Indexed Concatenation Notation: A Novel Way to Summarize Networks and Other Complex Systems

Kenneth Caviness[1][a], Colton Davis[1][b], Derek Renck[1][c], Charles Sarr[2], Scot Anderson[3][d], Heaven Robles[4][e] and Rhys Sharpe[3][f],

[1]*School of Engineering and Physics, Southern Adventist University, Taylor Circle, Collegedale, U.S.A.*

[2]*Laurelbrook Academy, Campus Drive, Dayton, U.S.A.*

[3]*School of Computing, Southern Adventist University, Taylor Circle, Collegedale, U.S.A.*

[4]*Biology and Allied Health Department, Southern Adventist University, Taylor Circle, Collegedale, U.S.A.*

Keywords: Graph Identification, Concatentation, Indexed Concatenation, Lossless Compression, Edge Difference Set List.

Abstract: The indexed concatenation notation presented in this paper extends the concept of concatenation in a way similar to the extension of addition to the indexed sum, allowing compact representations of strings, lists, matrices, etc., having internal repetitive or describable structure. In particular, it allows the edge difference set list of any graphical network with a visible pattern to be summarized in an extremely compact and lossless way. Examples highlight the information compression of the technique and showcase its ability to represent complicated, infinite patterns in closed form.

1 INTRODUCTION

Graph theory provides mathematicians and computer scientists with many tools for the study of networks, but none compactly identify a large or infinite network. That situation has now changed: the Indexed Concatenation Notation (ICN) allows any network with a visible pattern to be "reduced" to form an extremely compact summary, which could then serve, for example, as a "dictionary entry" in a list of networks studied.

The notation defined here owes much to the indexed sum, product, union and notations already widely used in mathematics; it is a modern reincarnation of an idea N. G. de Bruijn jotted down in 1977 (de Bruijn, 1977). Although our direct motivation comes originally from graph theory, concatenation has deep roots in programming languages and in computer science in general. As far back as the early 70s, the theory of concatenation included not only strings, but lists as well (Campbell, 1971). Almost all programming languages include concatenation operators, but C can perform this operation in at least five different ways, using either functions or more primitive manipulations (WsCubeTech, 2025). Concatenation forms a fundamental operation in the theory of computation, where it is essential in the understanding of different types of languages from regular expressions to Turing machines (Sipser, 2012; Maheshwari and Smid, 2024).

The proposed extension of concatenation to allow indexing strengthens the notation, which can then be used with strings, lists, sequences, etc.

The ICN can be considered a new form of lossless information compression for strings, lists, sequences, and networks.

2 MOTIVATION

In our work over several years, we have consistently encountered networks having some obvious visual pattern. One such example is shown in Figure 1.

Only a small part of the infinite two-dimensional network is shown, with the first vertex of the graph indicated by the small black arrow at the bottom right.

[a] https://orcid.org/0009-0008-5240-6260
[b] https://orcid.org/0000-0003-4348-7923
[c] https://orcid.org/0009-0002-7564-2123
[d] https://orcid.org/0009-0009-5053-555X
[e] https://orcid.org/0009-0008-5434-7444
[f] https://orcid.org/0009-0001-9671-3763

Caviness, K., Davis, C., Renck, D., Sarr, C., Anderson, S., Robles, H. and Sharpe, R.
Indexed Concatenation Notation: A Novel Way to Summarize Networks and Other Complex Systems.
DOI: 10.5220/0013514700003970
In *Proceedings of the 15th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2025)*, pages 39-50
ISBN: 978-989-758-759-7; ISSN: 2184-2841

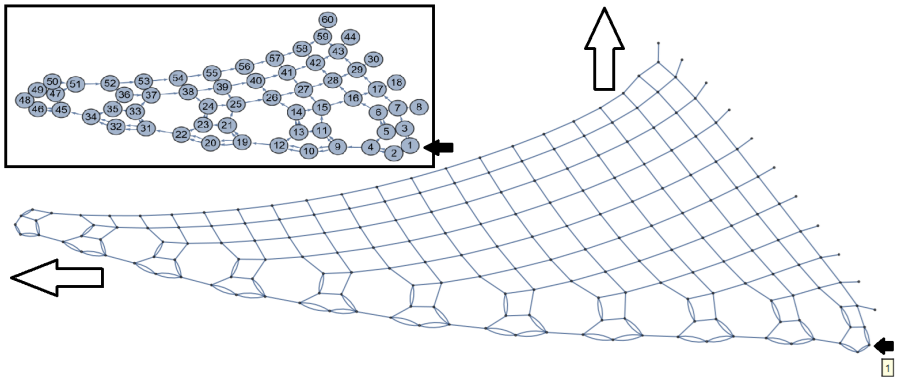


Figure 1: An infinite two-dimensional network with a consistent internal pattern.

Table 1: The first edges of the network shown in Figure 1.

$1 \to 2$	$1 \to 2$	$1 \to 3$	$1 \to 3$	$2 \to 4$
$2 \to 4$	$3 \to 5$	$3 \to 7$	$4 \to 5$	$4 \to 9$
$5 \to 6$	$5 \to 6$	$6 \to 7$	$6 \to 16$	$7 \to 8$
$7 \to 17$	$9 \to 10$	$9 \to 10$	$9 \to 11$	$9 \to 11$

The graph grows without bound, extending indefinitely up and left. (See outlined arrows.) The small inset shows the first 60 vertices and edges between them, with vertex numbers indicating the order in which they were added to the graph as it was constructed.

Although the layout is arbitrary, the graph has clear patterns along its growing edges and in its interior. Visually, the lower edge of the graph is made up of alternating pentagons and heptagons, with one quadrilateral sitting on each pentagon. Each polygon is made of single and double edges, always in the same positions. The interior of the graph is single-edged quadrilaterals only, but the right-most one on each row has a missing right side.

But such descriptions are not definitive and might apply equally well to many other networks. Similarly, graph-theoretic properties such as order and size (both infinite here), connectivity, vertex degrees, density, girth, radius, diameter, height, etc., do not uniquely identify a graph. It is frustrating to repeatedly come across similar-seeming graphs that, on closer inspection, turn out to be different. A graph is uniquely defined by its set of vertices and its set of edges, and the only reliable identification comes from considering those sets. We looked for ways to summarize this data, without any loss of information. The first few directed edges of our example graph, sorted by vertices connected, are as shown in Table 1.

Any pattern visible to human eyes in the graph above is hidden in this edge list, but the simple operation of first grouping edges according to their starting vertex and then taking the differences of starting and ending vertex numbers for each edge suddenly reveals new vistas for the pattern seeker.

No information is lost in this process, since each set, S_i, completely specifies the edges originating at vertex i. The same information is in both the list of edges (grouped by originating vertex) and the *edge difference set list* (EDSL). Table 2 demonstrates how an EDSL is formed.

Table 2: Network edges and the corresponding EDSL.

Edges:	EDSL:
$\{1 \to 2, 1 \to 2, 1 \to 3, 1 \to 3\}$	$\{1,1,2,2\}$
$\{2 \to 4, 2 \to 4\}$	$\{2,2\}$
$\{3 \to 5, 3 \to 7\}$	$\{2,4\}$
$\{4 \to 5, 4 \to 9\}$	$\{1,5\}$
$\{5 \to 6, 5 \to 6\}$	$\{1,1\}$
$\{6 \to 7, 6 \to 16\}$	$\{1,10\}$
$\{7 \to 8, 7 \to 17\}$	$\{1,10\}$
$\{\}$	$\{\}$
$\{9 \to 10, 9 \to 10, 9 \to 11, 9 \to 11\}$	$\{1,1,2,2\}$
...	...

The EDSL of a network having some intrinsic pattern frequently includes subsequences of exact duplicate sets, providing an obvious first step toward reduction to a summary. When we see, for instance, 2 adjacent copies of $\{1,10\}$, 4 adjacent copies of $\{1,12\}$, 6 copies of $\{1,14\}$, 8 copies of $\{1,16\}$, and so on, we temporarily represent these duplicate subsequences by $\in^2[\{1,10\}]$, $\in^4[\{1,12\}]$, $\in^6[\{1,14\}]$, $\in^8[\{1,16\}]$, etc. (More on the notation shortly.) Compressing those allows us to notice other similar cases, and soon we have the first reduction of this graph's EDSL:

$$\{\{1,1,2,2\},\{2,2\},\{2,4\},\{1,5\},\{1,1\},\overset{2}{\Subset}[\{1,10\}],\{\},\{1,1,2,2\},\{2,2\},\{2,4\},\{1,7\},\{1,1\},\overset{4}{\Subset}[\{1,12\}],\{\},\{1,1,2,2\},\{2,2\},\{2,4\},\{1,9\},\{1,1\},\overset{6}{\Subset}[\{1,14\}],\{\},\{1,1,2,2\},\{2,2\},\{2,4\},\{1,11\},\{1,1\},\overset{8}{\Subset}[\{1,16\}],\{\},\ldots\} \quad (1)$$

In this case no further reduction can be achieved based on exact duplicates of adjacent sets or exact duplicates of subsequences, but each 7-element row (subsequence) above has a common pattern, with only 3 numbers changing in each row. In the first row, these numbers are 5, 2, and 10; in the k^{th} row, they are $2k+3$, $2k$, and $2k+8$. Before moving on to the details of the Indexed Concatenation Notation, glance at the fully reduced form of this EDSL in all its glory.

$$\left\{\overset{\infty}{\underset{k=1}{\Subset}}\left[\{1,1,2,2\},\overset{2}{\underset{j=1}{\Subset}}[\{2,2j\}],\{1,2k+3\},\{1,1\},\overset{2k}{\underset{i=1}{\Subset}}[\{1,2k+8\}],\{\}\right]\right\} \quad (2)$$

Note the index variables, i, j and k, which have been introduced. Just as one would expect by analogy to an indexed summation, i, j and k take on the initial value of 1 and are incremented until the specified final values are reached: here, $2k$, 2, and infinity, respectively. Assuming the pattern has been well established and will continue indefinitely, we have succeeded in summarizing the entire infinite network's EDSL in two scant lines!

3 NOTATION AND PRECEDENTS

We define the operators both for standard and indexed concatenation, explaining the background of the current indexed concatenation notation.

3.1 Binary (Infix) Operator

There is no generally accepted binary operator for concatenation. Different programming languages implement different operators for strings, such as S_1+S_2 in C++, Java, Pascal, and Python; $S_1.S_2$ in PHP and Perl; S_1*S_2 in Julia; $S_1||S_2$ in SQL; $S_1\&S_2$ in Ada, BASIC, and .NET; $S_1 \sim S_2$ in D; $S_1//S_2$ in Fortran; $S_1\hat{}S_2$ in F#; $S_1..S_2$ in Lua; S_1,S_2 in Smalltalk and APL; and $S_1 <> S_2$ in Wolfram Mathematica. In our opinion, each of these options has significant drawbacks, each having widely varying meanings in other contexts. Nor would the situation be improved by importing the function composition operator used in mathematics: $f_1 \circ f_2$.

Several languages, including Haskell, Zig, and Erlang use two consecutive plus signs (++) for concatenation, and in the case of Haskell, the shifted overstrike (Unicode 10746, hex 29FA): ⧺. This symbol overcomes many of the drawbacks of the others. We propose to use it as the binary or infix version of concatenation. For example, we write

$$\{1,2,3\} ⧺ \{5,4,3\} ⧺ \{7\} = \{1,2,3,5,4,3,7\} \quad (3)$$

3.2 Indexed Operator

Consider other familiar notations, such as sum, product, and union, both simple (binary, two-argument) and indexed operators:

$$\sum_{i=1}^{3} a_i = a_1 + a_2 + a_3 \quad (4)$$

$$\prod_{i=1}^{3} a_i = a_1 \cdot a_2 \cdot a_3 = a_1 a_2 a_3 \quad (5)$$

$$\bigcup_{i=1}^{3} S_i = S_1 \cup S_2 \cup S_3 \quad (6)$$

Some examples with numbers:

$$\sum_{i=1}^{4} i = 1+2+3+4 = 10 \quad (7)$$

$$\prod_{i=1}^{3} i = 1 \cdot 2 \cdot 3 = (1)(2)(3) = 6 \quad (8)$$

$$\bigcup_{i=1}^{4} \{i, i^2\} = \{1,1\} \cup \{2,4\} \cup \{3,9\} \cup \{4,16\} = \{1,2,3,4,9,16\} \quad (9)$$

Each indexed operator is defined as an extension of the corresponding binary operator: $+ \rightarrow \sum$, $\cdot \rightarrow \prod$, $\cup \rightarrow \bigcup$. The sum and product use different symbols for the two, while the indexed union of sets reuses the binary operator symbol for the indexed case, as does the intersection.

Having a choice of precedents to follow, we opt for the following for indexed concatenation:

$$\overset{3}{\underset{i=1}{\Subset}} S_i = S_1 ⧺ S_2 ⧺ S_3 \quad (10)$$

We settled on the euro symbol, which exists widely, but not in this context, so no misunderstanding should occur. Visually, it resembles a "C" for "Concatenate", and also resembles our choice for the infix concatenation operator, $+\!\!+$, selected from the Haskell computer language.

We begin by closely following the indexed union, defining the indexed concatenation in terms of the simple operator. Crucially, unlike indexed union, concatenation does not eliminate duplicates nor sort the resulting list. For example,

$$\begin{aligned}\Concat_{i=1}^{4} \{i, i^2\} &= \{1,1\} +\!\!+ \{2,4\} +\!\!+ \{3,9\} \\ &+\!\!+ \{4,16\} = \{1,1,2,4,3,9,4,16\}\end{aligned} \tag{11}$$

Similar work has been attempted previously. Nicolaas Govert "Dick" de Bruijn, a Dutch mathematician noted for his contributions to analysis, number theory, combinatorics and logic, published a memorandum describing a notation for both simple and indexed concatenation (de Bruijn, 1977). While we have based our ICN on the more familiar summation notation, de Bruijn's approach is slightly less intuitive. He uses a three-sided box over each item to be concatenated, forming what he terms *comb notation*. His definitions are solidly rooted in set theory and include a recursive definition for the indexed concatenation operator. But their time had, apparently, not yet come. "This one's for you, Dick!"

4 DEFINITIONS

We define the concatenation operator as applying to strings, sequences, lists (shown here as sets, but with no presumed sorting or removal of duplicates), and any other similar objects (Table 3). Notice that the concatenation of two strings results in a string, that of two lists gives a list, etc. In all cases this can be thought of as retaining the outer delimiters (those before a_1 and after b_n) while the inner delimiters, those adjacent to the concatenation symbol ("$+\!\!+$", $\}+\!\!+\{$, or $]+\!\!+[$), disappear with it. The use of curly braces $\{\}$ for lists aligns well with common usage for sets in mathematics, where sequences are often shown without delimiters, and occasionally with parentheses (Rehmann, 2020). The square brackets [] used above for sequences is another nod to the Haskell language, but we treat this as a *vanishing* delimiter (see below).

Indexed concatenation (of any n objects S_i of the same type) is now defined as a generalization of the binary operator:

$$\Concat_{i=1}^{n} S_i = S_1 +\!\!+ S_2 +\!\!+ \ldots S_n \tag{12}$$

Of course, any index may be used, or if not essential for defining the process, it can be omitted. If the initial value is omitted, a desired default (generally 1 or 0) should be specified. We saw examples of this in the Motivation section.

5 SIMPLE EXAMPLES

We provide multiple examples of the capabilities of indexed concatenation when applied to lists, matrices, and other datatypes. In each case indexed concatenations shows promising potential to concisely summarize repeated patterns.

5.1 Strings

Concatenation of strings works exactly as expected:

$$\begin{aligned}\Concat_{i=1}^{3} \text{“ABC”} &= \text{“}ABC\text{”} +\!\!+ \text{“}ABC\text{”} +\!\!+ \text{“}ABC\text{”} \\ &= \text{“}ABCABCABC\text{”}\end{aligned} \tag{13}$$

Overloading the "+" operator to advance the character a certain number of steps in a given alphabet (such as adding an int to a char in C-like languages), allows this nested example:

$$\begin{aligned}\Concat_{i=0}^{3} \Concat_{j=0}^{2} (\text{“}A\text{”} + i + j) &= \text{“}ABC\text{”} +\!\!+ \text{“}BCD\text{”} \\ +\!\!+ \text{“}CDE\text{”} +\!\!+ \text{“}DEF\text{”} &= \text{“}ABCBCDCDEDEF\text{”}\end{aligned} \tag{14}$$

5.2 Integers and Digits

The notation might even extend to concatenating the digits of numbers, when the argument is neither a list nor a sequence, but an integer: the concatenation of integers should be an integer.

$$\Concat_{i=1}^{5} i^2 = 1 +\!\!+ 4 +\!\!+ 9 +\!\!+ 16 +\!\!+ 25 = 1491625 \tag{15}$$

That would be in contrast to using [], indicating a sequence to be summarized:

$$\begin{aligned}\Concat_{i=1}^{5} [i^2] &= [1] +\!\!+ [4] +\!\!+ [9] +\!\!+ [16] +\!\!+ [25] \\ &= [1,4,9,16,25]\end{aligned} \tag{16}$$

Table 3: Operation of infix concatenation on various datatypes.

Strings:	$\text{“}a_1a_2\ldots a_m\text{”} +\!\!+ \text{“}b_1b_2\ldots b_n\text{”} = \text{“}a_1a_2\ldots a_mb_1b_2\ldots b_n\text{”}$
Lists:	$\{a_1,a_2,\ldots,a_m\} +\!\!+ \{b_1,b_2,\ldots,b_n\} = \{a_1,a_2,\ldots,a_m,b_1,b_2,\ldots,b_n\}$
Sequences:	$[a_1,a_2,\ldots,a_m] +\!\!+ [b_1,b_2,\ldots,b_n] = [a_1,a_2,\ldots,a_m,b_1,b_2,\ldots,b_n]$

Table 4: Examples of using indexed concatenation with digits.

Fraction	Decimal	Overline Notation	Indexed Concatenation
$\frac{1}{3}$	$0.333\ldots$	$0.\overline{3}$	$0.\in^{\infty}3$
$\frac{1}{7}$	$0.142857142857\ldots$	$0.\overline{142857}$	$0.\in^{\infty}(142857)$
$\frac{200}{111}$	1.801801801	$1.\overline{801}$	$1.\in^{\infty}(801)$
	$0.010010001\ldots$		$0.\in_{i=1}^{\infty}\left(\left(\in_{j=1}^{i}0\right)1\right)$

Table 4 provides several examples of compressing digits with indexed concatenation. We note here that the common overline notation looks much simpler, and probably should be retained except in cases such as the last example, which is an irrational number.

5.3 Lists and Sequences

We now turn to indexed concatenation of lists and sequences. Some examples:

$$\begin{aligned}\in_{i=1}^{3}\{i,i^2\} &= \{1,1\} +\!\!+ \{2,4\} +\!\!+ \{3,9\} \\ &= \{1,1,2,4,3,9\}\end{aligned} \tag{17}$$

$$\begin{aligned}\in_{i=1}^{5}\{i,10-i\} &= \{1,9\} +\!\!+ \{2,8\} +\!\!+ \{3,7\} \\ +\!\!+\{4,6\} +\!\!+ \{5,5\} &= \{1,9,2,8,3,7,4,6,5,5\}\end{aligned} \tag{18}$$

Again, the concatenation of lists is a list, without adding nested levels of the list structure. Of course, to make a list with sublists, we can nest the lists that serve as input to the concatenation operator. A concatenation of lists with one level of sublists is a list with one level of sublists:

$$\begin{aligned}\in_{i=1}^{3}\{\{i,i^2\}\} &= \{\{1,1\}\} +\!\!+ \{\{2,4\}\} \\ +\!\!+\{\{3,9\}\} &= \{\{1,1\},\{2,4\},\{3,9\}\}\end{aligned} \tag{19}$$

With sequences, the only difference is that all sequence delimiters that remain after the concatenation will vanish if inside another structure, so subsequences expand back out without adding additional levels of nesting. (Outer sequence delimiters remain, so the following is a sequence, not a list.)

$$\begin{aligned}\in_{i=1}^{3}\in_{j=1}^{3}[ji] &= \in_{i=1}^{3}[i,2i,3i] = [1,2,3] \\ +\!\!+[2,4,6] +\!\!+ [3,6,9] &= [1,2,3,2,4,6,3,6,9]\end{aligned} \tag{20}$$

But invoking the disappearance of our sequence-delimiter when inside some other object, one can concatenate subsequences in an outer list:

$$\begin{aligned}\left\{\in_{j=1}^{3}\in_{j=1}^{3}[ji]\right\} &= \{[1,2,3,2,4,6,3,6,9]\} \\ &= \{1,2,3,2,4,6,3,6,9\}\end{aligned} \tag{21}$$

These can be nested to any level desired. So we have two ways to represent a list of integers having an internal pattern. For example,

$$\begin{aligned}\{1,1,1,1,1,2,2,2,2,1,1,1,1,1,2,2,2,2,1,&\\ 1,1,1,1,2,2,2,2\} \to \left\{\in^{5}[1],\in^{4}[2],\in^{5}[1],\right.&\\ \left.\in^{4}[2],\in^{5}[1],\in^{4}[2]\right\}&\\ \to \left\{\in^{3}\left[\in^{5}[1],\in^{4}[2]\right]\right\}&\end{aligned} \tag{22}$$

$$\begin{aligned}
&\{1,1,1,1,1,2,2,2,2,1,1,1,1,1,2,2,2,2,1,\\
&1,1,1,1,2,2,2,2\} \rightarrow \{1,1,1,1,1\} +\!\!+ \{2,2,2,2\}\\
&\{1,1,1,1,1\} +\!\!+ \{2,2,2,2\} +\!\!+ \{1,1,1,1,1\}\\
&+\!\!+ \{2,2,2,2\} \rightarrow \overset{5}{\Subset} \{1\} +\!\!+ \overset{4}{\Subset} \{2\}\\
&+\!\!+ \overset{5}{\Subset} \{1\} +\!\!+ \overset{4}{\Subset} \{2\} +\!\!+ \overset{5}{\Subset} \{1\} +\!\!+ \overset{4}{\Subset} \{2\}\\
&\rightarrow \overset{3}{\Subset} \left(\overset{5}{\Subset} \{1\} +\!\!+ \overset{4}{\Subset} \{2\} \right)
\end{aligned} \tag{23}$$

The above examples have primarily involved increasing values, increasing number of values, or both. However, because any formula based on the concatenation indices can be used, this notation can also represent sequences involving descending numbers of elements and descending values.

$$\overset{10}{\underset{i=1}{\Subset}} \{2^{10-i}\} = \{512,256,128,64,32,16,8,4,2,1\} \tag{24}$$

$$\begin{aligned}
&\overset{3}{\underset{n=0}{\Subset}} \left\{ \overset{3-n}{\underset{i=0}{\Subset}} \{3-i-n\} \right\}\\
&= \{\{3,2,1,0\},\{2,1,0\},\{1,0\},\{0\}\}
\end{aligned} \tag{25}$$

We can even produce Pascal's Triangle, grouped by rows, to any desired row. (Again, replacing the inner, {}, by the disappearing subsequence delimiters, [], would produce a single list.)

$$\begin{aligned}
&\overset{5}{\underset{n=0}{\Subset}} \left\{ \overset{n}{\underset{k=0}{\Subset}} \left\{ \frac{n!}{k!(n-k)!} \right\} \right\}\\
&= \{\{1\},\{1,1\},\{1,2,1\},\{1,3,3,1\},\\
&\quad \{1,4,6,4,1\},\{1,5,10,10,5,1\}\}
\end{aligned} \tag{26}$$

5.4 Matrices

A generalized matrix (a_{ij}), with $1 \le i \le m$, $1 \le j \le n$, can be written in IC notation as a concatenated list of concatenated lists:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_n \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\mathrm{m1}} & a_{\mathrm{m2}} & \cdots & a_{\mathrm{mn}} \end{pmatrix} = \overset{m}{\underset{i=1}{\Subset}} \left\{ \overset{n}{\underset{j=1}{\Subset}} \{a_{ij}\} \right\} \tag{27}$$

If the matrix has some additional structure, that could also be shown explicitly in the IC form. For example, in the matrix mechanics representation of quantum physics, the annihilation operator $\hat{a}$, which lowers the quantum state $|n\rangle$ to $|n-1\rangle$ for a one-dimensional harmonic oscillator, is given in the energy eigenbasis by an infinite matrix whose pattern that can be explained in words: "All diagonal elements are zero. The entries $\sqrt{n}$ appear in the first subdiagonal above the diagonal. All other elements are 0." Yes, that is clear. But how much better to give a mathematically precise definition in IC notation!

$$\begin{aligned}
&\begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & \sqrt{2} & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \sqrt{3} & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \sqrt{4} & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \sqrt{5} & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{6} & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}\\
&= \overset{\infty}{\underset{k=1}{\Subset}} \left\{ \left\{ \overset{k}{\Subset} [0], \sqrt{k}, \overset{\infty}{\Subset} [0] \right\} \right\}
\end{aligned} \tag{28}$$

Very concise. Similarly, the infinite matrix representations of the creation operator ($\hat{a}^\dagger$), the position ($\hat{x}$), and momentum ($\hat{p}$) operators, can all be written in nested IC form.

6 CANTOR'S ENUMERATION OF ALL FRACTIONS & THE FAILURE OF SET-BUILDER NOTATION

Finally, let's consider an example that will resonate with mathematicians: the IC notation construction of the first enumeration of the rationals, published in (Cantor, 1874). Georg Cantor, the father of transfinite mathematics, published his proof of the countability of the rationals in 1874, over 150 years ago. One might presume that in the intervening time someone would have come up with a concise, mathematically precise way to describe the process. Sadly, that

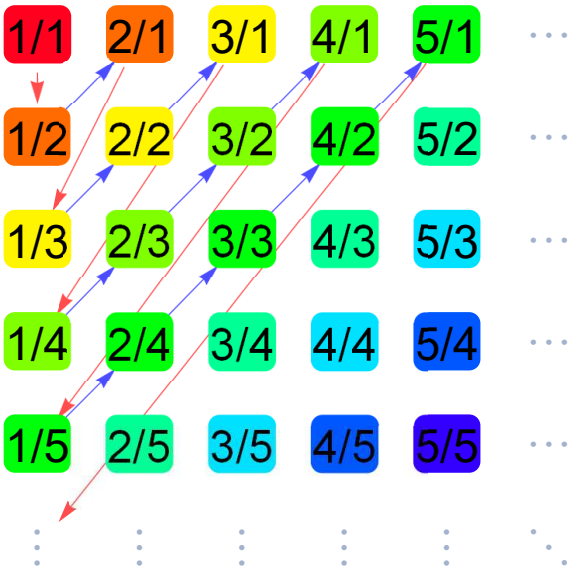


Figure 2: A traversal producing Cantor's enumeration of fractions.

expectation would be incorrect. Words to describe Cantor's method are easy:

1. Imagine an infinite table in which the first element of the first row is $\frac{1}{1}$, and then moving one column to the right always increases the numerator by 1, while moving one row down increases the denominator by 1.
2. Following any (infinite) row or column would mean never getting to the next one; instead, we traverse the array one (finite) diagonal at a time, starting from the upper left corner. Note that for all elements $\frac{n}{d}$ on a given diagonal, $n+d$ is a constant, and in fact, on diagonal number *diag*, $n+d = diag+1$.

Cantor actually used a "snake-like" traversal, winding back-and-forth on alternate diagonals, but for our purposes a "trace and retrace" pattern is preferable.

Figure 2 first appeared in (Caviness, 2011), and the code to generate it was updated in (Nachbar, 2023).

The following IC constructs the diagonals in the order Cantor visits them, but traverses each diagonal from left-to-right (by increasing numerator), as shown by the small blue arrows, before advancing to the next diagonal (longer red arrows). We do this by two nested IC objects, the outer specifying the diagonal *diag*, the inner giving the n^{th} element on the diagonal. If the nested list structure is not desired, the inner set of curly braces could be replaced by the disappearing square brackets, generating sequences that are simply spliced into a single list.

$$\begin{gathered}\left\{\left\{\frac{1}{1}\right\},\left\{\frac{1}{2},\frac{2}{1}\right\},\left\{\frac{1}{3},\frac{2}{2},\frac{3}{1}\right\},\right.\\ \left.\left\{\frac{1}{4},\frac{2}{3},\frac{3}{2},\frac{4}{1}\right\},\left\{\frac{1}{5},\frac{2}{4},\frac{3}{3},\frac{4}{2},\frac{5}{1}\right\}\cdots\right\}\\ \rightarrow \mathop{\Large\in}_{diag=1}^{\infty}\left\{\mathop{\Large\in}_{n=1}^{diag}\left\{\frac{n}{1+diag-n}\right\}\right\}\end{gathered} \tag{29}$$

This enumeration cannot be easily produced by conventional mathematical notation, such as using set-builder notation. For example,

$$\left\{\frac{p}{q} \mid p,q \in \mathbb{Z}^+\right\} \tag{30}$$

produces all fractions, but loses Cantor's diagonal ordering.

$$\left\{\frac{n}{diag-n+1} \mid diag \in \mathbb{Z}^+, n \in \{1,\ldots,diag\}\right\} \tag{31}$$

again produces all fractions, but assumes the user will not sort the resulting list (i.e., non-standard treatment for sets), and has no interest in grouping entries by diagonal. The Indexed Concatenation form practically writes itself, while set-builder notation requires significant mathematical gymnastics to produce the same result.

7 A MULTIPLY-NESTED EXAMPLE

One of the authors (Davis) created many infinite networks having some visual symmetry and patterned structure, identified vertices and edges for the beginning of each graph (again following some clear, repeating pattern), constructed its EDSL and then reduced it. One interesting case consists of nested, growing, interconnected pentagons, as shown in Figure 3.

The answer found, which completely defines this structure out to the 8th pentagon, was this interesting triply nested structure.

$$\begin{gathered}\left\{\mathop{\Large\in}_{n=1}^{8}\left[\{1,5n+4,5n+5\},\mathop{\Large\in}_{k=1}^{4}\right.\right.\\ \left.\left.\left[\mathop{\Large\in}_{i=1}^{n}[\{1\}],\{1,k+5n+5\}\right],\mathop{\Large\in}_{j=1}^{n-1}[\{1\}],\{\}\right]\right\}\end{gathered} \tag{32}$$

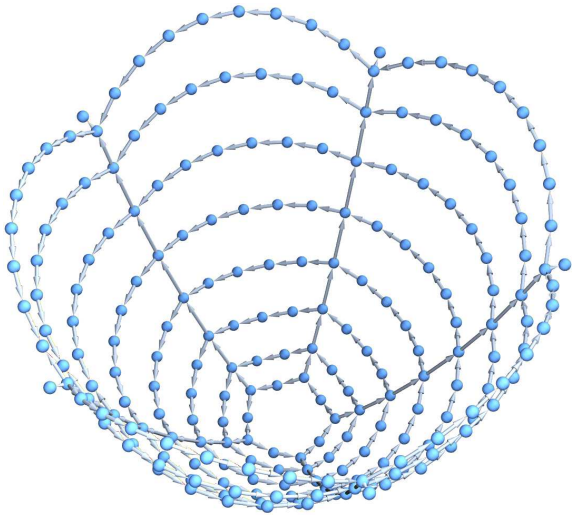


Figure 3: A multidimensional network with a triply nested concatenation.

Expanding the outer IC levels shows an fascinating feature of the IC notation. Although by construction all graph edges connect lower numbered vertices to higher numbered ones, so the EDSL contains only positive numbers, a 0 appeared in the final IC for $n = 1$, but only as the ending value for the index: $\in_{j=1}^{0}[\{1\}]$. Further examples showed that a 0 or even a negative ending value is perfectly legal for an indexed concatenation: That subsequence is simply omitted from the fully expanded list. Similarly, $\in_{i=1}^{1}[\{1\}]$ only expands out to a single copy of $\{1\}$.

$$\Big\{\{1,9,10\},\in_{i=1}^{1}[\{1\}],\{1,11\},\in_{i=1}^{1}[\{1\}],\{1,12\},\in_{i=1}^{1}[\{1\}],\{1,13\},\in_{i=1}^{1}[\{1\}],\{1,14\},\in_{j=1}^{0}[\{1\}],\{\},\{1,14,15\},\in_{i=1}^{2}[\{1\}],\{1,16\},\in_{i=1}^{2}[\{1\}],\{1,17\},\in_{i=1}^{2}[\{1\}],\{1,18\},\in_{i=1}^{2}[\{1\}],\{1,19\},\in_{j=1}^{1}[\{1\}],\{\},\{1,19,20\},\in_{i=1}^{3}[\{1\}],\{1,21\},\in_{i=1}^{3}[\{1\}],\{1,22\},\in_{i=1}^{3}[\{1\}],\{1,23\},\in_{i=1}^{3}[\{1\}],\{1,24\},\in_{j=1}^{2}[\{1\}],\{\},\ldots\Big\} \quad (33)$$

Clearly, once a pattern is detected, it is worth trying to extend it backwards in the list: it may apply even where not initially noticed. For reference, the fully expanded EDSL begins in this way:

$$\{\{1,9,10\},\{1\},\{1,11\},\{1\},\{1,12\},\{1\},\{1,13\},\{1\},\{1,14\},\{\},\{1,14,15\},\{1\},\{1\},\{1,16\},\{1\},\{1\},\{1,17\},\{1\},\{1\},\{1,18\},\{1\},\{1\},\{1,19\},\{1\},\{\},\{1,19,20\},\{1\},\{1\},\{1\},\{1,21\},\{1\},\{1\},\{1\},\{1,22\},\{1\},\{1\},\{1\},\{1,23\},\{1\},\{1\},\{1\},\{1,24\},\{1\},\{1\},\{\},\{1,24,25\},\{1\},\{1\},\{1\},\{1\},\{1,26\},\{1\},\{1\},\{1\},\{1\},\{1,27\},\{1\},\{1\},\{1\},\{1\},\{1,28\},\{1\},\{1\},\{1\},\{1\},\{1,29\},\{1\},\{1\},\{1\},\{\},\{1,29,30\},\ldots\} \quad (34)$$

8 A SELECTION OF NETWORKS

Here is a small collection of different networks we have successfully compressed, each shown together with its compressed edge difference set list (EDSL) in indexed concatenation form (IC). Many of these include nested concatenations. All of the networks can be extended by increasing the end value of the index variable in the outer indexed concatenation. Replacing the end value by ∞ results in an infinite network, without adding any complexity to the IC form shown.

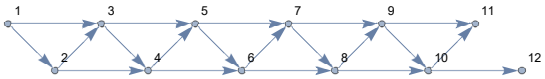
Figures 4 and 5 showcase the examples selected, together with their concatenated EDSLs. Figure 4 consists of examples of networks that can extend indefinitely in one dimension, whether or not a close view appears one- or two-dimensional. (Cases that are locally three-dimensional have also been found.) Figure 5 shows several higher-dimensional networks along with their EDSLs. Most of these visually appear to be two-dimensional (expanding in two directions), and all include at least two levels of indexed concatenation, one nested inside the other, although such nesting does not guarantee two-dimensionality, as can be seen from Figure 4.

One example (Figure 5d) is clearly three-dimensional (expanding in three directions as the outer index increases), and is summarized by a three-level nesting of indexed concatenations. Yet others we have investigated are too dense to "fit" in three dimensions. Figure 5f provides an example of this last type, which we refer to as growing "exponentially." At this point we have no clear connection between IC nesting and the "growth dimensionality" of the graph represented.

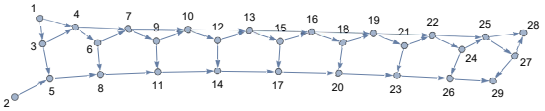
In each of these cases, a complex geometric pattern is captured mathematically by indexed concatenation.

(a) $\{\in^{10}[\{1\}]\}$

(b) $\{\in^{5}[\{1\},\{\}]\}$

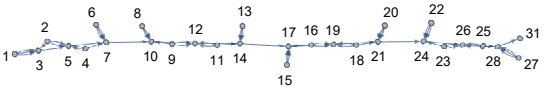


(c) $\{\in^{10}[\{1,2\}]\}$

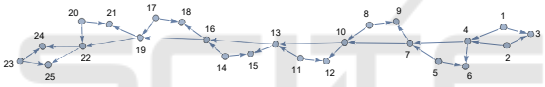


(d) $\{\in^{9}[\{2,3\},\{3\},\{1,2\}]\}$

(e) $\{\in^{3}[\in_{k=1}^{3}[\{2k-1,2k\}],\in^{2}[\{1,2\},\in^{2}[\{\}]]]\}$

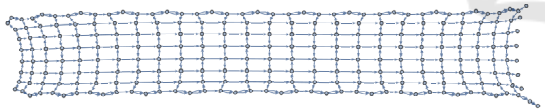


(f) $\{\in^{4}[\{2,2,2\},\{1,3,3\},\{2\},\{1,1,3\},\{2\},\{1,1,1\},\{3\}]\}$

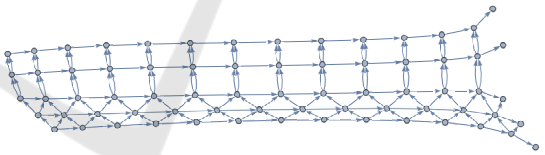


(g) $\{\in^{8}[\{2,3\},\{1,2\},\{\}]\}$

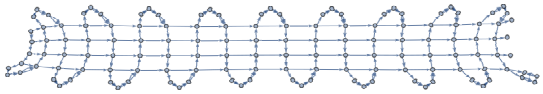
(h) $\{\in^{7}[\{1,6,8\},\{\},\{\in^{4}[2]\},\in^{2}[\{1,\in^{3}[3]\},\{\}]]\}$



(i) $\{\in_{n=0}^{25}[\in_{k=1}^{7}[\{1,18-2k\}],\in_{k=8}^{9}[\{1,1\}]]\}$

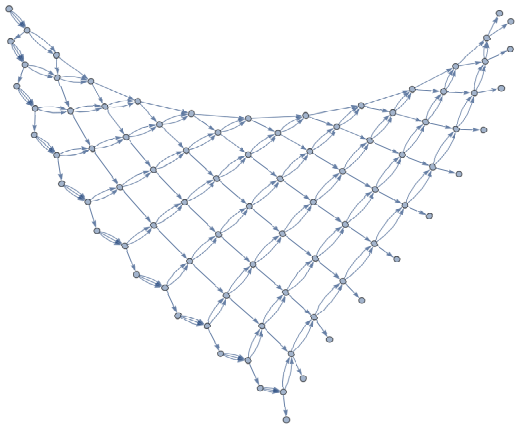


(j) $\{\in^{13}[\in^{2}[\{1,7,8\}],\in^{2}[\{1,1,7\}],\{7\},\{\},\{\}]\}$

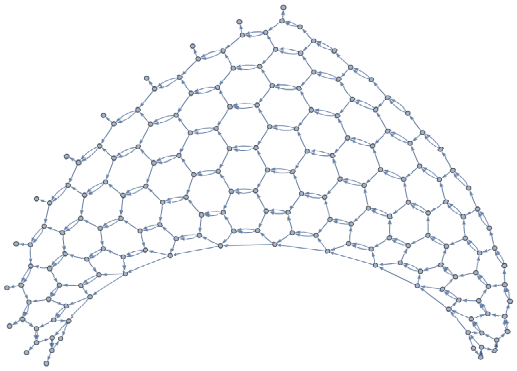


(k) $\{\in_{n=0}^{18}[\in_{k=1}^{4}[\{1,12-2k\}],\in_{k=5}^{7}[\{1,1\}]]\}$

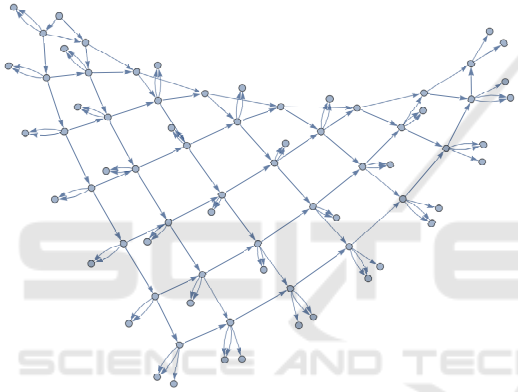
Figure 4: A selection of networks and their concatenated EDSLs.



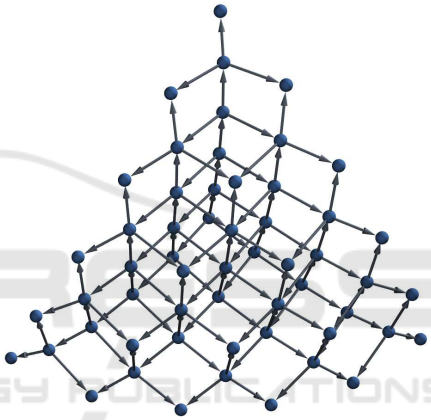
(a) $\{\in_{i=1}^{10}[\{1,1,1\},\{1,1,i+1\},\ \in^{i-1}[\{1,1,i+2\}],\{i+2,i+3\}]\}$



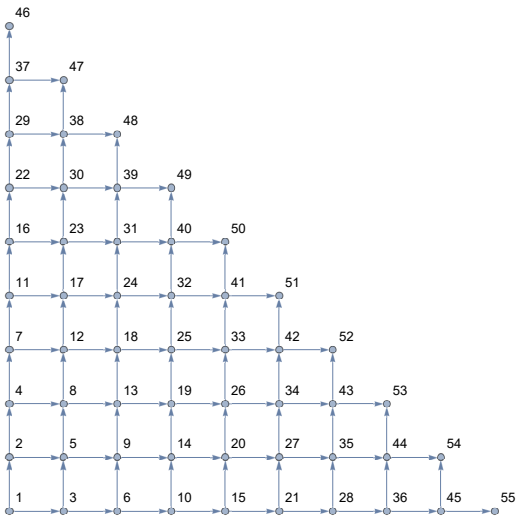
(b) $\{\in_{k=1}^{12}[\{2k-1,2k+1\},\in^{k}[\{1,1\},\{1,2k\}]]\}$



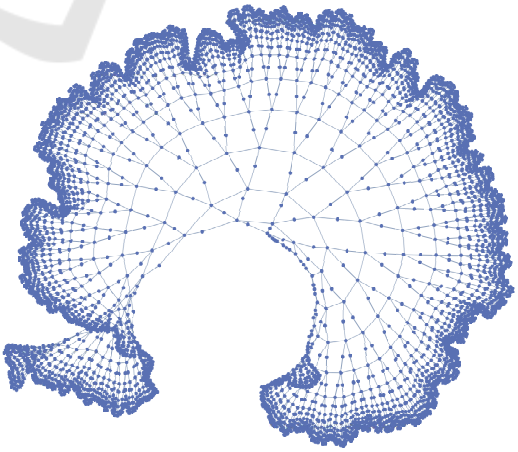
(c) $\{\in_{k=1}^{8}[\in^{k-1}[\{\},\{1,1,2,2k\}],\ \{\},\{2k,2(k+1)\}]]\}$



(d) $\{\in_{n=1}^{5}[\in_{i=1}^{n}[\in^{i}[\{z=\frac{1}{2}n(n+1),\ z+i,z+i+1\}]]]\}$



(e) $\{\in_{k=1}^{9}[\in^{k}[\{k,k+1\}]]\}$



(f) $\left\{\in_{k=1}^{10}\left[\{2^k-1,2^k,2^k+1\},\in^{2^k+1}\left[\{2^k+1\}\right],\{1\},\in_{j=1}^{2^k-1}\left[\{1,j+2^k-1,j+2^k\}\right]\right]\right\}$

Figure 5: A selection of higher-dimensional networks and their EDSLs.

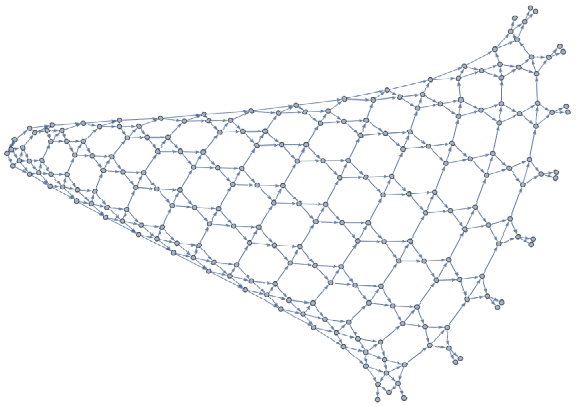


Figure 6: A two-dimensional network with multiple possible summarizations.

One network may have multiple possible summarizations, as is demonstrated in Figure 6. This network's EDSL could be concatenated as either

$$\left\{\mathop{\Large\in}_{n=1}^{7}[\{1,2,3\},\{3n+1,3n+2\},\{1,2\},\right.$$
$$\{3n-1,3n+1\},\mathop{\Large\in}^{n-1}\left[\mathop{\Large\in}^{2}[\{1,2\}],\{3n,3n+1\}\right],$$
$$\{3n+2,3n+3\},\{1,2,3\},\{3n+3,3n+4\},\{1,2\},$$
$$\{3n+1,3n+3\},\mathop{\Large\in}^{n-1}\left[\mathop{\Large\in}^{2}[\{1,2\}],\{3n+2,\right. \tag{35}$$
$$\left.\left.3n+3\}],\mathop{\Large\in}^{m}\{1,2\},\{1,3n+6\},\{3n+2,3n+3\}\right]\right\}$$

or, alternatively, as

$$\left\{\mathop{\Large\in}_{n=1}^{7}\left[\mathop{\Large\in}_{m=0}^{1}[\{1,2,3\},\{2m+3n+1,2m+3n+2\},\right.\right.$$
$$\{1,2\},\{2m+3n-1,2m+3n+1\},$$
$$\mathop{\Large\in}^{n-1}\left[\mathop{\Large\in}^{2}[\{1,2\}],\{2m+3n,2m+3n+1\}\right] \tag{36}$$
$$\left.\left.\mathop{\Large\in}^{m}[\{1,2\},\{1,3n+6\}],\{3n+2,3n+3\}\right]\right]\right\}$$

9 CONCLUSIONS

Throughout the history of mathematics and science, good notation has contributed to increased understanding, greater insights, and more rapid and successful advancement in the field under study. Examples include the decimal system (replacing previous use of Roman numerals and dependence on fractions), Leibnitz' representation of the derivative of y with respect to x as $\frac{dy}{dx}$, vector notation to conveniently describe relationships in three-dimensional space, the 4-vector notation in Special Relativity to include the time coordinate, and the summation convention used in tensor calculus and General Relativity. Good notation facilitates good work.

The Indexed Concatenation Notation, following the lead of other indexed operators, allows compact representations of strings, lists, sequences, integer lists, matrices, and other similar expressions, examples of which we have considered above. It capitalizes on repetitions, periodicities and other patterns to compress information into a reduced form. It may be hoped that such notation summarizing internal patterns may not only serve as a new means of information compression, but also contribute to simpler and more insightful use of the data.

10 PROSPECTS

Where do we go from here?

1. Fine-tune automated manipulation of IC notation. Our code to partially or completely expand out Indexed Concatenation lists and sequences works for any finite nesting level. However, we have encountered two main shortcomings:
   (a) The reverse direction is harder; our algorithm currently identifies and reduces 70% of the sample cases attempted.
   (b) Treatment of strings and integers has not yet been implemented in either direction.
2. Publish a follow-up paper showcasing the algorithm itself once the code is further refined.
3. Construct and/or find other interesting graphs to reduce.
4. Construct an algorithm to generate an IC summary of the complete list of the edges of a graph from its EDSL (edge differences set list).
5. What else it might be used for?
   (a) Musical scores of various music genres?
   (b) Summarizing DNA sequences?

Our research team is actively pursuing these options, in particular the application of the IC notation to music and bioinformatics and the refinment of a network summarization algorithm.

ACKNOWLEDGEMENTS

The authors acknowledge the generous support of the Academic Research Committee of Southern Adventist University for funding this project and related work which led to it.

REFERENCES

Campbell, J. (1971). Comparative survey of programming languages. *COMPUTING AS A LANGUAGE*, pages 391–484.

Cantor, G. (1874). Über eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen. *Journal für die reine und angewandte Mathematik*, 77:258–262.

Caviness, K. (2011). Indexing strings and rulesets. *The Mathematica Journal*, 13.

de Bruijn, N. G. (1977). Notation for concatenation. Technical report, Technische Hogeschool Eindhoven.

Maheshwari, U. and Smid, M. (2024). *Introduction to Theory of Computation*. School of Computer Science, Carleton University.

Nachbar, R. (2023). Reply to changed methods for displaying graphs (networks). https://community.wolfram.com/groups/-/m/t/2862755?p_p_auth=iQ7ZaUTu.

Rehmann, U. (2020). Sequence. https://encyclopediaofmath.org/in-dex.php?title=Sequence&oldid=48671.

Sipser, M. (2012). *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition.

WsCubeTech (2025). C program to concatenate two strings (5 ways). https://www.wscubetech.com/resources/c-programming/programs/concatenate-two-strings.