Chapter 1

Finite Functions and Graphs

1.1 Abstract

By a particular mapping, we know that graphs, functions, and algorithms can be identified as particular members of a finite powerset. By defining what it means for some algorithm M to express another algorithm X, we can use set theory tricks and the knowledge about the maximum size of particular expressible functions to show that some functions cannot be expressed at all.

1.2 The $\Omega-$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 $\chi_f := \{n \in \mathbb{N} : n \leq |dom(f) \cup ran(f)|\}$. Define the basis of f as $\beta_f := dom(f) \cup ran(f) \cup \chi_f$. Then define the extension of a function as the powerset of the cross 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 $\Omega-$function on a countable function as a bijective function identified by some bijection I, such that $f_{\Omega(I)} : \chi_{EX_f} \to EX_f$. Then define the inverse of a $\Omega-$function defined by some I as the *address* function, written as $f^a_{\Omega(I)} : EX_f \to \chi_{EX_f}$. If $x \in EX_f$, we say $f^a_{\Omega(I)}(x)$ is the *address* of x in EX_f.

1.2.2 For Graphs

Let G be an 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 $\beta_G := V_G \cup \chi_G$. 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 of G to be identified by some bijection I, such that $G_{\Omega(I)} : \chi_{EX_G} \to EX_G$ and the address function of G to be $G^a_{\Omega(I)} : EX_G \to \chi_{EX_G}$.

1.2.3 The Construction of $f^a_{\Omega(I)}$

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 $(a, b) \in \beta_f \times \beta_f$, define the *order* of the pair as the cantor pairing function of (a, b) using $x = I(a)$ and $y = I(b)$, where the cantor pairing function is defined as $C(p_n) = C(x, y) = (1/2)(x + y)(x + y + 1) + y$. Then the address of f with respect to I is $f^a_{\Omega(I)}(f) = \sum_{k=1}^{n} 2^{(C(p_n))}$.

In this way, given two functions g, f, and $\beta_G \subset f_\beta$, the address function can be used to refer to any function g at or below f's size.

1.2.4 Graph Isomorphisms in Using Addresses

Given two graphs $G = (V_G, E_G)$, $G' = (V'_G, E'_G)$, I_G, $I_{G'}$, if there exists a $\phi : V_G \cup V_{G'} \to V_G \cup V_{G'}$ such that $G^a_{\Omega(\phi \circ I_G)}(G) = (G')^a_{\Omega(\phi \circ I_{G'})}(G')$, then we say that G is isomorphic to G'.

Let $\alpha = G^a_{\Omega(\phi \circ I_G)}(G)$ and $\beta = (G')^a_{\Omega(\phi \circ I_{G'})}(G')$. Then define $bin(x)$ as the binary representation of some number x. 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 H subgraph of G'. Then define $SI(G, G') = \{\phi : bin(\alpha) \leq_c bin(\beta)\}$. Then nonempty $SI(G, G')$ gives a yes answer if G is isomorphic to some H in G' and an empty $SI(G, G')$ gives a no answer if G is not isomorphic to some H in G'. If ϕ identifies similar vertices, then note that empty $SI(G, G')$ means ϕ is the identity function.

1.2.5 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 $\forall i, |S_i| \leq m \ \exists m \in \mathbb{N}$.

1.3 Lemma 1

Given a finite function f, let the algorithm of f be called f_M and f_M 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. Pick an arbitrary indexing I of $dom(f)$. Then I is a bijection such that $I : \chi_{dom(f)} \to dom(f)$. 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, $E_G(V1, V2)$ and $E(V2, V1)$). Then M is a graph transversal by definition. Then functions f can be viewed as graph transversals, and subsequently subgraphs of a graph. Then for any function f, call its graph representation as $f_R = (S, E_f)$ where $(a, b) \in S \to E(a, b) \in E_f$ and the algorithm of f as $f_{M(I)}$ (since there are many choices of the indexing I), or just f_M if we take any arbitrary I.

1.4 Lemma 2

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

Proof

Let $f_M : \chi_f \to S$, and $f = S$ for some finite set S. Since $f_M \notin EX_G = \mathcal{P}(\beta_M \times \beta_M)$, then there is some pair (a, b) where $a \in \chi_f$ and $b \in S$. Then $a \notin \beta_M$ means $|S| > |dom(G) \cup ran(G)|$, so f has more distinct elements than G. Then $b \notin \beta_M$ means G does not have all the elements of f, and so $f \notin EX_G$. Then $f_m \notin EX_G$ means no algorithm in EX_G can express f since $f \notin EX_G$. Then by lemma 2, it suffices to show that $\chi_f \not\subset \chi_G$ immediately 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

Since $H \subset M$, $H \in EX_M$. Then pick $M' = \{(1, H)\}$.

1.5.2 Returns implies Expression

Given that M' returns some $H = S_n$ for some set S_n, we can find some M that expresses H.

Proof

Since S_n finite, we can construct $\beta_M = dom(M') \cup ran(M') \cup S_n \cup \chi_{M' \cup S_n}$. Then $H \in EX_M$ and so M expresses H. Then M in part 1.5.1 is related to M' in 1.5.2 through the following relation: $\beta_M \subset \beta_{M'}$

1.6 Lemma 4

Let G, G' be graphs. By lemma 1, $SI(G, G')$ can also take any algorithm f by using any f_M as an argument. Then if $\phi \in SI(G, G')$, $\phi \notin EX_G$ and $\phi \notin EX_{G'}$.

Proof

Since $\phi : V_G \cup V_{G'} \rightarrow V_G \cup V'_G$, $\phi_M : \chi_{V_G \cup V_{G'}} \mapsto (a, b)$ such that $(a, b) \in V_G \cup V_{G'} \times V_G \cup V_{G'}$, then $|\chi_G| + |\chi_{G'}| \in \chi_\phi$. Then $|\chi_G| + |\chi_{G'}| \notin \chi_G$ and $|\chi_G| + |\chi_{G'}| \notin \chi_{G'}$. Then $\chi_\phi \not\subset \chi_G$ and $\chi_\phi \not\subset \chi_{G'}$. Then by lemma 2, $\phi \notin EX_G$ and $\phi \notin EX_{G'}$.

1.7 Subgraph Isomorphism Application

Assume there exists an algorithm T such that T solves $SI(G, G')$ for any two graphs. Then consider $SI(G, T_R)$. By lemma 4, $\phi \notin EX_G$ and $\phi \notin EX_{T_R}$. By lemma 2, since $\phi \notin EX_{T_R}$, for any algorithm T, T cannot express the solution ϕ this particular kind of subgraph isomorphism. Then there does not exist a writable general algorithm T that solves subgraph isomorphism.

Now define $SI - D(G, G')$ as the decision version of subgraph isomorphism. $SI - D(G, G') := \{0, 1\}$ where $SI - D(G, G') = 0$ if $SI(G, G')$ empty, and $SI - D(G, G') = 1$ if $SI(G, G')$ nonempty. Then $\phi \in SI - D(G, G')$ is a function, $\phi : (G, G') \rightarrow 0, 1$. Then $\chi_\phi = |\chi_G| + |\chi_{G'}| + |\{0, 1\}|$. Then similar to the previous problem, $\chi(\phi) \not\subset \chi_G$ and $\chi_\phi \not\subset \chi_{G'}$. Then by assuming that there exists an algorithm T that solves $SI - D(G, G')$ for any G, G' and looking at $SI - D(G, T_R)$ for an arbitrary, G, by lemma 2, a writeable general algorithm T does not exist either.

1.8 Reducing Finite descriptions to finite functions

introduce: on the example: f(x) = x quine atom (object is itself) so say Quine("desc") = (obj,obj). Can use quine atom on "f(x) = x" and on "x" (even on partial stuff like "f(" but then "f(" has no "meaning"). define syntax define semantics algorithm of quine

atom is algorithm (1,obj) then use algorithm lemmas 1.5.1/2 to show that all writable objects are "finite relations" on the syntactical level.

¡ USING SOME KIND OF INDUCTION TO JUSTIFY MAPPINGS LIKE semantics of N to the symbol N!!!!!!!!!!!! ¿

mind blowing: it's a theorem "about" theorems and it's also a theorem itself, but now I know what the "levels" are and shit like that

do you think we have an infinite anything? if anything, functions like f(x)=x are just the *ideas* of mapping x to x, and that the idea of mapping x to x is finite as well, but not the actual mapping. the description only allows you to get finite information from arbitrarily anywhere then the function is finite syntactically but not semantically if an obj is finite semantically it is finite syntactically composition of finite functions is finite.