Finite Relations Primer

Robert Papa, UCI

1 Motivation

The general idea is to simulate terminating algorithms using set theory techniques. Since we see everything as a finite set, we also know that if objects are not in a specific powerset, we can say some specifics about how any terminating algorithm cannot state certain information.
One failure of the $\in$-relation in the set theory perspective is that a simple replacement of one symbol for another means that while you have the same structure and talk about the same things, using different symbols means that the $\in$-relation simply does not recognize the two structures as the same. Then we turn to modeling terminating algorithms as graphs in a specific powerset and look at their graph structure to say that two algorithms are "the same".

1.1 Step 1: How to Model any Terminating Algorithm

Then assume that any terminating algorithm A can be simulated as a finite sequence of finite states s_i:

$$A = \{s_1, s_2, ..., s_n\}, \exists n \in \mathbb{N}$$

.

1.2 Step 2: Force Algorithm $\leftrightarrow$ Graph Equivalence

For any finite set A, we can construct $\chi_A := \{n \in \mathbb{N} : n \leq |A|\}$, where $|A|$ is the size of A as a set.
Then for any set A, let's introduce a bijection M that automatically constructs a graph representation of A:

$$M_A : \chi_A \rightarrow A$$

Then the graph vertices are $V_A := \chi_A \cup A$ and the graph edges are simply M_A. Then the graph G_A constructed from M_A is $G_A = (V_A, M_A)$.

1.2.1 Remark

Note that M_A already contains enough information to construct G_A, so we should just work on M_A only, as M_A can also be viewed as a set of pairs.

1.2.2 Redefining Algorithm to Fit Set Theory

Let S be a set. Then define $\chi_S := \{n \in \mathbb{N} : n \leq |S|\}$. (This is the same definition as before.)
Define an *algorithm* M as a bijection that maps finite sets in a sequence $M : \chi_S \rightarrow S$ 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 M_A in 1.2 can be viewed as the *algorithm* of A, and A itself as just a set.

1.3 Step 3: Ask for all answers to be in some canonical form

Here we define what it means for an algorithm to express some answer B. Since we are looking for some finite state B which provides information, we can also look at B as an algorithm

$$B : \chi_\gamma \to \gamma$$

Where $\gamma = \{j_1, j_2, ..., j_n\}, \exists n \in \mathbb{N}$ and each j_i is just an arbitrary finite set. Then we say that A *expresses* answer B if $B \subset A$.

1.4 Step 4: What can't a terminating algorithm A say?

First let's force A and M_A to be in the same powerset using a specific basis:

$$\beta_A := \{dom(A) \cup ran(A) \cup \chi_A \cup A\}$$

Then A and M_A are in this powerset:

$$\mathcal{P}(\beta_A \times \beta_A)$$

where $\beta_A \times \beta_A$ is a Cartesian product.
Then let's abuse $\mathcal{P}(\beta_A \times \beta_A)$ in order to get a(n) (un)computability proof.

1.4.1 Lemma 2 from the paper (rewritten)

Let A, B be algorithms (so $A : \chi_\alpha \to \alpha$ and $B : \chi_\gamma \to \gamma$ for some arbitrary finite sets α and γ). If $B \subset A$, we say A *expresses* B as an answer.

Look at M_B. If $M_B \notin \mathcal{P}(\beta_A \times \beta_A)$, then $B \notin \mathcal{P}(\beta_A \times \beta_A)$, and no algorithm in $\mathcal{P}(\beta_A \times \beta_A)$ can express B.
Proof
Let $M_B : \chi_B \to B$. Let $(a, b) \in M_B$ and $a \in \chi_B, b \in B$. Since $M_B \notin \mathcal{P}(\beta_A \times \beta_A)$, force (a, b) to be a pair $(a, b) \in M_B$ such that $a \notin \beta_A$ or $b \notin \beta_A$.
If $a \notin \beta_A$, then we can say $a \notin \chi_A$, and we know $|\chi_B| > |\chi_A|$, so B has more distinct elements than any $A' \in \mathcal{P}(\beta_A \times \beta_A)$. Then $B : \chi\gamma \to \gamma$ cannot be a subset of $\beta_A \times \beta_A$ and is not an element of $\mathcal{P}(\beta_A \times \beta_A)$. For any $A' \in \mathcal{P}(\beta_A \times \beta_A)$, $|B| > |A'|$ and since A', B are finite sets, $B \not\subset A'$.
If $b \notin \beta_A$, then for any $A' \in \mathcal{P}(\beta_A \times \beta_A)$ we can say $b \in B$ and $b \notin A'$, and so $B \not\subset A'$. Then $B : \chi\gamma \to \gamma$ cannot be a subset of $\beta_A \times \beta_A$ and is not an element of $\mathcal{P}(\beta_A \times \beta_A)$.
Corollary
Then it suffices to say that for any finite f, g that $\chi_f \not\subset \chi_G \implies f \notin \mathcal{P}(\beta_G \times \beta_G)$ (and so g cannot express f).
Then $M_B \notin \mathcal{P}(\beta_A \times \beta_A)$ means no algorithm in $\mathcal{P}(\beta_A \times \beta_A)$ can express B.

1.5 Step 5: Subgraph Isomorphism Restrictions (Fully justified in 1.2.5 and Lemma 4 of the paper)

Let G, H be finite graphs. Then define a function $SI(G, H)$ such that $\phi \in SI(G, H) \Leftrightarrow G$ is isomorphic to some $J \subset H$. One such ϕ is $\phi : V_G \cup V_H \to V_G \cup V_H$ where ϕ identifies similar vertices on a graph.
Then by Lemma 4, we know that $\chi_\phi \not\subset \chi_G, \chi_\phi \not\subset \chi_H$.

1.5.1 Lemma 4

Let $G = (V_G, E_G)$, $H = (V_H, E_H)$ be graphs. Then if $\phi \in SI(G, H)$, $\phi \notin \mathcal{P}(\beta_G \times \beta_G)$ and $\phi \notin \mathcal{P}(\beta_H \times \beta_H)$.
NOTE: the basis for graphs are different: $\beta_G = \{V_G \cup E_G \cup \chi_G\}$ for any finite graph G. Also for graphs to express algorithms just define graph G expresses algorithm J if $J \subset E_G$.
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 corollary, $\phi \notin \mathcal{P}(\beta_G \times \beta_G)$ and $\phi \notin \mathcal{P}(\beta_H \times \beta_H)$, and so G and H cannot express ϕ.

1.6 Step 6: Subgraph Isomorphism with a terminating algorithm T

Asume a terminating algorithm $T : \chi_\gamma \to \gamma$ solves a generic $SI(G, H)$ for any two finite graphs G, H. Then since T is a terminating algorithm (and also a graph), look at $SI(T, G)$ for an arbitrary finite graph G. By lemma 4, we know that $\chi_\phi \not\subset \chi_T, \chi_\phi \not\subset \chi_G$. By lemma 2 corollary, $\phi \notin \mathcal{P}(\beta_T \times \beta_T)$ and $\phi \notin \mathcal{P}(\beta_G \times \beta_G)$, and so T and G cannot express ϕ.
So if you want a terminating algorithm T to solve $SI(G, H)$, I can show you a problem T cannot solve, namely $SI(T, G)$ for any arbitrary finite graph G.