ and an algorithm of f as $M(I_f)$ since there are many choices of index I_f or just M_f for any arbitrary I_f.

1.4 Lemma 2

Let M, H be algorithms. If $H \subset M$, we say M *expresses* H. Let G be a graph. If $M_f \notin EX_G$, then $f \notin EX_G$, and no algorithm in EX_G can express f.

Proof

Let $M_f : \chi_f \to f$. Let $(a, b) \in M_f$ and $a \in \chi_f, b \in f$. Since $M_f \notin EX_G = \mathcal{P}(\beta_G \times \beta_G)$, there is some pair $(a, b) \in M_f$ such that where $a \notin \beta_G$ or $b \notin \beta_G$. If $a \notin \beta_G$, then $|\chi_f| > |\chi_G|$, so f has more distinct elements than G. If $b \notin \beta_G$, then G does not have all the elements of f, so $f \notin G$. Then $M_f \notin EX_G$ means no algorithm can express f in EX_G.

Corollary

Then it suffices to say that for any finite f, g that $\chi_f \not\subset \chi_G \implies f \notin EX_G$.

1.5 Lemma 3

1.5.1 Expression implies Returns

Given an algorithm $M : \chi_M \to \{S_1, S_2, ...S_n\}$, say M *returns* H if $H = S_n$. If an algorithm M expresses H, then there exists an $M' : \chi_{M'} \to \{S_1, S_2, ...S_n\}$ in EX_M such that M' returns H.

Proof
Let $H' = (1, H)$. Since $1 \in \chi_M \subset \beta_M, H \in S \subset \beta_M$, then $(1, H) \in \beta_M \times \beta_M \implies M' = (1, H) \in EX_M$.

1.5.2 Returns implies Expression

Given that M' returns some algorithm H, we can find some basis β_M such that $\beta_{M'} \subset \beta_M$ such that some $M \in \mathcal{P}(\beta_M \times \beta_M)$ expresses H.
Proof
Pick $\beta_M = dom(M') \cup ran(M') \cup M' \cup dom(H) \cup ran(H) \cup H \cup \chi_{M' \cup H}$. Then $H \in \mathcal{P}(\beta_M \times \beta_M)$ and H expresses itself.

1.6 Lemma 4

Let G, H be graphs. By lemma 1, $SI(G, H)$ can also take any finite function f by using any M_f as an argument. Then if $\phi \in SI(G, H)$, $\phi \notin EX_G$ and $\phi \notin EX_H$.
Proof
Since $\phi : V_G \cup V_H \to \ V_G \cup V_H$, construct $M_\phi : \chi_{V_G \cup V_H} \to \phi$. Then $|\chi_G| + |\chi_H| \in \chi_\phi$. Then $\chi_\phi \not\subset \chi_G, \chi_\phi \not\subset \chi_H$. By lemma 2, $\phi \notin EX_G$ and $\phi \notin EX_H$.

1.7 Subgraph Isomorphism Application

Assume there exists an algorithm T such that T solves $SI(G, H)$ for any two finite graphs. Then consider $SI(G, R_T)$. By lemma 4, $\phi \notin EX_G$ and $\phi \notin EX_{R_T}$. By lemma 2, since $\phi \notin EX_{R_T}$, for any algorithm T, T cannot express solution ϕ for this particular kind of subgraph isomorphism. Then there does not exist a writable general algorithm T that solves subgraph isomorphism.
Now define $SI_D(G, H)$ as the decision version of subgraph isomorphism. Then $SI_D(G, H)$ is 0 if $SI(G, H)$ is empty, and 1 otherwise. Then $SI_D(G, H)$ is a function $\psi : (G, H) \to \{0, 1\}$. Then $\chi_\psi = 1$. Then use Lemma 1.5.2 to construct a ψ' such that we express (G, H) in a readily identifiable way. Let ψ' list V_G, V_H and the result of $SI_D(G, H)$. Then $\chi_{\psi'} = |\chi_G| + |\chi_H| + 1$. Then similar to $SI(G, H)$, $\chi_{\psi'} \not\subset \chi_G$ and $\chi_{\psi'} \not\subset \chi_H$. Then assume there exists an algorithm T that solves $SI_D(G, H)$ for finite graphs G, H. Since ψ' is finite, it is possible to ask T to express ψ' instead of ψ. Looking at $SI_D(G, R_T)$ for any arbitrary finite graph G we say $\chi_{\psi'} \not\subset \chi_{R_T}$. Then lemma 2 says that a writable general algorithm T for $SI_D(G, H)$ does not exist either.