Ben-Gen: A Symbolic Pattern Language for Representing Astronomically Large Numbers

Ayansh Macharla

2026

Abstract

We introduce *Ben-Gen*, a symbolic pattern language designed for the concise representation of astronomically large integers exhibiting regular structural patterns. Ben-Gen is not a general arithmetic system nor a data compression method. Instead, it provides a formal notation for describing growth-oriented numeric structures such as zero-dominant expansions and structured concatenations. The system is motivated by educational, visualization, and procedural generation needs, particularly where explicit digit-level representations are infeasible. We present the syntax, semantics, axiomatic foundation, examples, limitations, and potential applications of the language.

1 Introduction

Human intuition struggles with numbers beyond everyday scale. Quantities encountered in astronomy, computation, and theoretical mathematics often exceed practical digit-level representation. While scientific notation and power towers provide compact forms, they obscure internal digit structure and pattern behavior.

This paper presents *Ben-Gen*, a symbolic notation system developed from an attempt to reason about extremely large numbers during routine academic problem-solving. The language focuses on expressing numeric *patterns* rather than explicit values, enabling structured descriptions of growth without requiring full expansion.

2 Existing Notations

Several systems attempt to describe large numbers succinctly:

- **Scientific notation** (1×10^n) compactly expresses magnitude but hides digit structure.
- **Knuth's up-arrow notation** describes rapid growth but is non-intuitive for learners.
- **Run-length encoding** compresses repetition but is not inherently mathematical.

Ben-Gen differs by prioritizing *symbolic pattern generation* over numerical evaluation.

3 Motivation

The motivation for Ben-Gen arose from the need to conceptualize very large numbers without materializing them. During exam preparation, exploratory reasoning led to the idea of representing growth patterns symbolically. This evolved into a formal system intended to support:

- Mathematical education
- Visualization of exponential growth
- Procedural numeric generation

4 Formal Definition

4.1 Syntax

A Ben-Gen expression consists of numeric symbols combined with pattern operators.

Definition 1 (Ben-Gen Expression)**.** *A Ben-Gen expression is defined recursively as:*

$$E ::= N \mid (E) \mid E \circ E$$

where N is a finite decimal numeral and $\circ$ is a Ben-Gen operator.

4.2 Operators

Ben-Gen defines the following primary operators:

- $\odot$: Zero-dominant expansion operator
- $\otimes$: Structured multiplicative concatenation operator
- $\oplus$: Symbolic concatenation operator

5 Axiomatic Foundation

Axiom 1 (Symbolic Determinism)**.** *Every valid Ben-Gen expression corresponds to exactly one pattern under the defined semantics.*

Axiom 2 (Pattern-First Principle)**.** *Ben-Gen expressions describe generative numeric patterns rather than explicit numeric values.*

Axiom 3 (Single-Digit Atomicity)**.** *Single-digit numerals are treated as indivisible pattern units.*

Axiom 4 (Dual Multiplicative Semantics)**.** *Ben-Gen supports two forms of multiplication:*

(a) Arithmetic multiplication for numerical interpretation.

(b) Pattern multiplication for structural growth.

Axiom 5 (Decimal Interpretation)**.** *Decimal numerals act as pattern modifiers and do not imply floating-point arithmetic.*

Axiom 6 (Finite Description, Unbounded Expansion)**.** *Every Ben-Gen expression has finite length, though its expansion may be arbitrarily large.*

Axiom 7 (Non-Universality)**.** *Ben-Gen does not represent arbitrary digit sequences.*

Axiom 8 (Expansion Consistency)**.** *Partial expansions must be consistent with full expansions.*

Axiom 9 (Operator Closure)**.** *All Ben-Gen operators map valid expressions to valid expressions.*

[11pt]article
amsmath, amssymb, amsthm enumitem hyperref geometry margin=1in
Axiom Definition Example
Ben-Gen: A Symbolic Pattern Language for Representing Astronomically Large Numbers Ayansh Macharla 2026

Abstract

We introduce *Ben-Gen*, a symbolic pattern language designed for the concise representation of astronomically large integers exhibiting regular structural patterns. Ben-Gen is not a general arithmetic system nor a data compression method. Instead, it provides a formal notation for describing growth-oriented numeric structures such as zero-dominant expansions and structured concatenations. The system is motivated by educational, visualization, and procedural generation needs, particularly where explicit digit-level representations are infeasible. We present the syntax, semantics, axiomatic foundation, examples, limitations, and potential applications of the language.

6 Introduction

Human intuition struggles with numbers beyond everyday scale. Quantities encountered in astronomy, computation, and theoretical mathematics often exceed practical digit-level representation. While scientific notation and power towers provide compact forms, they obscure internal digit structure and pattern behavior.

This paper presents *Ben-Gen*, a symbolic notation system developed from an attempt to reason about extremely large numbers during routine academic problem-solving. The language focuses on expressing numeric *patterns* rather than explicit values, enabling structured descriptions of growth without requiring full expansion.

7 Existing Notations

Several systems attempt to describe large numbers succinctly:

- **Scientific notation** (1×10^n) compactly expresses magnitude but hides digit structure.
- **Knuth's up-arrow notation** describes rapid growth but is non-intuitive for learners.
- **Run-length encoding** compresses repetition but is not inherently mathematical.

Ben-Gen differs by prioritizing *symbolic pattern generation* over numerical evaluation.

8 Motivation

The motivation for Ben-Gen arose from the need to conceptualize very large numbers without materializing them. During exam preparation, exploratory reasoning led to the idea of representing growth patterns symbolically. This evolved into a formal system intended to support:

- Mathematical education
- Visualization of exponential growth
- Procedural numeric generation

9 Formal Definition

9.1 Syntax

A Ben-Gen expression consists of numeric symbols combined with pattern operators.

Definition 2 (Ben-Gen Expression)**.** *A Ben-Gen expression is defined recursively as:*

$$E ::= N \mid (E) \mid E \circ E$$

where N is a finite decimal numeral and $\circ$ is a Ben-Gen operator.

9.2 Operators

Ben-Gen defines the following primary operators:

- $\odot$: Zero-dominant expansion operator
- $\otimes$: Structured multiplicative concatenation operator
- $\oplus$: Symbolic concatenation operator

10 Axiomatic Foundation

Axiom 10 (Symbolic Determinism)**.** *Every valid Ben-Gen expression corresponds to exactly one pattern under the defined semantics.*

Axiom 11 (Pattern-First Principle)**.** *Ben-Gen expressions describe generative numeric patterns rather than explicit numeric values.*

Axiom 12 (Single-Digit Atomicity)**.** *Single-digit numerals are treated as indivisible pattern units.*

Axiom 13 (Dual Multiplicative Semantics)**.** *Ben-Gen supports two forms of multiplication:*

(a) Arithmetic multiplication for numerical interpretation.

(b) Pattern multiplication for structural growth.

Axiom 14 (Decimal Interpretation)**.** *Decimal numerals act as pattern modifiers and do not imply floating-point arithmetic.*

Axiom 15 (Finite Description, Unbounded Expansion)**.** *Every Ben-Gen expression has finite length, though its expansion may be arbitrarily large.*

Axiom 16 (Non-Universality)**.** *Ben-Gen does not represent arbitrary digit sequences.*

Axiom 17 (Expansion Consistency)**.** *Partial expansions must be consistent with full expansions.*

Axiom 18 (Operator Closure)**.** *All Ben-Gen operators map valid expressions to valid expressions.*

11 Semantics

11.1 Zero-Dominant Expansion

Definition 3. *For integers a, b:*

$$a \odot b \rightarrow \text{“1 followed by } a \times b \text{ zeros”}$$

11.2 Structured Multiplicative Concatenation

Definition 4. *For two-digit numbers ab and cd:*

$$ab \otimes cd \to (a \times c)(b \times d)(ab \times c)(ab \times d)$$

concatenated symbolically.

11.3 Symbolic Concatenation

Definition 5.

$$x \oplus y \to \textit{symbolic concatenation of } x \textit{ and } y$$

12 Examples

Example 1.

$$10 \odot 10 \Rightarrow 1 \textit{ followed by } 10 \textit{ zeros}$$

Example 2.

$$12 \otimes 34 \Rightarrow (1 \times 3)(2 \times 4)(12 \times 3)(12 \times 4)$$

Example 3.

$$100 \oplus 10 \Rightarrow \textit{symbolic pattern ``10010''}$$

13 Limitations

Ben-Gen is subject to inherent constraints:

- It cannot encode arbitrary or random data.
- It is not a compression algorithm.
- It does not replace conventional arithmetic.

These limitations arise from information-theoretic bounds and are intrinsic to the design.

14 Applications

14.1 Education

Ben-Gen provides intuitive visualization of exponential growth and large numbers.

14.2 Visualization

Symbolic patterns enable representation of otherwise intractable quantities.

14.3 Procedural Generation

Structured numeric patterns can be generated without materialization.

15 Future Work

Future directions include:

- **Benma**: a domain-specific language for Ben-Gen expressions
- Symbolic evaluation engines
- Educational visualization tools

16 Conclusion

Ben-Gen introduces a formal symbolic language for describing large structured integers through patterns. While not universal or compressive, it offers clarity, educational value, and conceptual accessibility for astronomical-scale numbers. The language demonstrates how symbolic pattern thinking can complement traditional numerical representations.