Proof Re Human Function Expansion

J.H.DeBlois

December 11, 2025

1 Let's Look at Our Tech Future

This proof is offered to counteract the thought that software is taking over our lives. It isn't. The tech revolution continues for sure. So the number of syntactically and semantically correct computer programs running keeps growing. As do the programs that have flaws. We can view the number of computer programs as countable infinite.

On the other hand, if we try to count the number of human functions, it's harder. The number of human thoughts and actions (that we call human functions) is not only infinite but uncountably infinite (like the infinity of real numbers you can find between any two real numbers). I offer a proof that the number of human functions will continue to expand faster that the number of programs. Think of the number of meetings occurring all over the world. Now add to it the additional number of meetings in which humans are discussing AI. There is phenomenal growth on all sides.

Think also of the number of times in the day when your activity has nothing to do with a computer program. That may be less than in the past, or you may perceive it as that. (Perhaps you should keep your non-computer related human functions at some steady level to avoid downside risks?) But if you are not a computer person, you still may encounter doing many things a new way in your life.

2 Proof re Human Function Expansion

This proof has two main steps. First, we count the number of possible computer programs. Second, we count the number of human functions. Our answers demonstrate that the count of human functions exceeds of the computer programs. A similar proof has been taught in CS420 Theory of Computation and in CS622 Theory of Formal Languages. If we focus on the enormity of human functions, we would then be able to build and use code in proper prospective.

2.1 What is To Be Proved: Step 1

We would like to prove that whereas the number of possible computer programs is countably infinite, it is less than the number of human functions, which is uncountably infinite. Computer programs can be modeled as words in the language of all computer programs. Some are valid programs and some are invalid programs. We list them in order. We prove that the size of the set is countably infinite. The symbol $\aleph_0$ read as "aleph null" stands for a countable infinity.

2.2 What is To Be Proved: Step 2

Human functions can be modeled, but attempts to list them fail, and we can prove that the number of such functions is uncountably infinite. Thus, forever bigger than $\aleph_0$.

When a human activity is partially replaced by a computer program, some new human functions may become necessary. Our proof shows that the total of human functions must grow too, and by more. The reference for the next two sections is Theory of Formal Languages with Applications, Dan A. Simovici and Richard L. Tenney, World Scientific, 1999.

3 Count the Number of Computer Programs

To count the computer programs, we use the concepts of sets, cardinality, bijection, alphabet, words and language. As Prof. Simovici mentioned (p. 57), "Programs and the data they manipulate may be regarded as words over appropriate alphabets."

On p. 57, an alphabet is a finite non-empty set. The elements are symbols. Definition 2.2.1 A word of length n on an alphabet A is a sequence of length n of symbols of this alphabet.

Also, "Example 2.2.3, p. 58, Any C program is a word over the basic alphabet of this language that includes small and capital letters, as well as special symbols, such as parentheses, brackets, braces, spaces, new line characters, quotation marks, etc. Not all these characters are visible; in other words, some characters (such as spaces) appear on paper only as white spaces. For example, the famous C program:

```
#include <stdio.h> main(){ printf("hello, world\n"); }
```

can be looked at as a word."

Now, from cs622 class notes, "Numbering Words", we have: "Let $A = \{a_0, \ldots, a_{n-1}\}$ be an alphabet containing n symbols. Words over A can be encoded as natural numbers; in other words, we can define a bijection $\phi_A : A^* \longrightarrow N$ by:

$$\phi_A(x) = \begin{cases} 0 & \text{if } x = \lambda \\ n\phi_A(y) + i + 1 & \text{if } x = ya_i \end{cases}$$

for every x in A^*."

This is a recursive function that pushes left on each step. See Example, same page, if $A = \{a_0, a_1, a_2\}$, $x = a_0a_1a_0a_2$ and $y = a_2a_2a_2$, then:

$$\phi_A(x) = 3^3 \cdot 1 + 3^2 \cdot 2 + 3^1 \cdot 1 + 3 = 51$$
$$\phi_A(y) = 3^2 \cdot 3 + 3^1 \cdot 3 + 3 = 39$$

The example is hard to understand. If we have no characters in the program we assign it zero. If we have one character from the alphabet, we use 'n', the number of characters in the alphabet, to help us. If the program is the fifth character in the alphabet, we code it five plus one. (The alphabet started at $'a_0'$.) To add another character, we push the one we have to the left and add the new one on the right. Our string has two characters, $'a_first'$ and $'a_second'$. We multiply the code for $'a_first'$ by n and then we give $'a_second'$ its code from the alphabet plus one. Say that $'a_second'$ is the first character in the alphabet. We would have the count five plus one times n plus one plus one. Keep pushing left and multiplying by n as you expand the complete listing of a program.

Note that any word of a certain length will have its interval, no two distinct words in A^k can be mapped into the same number in $l(k)$, so ϕ_A defines an injection of A^k into $l(k)$; and for every number m in $l(k)$ there is a word x in A^k such that $\phi_A(x) = m$. Therefore, ϕ_A is a bijection between A^* and $\mathbb{N}$.

Therefore, the number of possible computer programs is a countable infinity.

4 Count the Number of Human Functions

Now, we count the number of human functions. Suppose we start with a list L of the basic things humans can do and think. Then, because ideas take form in different ways, we let each human function be the subset of that list that has a slight change. We list all the subsets (it's infinite). We set the subset indicator to zero if it did not change or to one if it did change. Each row in the last has a particular combination of small changes. Thus, we have a model of Human Function Expansion.

Next we show that $P(L)$ is not countable in a proof by contradiction. Suppose $P(L)$ is countable. Then we have the mapping $f : \mathbb{N} \longrightarrow P(L)$.

Per p. 44, we could have this list:

$$\begin{array}{lllllllll}
0: & a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & \cdots & \text{(a particular subset of changes)} \\
1: & a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & \cdots & \\
2: & a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & \cdots & \\
3: & a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & \cdots & \\
4: & a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & \cdots & \\
5: & a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & \cdots & \\
 & & \vdots & & & & & \\
\text{k}: & a_{k0} & a_{k1} & a_{k2} & a_{k3} & a_{k4} & \cdots & a_{kk}
\end{array}$$

where:

$$a_{ij} = \begin{cases} 0 & \text{if } j \notin f(i) \text{ where } f(i) \text{ is a particular subset} \\ 1 & \text{if } j \in f(i) \end{cases}$$

Then, quoting further from Simovici and Tenney, p. 44, "In other words, the a_{ij}s correspond to the characteristic function of the set $f(i)$. The set D is formed by 'going down the diagonal' and spoiling the possibility that $D = f(k)$, for each k. At row k, we look at a_{kk} in column k. If this is 1, i.e., if $k \in f(k)$ then we make sure that the corresponding position for the set D has a 0 in it by saying that $k \notin D$. On the other hand, if a_{kk} is a 0, i.e., $k \notin f(k)$, then we force the corresponding position for the set D to be a 1 by putting k into D. This guarantees that $D \neq f(k)$, because its characteristic functions differ from that of $f(k)$ in column k."

Thus no bijection. Therefore, we have a proof that the number of human functions is an uncountable infinity.

This proof technique, diagonalizaton, first appeared in the 1891 paper of Georg Cantor.

5 What Do You Think?

Please ask your questions! Thank you for taking this class.

6 References

1. DeBlois, J.H., https://www.cs.umb.edu/ hdeblois/0list51/. Click on JHD CV or explore cs410 student initial simple projects.

2. Joint Task Force on Computing Curricula, Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Progams in Software Engineering, August 23, 2004.

3. National Center for Science and Engineering Statistics, National Science Foundation, "Diversity and STEM: Women, Minorities and Persons with Disabilities: 2023", NSF report.

3. Roycroft, Sian, Elevate Your Teaching: Discover the Power of Educational Paradigms, Edge Education, https://edgeeducation.com/discover-the-power-of-educational-paradigms/, June 21, 2023.

4. Rubin, Kenneth S., Essential Scrum: A Practical Guide to the Most Popular Agil Processes, Addison-Wesley, 2013.

5. Simovici, Dan A. and Richard L. Tenney, Theory of Formal Languages with Applications, World Scientific, 1999.

6. Sotomayor, Sonia, Speaking at Radcliffe Day, June 2024, video on youtube.