Guarded Language Operators as Contractions in a Length-Based Ultrametric Space

Hristo Hristov , Atanas Ilchev * , Hristina Kulina , Boyan Zlatanov

Posted Date: 14 April 2026

doi: 10.20944/preprints202604.0952.v1

Keywords: formal languages; ultrametric spaces; wrapping operators; contraction mappings; Banach fixed point theorem; Picard iteration; recursive language equations; document validation

 doi:10.20944/preprints202604.0952.v1

Article

Guarded Language Operators as Contractions in a Length-Based Ultrametric Space

Hristo Hristov [1], Atanas Ilchev [2,*], Hristina Kulina [2] and Boyan Zlatanov [2]

[1] Department of Software Technology, Faculty of Mathematics and Informatics, University of Plovdiv Paisii Hilendarski, 4000 Plovdiv, Bulgaria
[2] Department of Mathematical Analysis, Faculty of Mathematics and Informatics, University of Plovdiv Paisii Hilendarski, 4000 Plovdiv, Bulgaria
* Correspondence: atanasilchev@uni-plovdiv.bg

Abstract

We study a class of wrapping operators acting on the space of formal languages over a fixed finite alphabet. The underlying space is equipped with a length-based ultrametric, in which two languages are close whenever they coincide on all sufficiently short words. We prove that every wrapping operator generated by a finite family of guards with positive total guard length is a contraction. As a consequence, Banach's contraction principle yields existence and uniqueness of a fixed point for the corresponding recursive language equation, together with convergence of the Picard iteration from an arbitrary initial language. We also obtain an explicit quantitative estimate for the rate of convergence. This makes it possible to determine how many iterations are sufficient to recover the fixed point correctly on all words up to a prescribed length. Several examples illustrate the theory, including operators with different guard lengths and a case showing that convergence in the length-based ultrametric does not coincide with set-theoretic convergence. An application to recursive structures and document validation is also presented, including recursive data formats, abstract syntax trees, and a restricted fragment of JSON schemas. The results provide a formal foundation for validation together with explicit bounds for correctness on inputs of bounded length.

Keywords: formal languages; ultrametric spaces; wrapping operators; contraction mappings; Banach fixed point theorem; Picard iteration; recursive language equations; document validation

1. Introduction

The study of fixed point equations on structured spaces is a central theme in both pure and applied mathematics. Since Banach's classical work [1], contraction mappings on complete metric spaces have provided a powerful framework for proving existence and uniqueness of solutions to recursive equations. One of the main advantages of this approach is its constructive nature: the fixed point is obtained as the limit of the Picard iteration sequence, and the contraction constant yields explicit information about the rate of convergence.

In the semantics of programming languages, metric and ultrametric ideas have played an important role in the analysis of recursive definitions. In particular, de Bakker and Zucker [2] showed that complete metric methods provide a natural setting for denotational semantics of concurrency, while America and Rutten [3] developed a categorical treatment of complete metric spaces and contraction-based fixed point constructions for recursive domain equations. A recurring idea in this line of work is that contractivity is closely related to guardedness: recursion becomes well behaved when each recursive occurrence is separated from the surrounding structure by a sufficient amount of observable information.

The present paper develops an analogous point of view in the setting of formal languages. Let Σ be a finite alphabet and let $\mathcal{L}$ denote the set of all formal languages over Σ. We equip $\mathcal{L}$ with a length-based ultrametric in which two languages are close whenever they coincide on all words up

 doi:10.20944/preprints202604.0952.v1

to a sufficiently large length. This makes $\mathcal{L}$ a natural environment for studying recursive language constructions by fixed point methods.

We introduce a class of wrapping operators of the form

$$T_{S,G}(L) = S \cup \bigcup_{r=1}^{p} u_r L v_r,$$

where $S \subseteq \Sigma^*$ is a fixed seed language and

$$G = \{(u_r, v_r) : r = 1, \ldots, p\} \subseteq \Sigma^* \times \Sigma^*$$

is a finite family of guards. The key assumption is that the total guard length satisfies

$$\min_{1 \le r \le p} \big(|u_r| + |v_r|\big) \ge m$$

for some integer $m \ge 1$. Under this hypothesis, every wrapped word carries a mandatory outer layer of total length at least m before the recursive core can be reached. This is the feature that yields contractivity.

Our main result shows that every such wrapping operator is a contraction on $(\mathcal{L}, d)$ with contraction factor 2^{-m}. Banach's fixed point theorem then implies that the corresponding recursive language equation

$$L = T_{S,G}(L)$$

admits a unique solution in $\mathcal{L}$. Moreover, the Picard iterates converge to this solution from any initial language, and the convergence is quantitatively controlled by the guard length. In particular, when the iteration starts from the empty language, one obtains an explicit bound ensuring correctness on all words up to any prescribed length.

Beyond its theoretical interest, the framework developed in this paper has a natural interpretation in computer science. In particular, recursive data structures, document formats with nested syntax, and schema-based validation mechanisms can be modeled using wrapping operators. In this setting, the contractive nature of guarded recursion provides not only well-defined semantics, but also explicit bounds on the number of steps required to validate inputs of bounded size. This connects the abstract fixed-point theory developed here with practical questions in parsing, serialization, and validation of recursive data.

The paper is organized as follows. Section 2 recalls the required preliminaries on formal languages, the length-based ultrametric on $\mathcal{L}$, and Banach's contraction principle. Section 3 introduces wrapping operators generated by finite families of guards and establishes the main fixed point and convergence results. Section 4 contains several examples illustrating the theory, including a case in which ultrametric convergence differs from set-theoretic convergence. Section 5 presents an application to recursive document validation.

2. Preliminaries

Throughout the paper, Σ denotes a fixed finite alphabet and Σ^* the set of all finite words over Σ, including the empty word ε. For a word $w \in \Sigma^*$, its length is denoted by $|w|$.

Definition 1 ([4,5]). *A formal language over Σ is any subset $L \subseteq \Sigma^*$.*

We denote by $\mathcal{L}$ the set of all formal languages over Σ.

 doi:10.20944/preprints202604.0952.v1

Definition 2 ([4,5]). *Each language $L \in \mathcal{L}$ is identified with its characteristic function*

$$\mathbf{1}_L : \Sigma^* \to \{0,1\}, \qquad \mathbf{1}_L(w) = \begin{cases} 1, & w \in L, \\ 0, & w \notin L. \end{cases}$$

For $u \in \Sigma^*$ and $L \subseteq \Sigma^*$, we define left and right concatenation by

$$uL := \{uw : w \in L\}, \qquad Lu := \{wu : w \in L\}.$$

Definition 3 ([2]). *For any two formal languages $L_1, L_2 \in \mathcal{L}$ define a distance between L_1 and L_2 by*

$$d(L_1, L_2) = \begin{cases} 0, & L_1 = L_2, \\ 2^{-k}, & k = \min\{\, |w| : \mathbf{1}_{L_1}(w) \neq \mathbf{1}_{L_2}(w) \,\}. \end{cases}$$

Lemma 1. *Let $L_1, L_2 \in \mathcal{L}$ be two formal languages. If $d(L_1, L_2) = 2^{-k}$, then L_1 and L_2 coincide on all words of length strictly less than k, and there exists a word w with $|w| = k$ such that*

$$\mathbf{1}_{L_1}(w) \neq \mathbf{1}_{L_2}(w).$$

This follows directly from Definition 3; see [2].

Theorem 1 ([2]). *The function d is an ultrametric on $\mathcal{L}$, i.e.,*

$$d(L_1, L_3) \leq \max\{d(L_1, L_2), d(L_2, L_3)\} \quad \text{for all } L_1, L_2, L_3 \in \mathcal{L}.$$

Definition 4. *A sequence $(L_n) \subseteq \mathcal{L}$ is Cauchy if for every $k \in \mathbb{N}$ there exists N such that*

$$\mathbf{1}_{L_n}(w) = \mathbf{1}_{L_m}(w) \quad \text{for all } |w| \leq k \text{ and all } n, m \geq N.$$

This is the standard notion of a Cauchy sequence in the metric space $(\mathcal{L}, d)$; see [2] for the underlying metric structure.

Theorem 2 ([2]). *The ultrametric space $(\mathcal{L}, d)$ is complete.*

Definition 5 ([1]). *Let (X, d) be a metric space. A mapping $T : X \to X$ is a contraction if there exists $c \in (0,1)$ such that*

$$d(Tx, Ty) \leq c\, d(x, y) \quad \text{for all } x, y \in X.$$

Definition 6 ([1]). *An element $x^* \in X$ is a fixed point of T if $T(x^*) = x^*$.*

Theorem 3 ([1]). *Let (X, d) be a complete metric space and let $T : X \to X$ be a contraction. Then T admits a unique fixed point x^*, and the iterates*

$$x_{n+1} = T(x_n)$$

converge to x^* for every $x_0 \in X$.

3. Main Result

Definition 7. *Let Σ be a finite alphabet. A finite set*

$$G = \{(u_r, v_r) : r = 1, \ldots, p\} \subseteq \Sigma^* \times \Sigma^*$$

is called a finite set of guards.

 doi:10.20944/preprints202604.0952.v1

Definition 8. *Let Σ be a finite alphabet, let $S \subseteq \Sigma^*$, and let*

$$G = \{(u_r, v_r) : r = 1, \dots, p\} \subseteq \Sigma^* \times \Sigma^*$$

be a finite set of guards.

We define an operator

$$T_{S,G} : \mathcal{L} \to \mathcal{L}$$

by

$$T_{S,G}(L) = S \cup \bigcup_{r=1}^{p} u_r L v_r, \qquad L \in \mathcal{L}.$$

We call $T_{S,G}$ a wrapping operator generated by the seed S and the guards G.

Definition 9. *Let*

$$G = \{(u_r, v_r) : r = 1, \dots, p\} \subseteq \Sigma^* \times \Sigma^*.$$

We say that G has guard length at least $m \geq 1$ if

$$\min_{1 \leq r \leq p} \left(|u_r| + |v_r| \right) \geq m.$$

Remark 1. *When no ambiguity arises, we write T instead of $T_{S,G}$.*

Lemma 2. *Let $u, v \in \Sigma^*$. Then for all $A, B \in \mathcal{L}$,*

$$d(uAv,\ uBv) \leq 2^{-(|u|+|v|)}\, d(A, B).$$

Proof. We consider two cases.

Case 1: $A = B$. Then $uAv = uBv$, hence

$$d(uAv, uBv) = 0 \leq 2^{-(|u|+|v|)} d(A, B).$$

Case 2: $A \neq B$. Let

$$k = \min\{\, |w| : \mathbf{1}_A(w) \neq \mathbf{1}_B(w) \,\}.$$

Then

$$d(A, B) = 2^{-k}.$$

By Lemma 1, the languages A and B coincide on all words of length strictly less than k, and there exists a word $w_0 \in \Sigma^*$ with $|w_0| = k$ such that

$$\mathbf{1}_A(w_0) \neq \mathbf{1}_B(w_0).$$

We claim that uAv and uBv coincide on all words of length strictly less than

$$k + |u| + |v|.$$

Let $x \in \Sigma^*$ satisfy

$$|x| < k + |u| + |v|.$$

If x does not begin with u, then clearly

$$x \notin uAv \quad \text{and} \quad x \notin uBv,$$

so

$$\mathbf{1}_{uAv}(x) = \mathbf{1}_{uBv}(x) = 0.$$

 doi:10.20944/preprints202604.0952.v1

Assume now that x begins with u. Then there exists a unique word $y \in \Sigma^*$ such that

$$x = uy.$$

From

$$|x| < k + |u| + |v|$$

we obtain

$$|y| < k + |v|.$$

If y does not end with v, then

$$y \notin Av \quad \text{and} \quad y \notin Bv,$$

hence

$$x \notin uAv \quad \text{and} \quad x \notin uBv.$$

Assume finally that y ends with v. Then there exists a unique word $z \in \Sigma^*$ such that

$$y = zv.$$

Since

$$|y| < k + |v|,$$

it follows that

$$|z| < k.$$

Therefore

$$\mathbf{1}_A(z) = \mathbf{1}_B(z),$$

because A and B coincide on all words of length strictly less than k. Hence

$$z \in A \iff z \in B,$$

which implies

$$y \in Av \iff y \in Bv$$

and therefore

$$x \in uAv \iff x \in uBv.$$

We have shown that uAv and uBv coincide on all words of length strictly less than

$$k + |u| + |v|.$$

Thus

$$d(uAv, uBv) \leq 2^{-(k+|u|+|v|)} = 2^{-(|u|+|v|)}\, 2^{-k} = 2^{-(|u|+|v|)}\, d(A, B).$$

This completes the proof. □

Lemma 3. *For all $A, B, S \in \mathcal{L}$,*

$$d(A \cup S,\ B \cup S) \leq d(A, B).$$

Proof. If $A = B$, then $A \cup S = B \cup S$ and $d(A \cup S, B \cup S) = 0 \leq d(A, B)$.

Let $A \neq B$ and let

$$k = \min\{\, |w| : \mathbf{1}_A(w) \neq \mathbf{1}_B(w) \,\} \geq 0.$$

Then $d(A, B) = 2^{-k}$.

We will prove that for all words w with $|w| < k$, we have $\mathbf{1}_{A \cup S}(w) = \mathbf{1}_{B \cup S}(w)$.

Let $|w| < k$. From the definition of k, we know that $\mathbf{1}_A(w) = \mathbf{1}_B(w)$. We consider two subcases:

 doi:10.20944/preprints202604.0952.v1

Case 1: $\mathbf{1}_A(w) = \mathbf{1}_B(w) = 1$. Then $w \in A$ and $w \in B$. Consequently, $w \in A \cup S$ and $w \in B \cup S$, so $\mathbf{1}_{A\cup S}(w) = 1 = \mathbf{1}_{B\cup S}(w)$.

Case 2: $\mathbf{1}_A(w) = \mathbf{1}_B(w) = 0$. Then $w \notin A$ and $w \notin B$. To determine whether $w \in A \cup S$, we need to check whether $w \in S$. We have:

$$\mathbf{1}_{A\cup S}(w) = \max\{\mathbf{1}_A(w), \mathbf{1}_S(w)\} = \max\{0, \mathbf{1}_S(w)\} = \mathbf{1}_S(w),$$
$$\mathbf{1}_{B\cup S}(w) = \max\{\mathbf{1}_B(w), \mathbf{1}_S(w)\} = \max\{0, \mathbf{1}_S(w)\} = \mathbf{1}_S(w).$$

Therefore $\mathbf{1}_{A\cup S}(w) = \mathbf{1}_S(w) = \mathbf{1}_{B\cup S}(w)$.

Thus we have proved that for all w with $|w| < k$, we have $\mathbf{1}_{A\cup S}(w) = \mathbf{1}_{B\cup S}(w)$. This means that the first possible disagreement between $A \cup S$ and $B \cup S$ occurs at length at least k. Consequently,

$$d(A \cup S, B \cup S) \leq 2^{-k} = d(A, B).$$

□

Lemma 4. *Let Σ be a finite alphabet, let $S \subseteq \Sigma^*$, and let*

$$G = \{(u_r, v_r) : r = 1, \ldots, p\} \subseteq \Sigma^* \times \Sigma^*$$

be a finite set of guards with guard length at least $m \geq 1$. Let

$$T = T_{S,G} : \mathcal{L} \to \mathcal{L}$$

be the wrapping operator generated by the seed S and the guards G. Then

$$d(T(A), T(B)) \leq 2^{-m} d(A, B) \quad \textit{for all } A, B \in \mathcal{L}.$$

Proof. By Definitions 8 and 9, there exist $S \subseteq \Sigma^*$ and finitely many pairs (u_r, v_r), $r = 1, \ldots, p$, such that

$$T(L) = S \cup \bigcup_{r=1}^{p} u_r L v_r \quad \text{for all } L \in \mathcal{L},$$

and

$$|u_r| + |v_r| \geq m \quad \text{for all } r = 1, \ldots, p.$$

Let $A, B \in \mathcal{L}$ be arbitrary. For each $r = 1, \ldots, p$, Lemma 2 gives

$$d(u_r A v_r, u_r B v_r) \leq 2^{-(|u_r|+|v_r|)} d(A, B) \leq 2^{-m} d(A, B).$$

We now estimate the distance between the unions

$$U_A := \bigcup_{r=1}^{p} u_r A v_r, \qquad U_B := \bigcup_{r=1}^{p} u_r B v_r.$$

If $U_A = U_B$, then

$$d(U_A, U_B) = 0 \leq 2^{-m} d(A, B).$$

Assume that $U_A \neq U_B$, and let k be the first length at which they differ. Then there exists a word w_0 with $|w_0| = k$ such that

$$\mathbf{1}_{U_A}(w_0) \neq \mathbf{1}_{U_B}(w_0).$$

Without loss of generality, assume that

$$\mathbf{1}_{U_A}(w_0) = 1 \quad \text{and} \quad \mathbf{1}_{U_B}(w_0) = 0.$$

 doi:10.20944/preprints202604.0952.v1

Then there exists an index r_0 such that

$$\mathbf{1}_{u_{r_0}Av_{r_0}}(w_0) = 1,$$

while

$$\mathbf{1}_{u_rBv_r}(w_0) = 0 \quad \text{for all } r = 1, \dots, p.$$

In particular,

$$\mathbf{1}_{u_{r_0}Bv_{r_0}}(w_0) = 0.$$

Hence

$$\mathbf{1}_{u_{r_0}Av_{r_0}}(w_0) \neq \mathbf{1}_{u_{r_0}Bv_{r_0}}(w_0),$$

so the first disagreement between $u_{r_0}Av_{r_0}$ and $u_{r_0}Bv_{r_0}$ occurs at some length not exceeding k. Therefore

$$d(U_A, U_B) = 2^{-k} \leq d(u_{r_0}Av_{r_0}, u_{r_0}Bv_{r_0}).$$

Using the estimate above, we obtain

$$d(U_A, U_B) \leq 2^{-m} d(A, B).$$

Finally, by Lemma 3,

$$d(T(A), T(B)) = d(S \cup U_A, S \cup U_B) \leq d(U_A, U_B) \leq 2^{-m} d(A, B).$$

This completes the proof. □

Theorem 4. *Let Σ be a finite alphabet, let $S \subseteq \Sigma^*$, and let*

$$G = \{(u_r, v_r) : r = 1, \dots, p\} \subseteq \Sigma^* \times \Sigma^*$$

be a finite set of guards with guard length at least $m \geq 1$. Let

$$T = T_{S,G} : \mathcal{L} \to \mathcal{L}$$

be the wrapping operator generated by the seed S and the guards G. Then the equation

$$L = T(L)$$

admits a unique solution $L^* \in \mathcal{L}$. Moreover, for every initial language $L^{(0)} \in \mathcal{L}$, the sequence defined by

$$L^{(n+1)} = T\big(L^{(n)}\big), \qquad n \geq 0,$$

converges to L^*.

Proof. The space $(\mathcal{L}, d)$ is complete by the completeness theorem stated in the Preliminaries. By Lemma 4, the operator T is a contraction on $(\mathcal{L}, d)$. Therefore, the conclusion follows from Banach's contraction principle. □

Lemma 5. *Let Σ be a finite alphabet, let $S \subseteq \Sigma^*$, and let*

$$G = \{(u_r, v_r) : r = 1, \dots, p\} \subseteq \Sigma^* \times \Sigma^*$$

be a finite set of guards with guard length at least $m \geq 1$. Let

$$T = T_{S,G} : \mathcal{L} \to \mathcal{L}$$

 doi:10.20944/preprints202604.0952.v1

be the wrapping operator generated by the seed S and the guards G. Then for any initial approximation $L^{(0)}$ and for any $n \geq 1$,

$$d(L^{(n)}, L^*) \leq 2^{-mn} d(L^{(0)}, L^*).$$

Proof. We prove the statement by induction on n.

For $n = 1$, from Lemma 4 we have

$$d(L^{(1)}, L^*) = d(T(L^{(0)}), T(L^*)) \leq 2^{-m} d(L^{(0)}, L^*).$$

Assume the statement holds for n. Then

$$\begin{aligned} d(L^{(n+1)}, L^*) &= d(T(L^{(n)}), T(L^*)) \\ &\leq 2^{-m} d(L^{(n)}, L^*) \quad \text{(by contractivity)} \\ &\leq 2^{-m} \cdot 2^{-mn} d(L^{(0)}, L^*) \quad \text{(by the inductive hypothesis)} \\ &= 2^{-m(n+1)} d(L^{(0)}, L^*). \end{aligned}$$

This completes the proof. □

Corollary 1. *Under the conditions of Lemma 5, if we want $L^{(n)}$ and L^* to coincide on all words of length less than N, it is sufficient to require*

$$d(L^{(n)}, L^*) \leq 2^{-N}.$$

A sufficient condition for this is

$$n \geq \frac{N + \log_2 d(L^{(0)}, L^*)}{m},$$

provided that $d(L^{(0)}, L^*) > 0$.

In particular, if $L^{(0)} = \emptyset$, then $d(L^{(0)}, L^*) \leq 1$, hence

$$n \geq \frac{N}{m}$$

is a sufficient condition.

If $d(L^{(0)}, L^*) = 0$, then $L^{(0)} = L^*$ and the conclusion is trivial.

Proof. If $d(L^{(0)}, L^*) = 0$, then $L^{(0)} = L^*$, hence $L^{(n)} = L^*$ for all $n \geq 0$, and the conclusion is immediate.

Assume now that $d(L^{(0)}, L^*) > 0$. By Lemma 5,

$$d(L^{(n)}, L^*) \leq 2^{-mn} d(L^{(0)}, L^*).$$

Thus it is sufficient to require

$$2^{-mn} d(L^{(0)}, L^*) \leq 2^{-N}.$$

Taking base-2 logarithms, we obtain

$$-mn + \log_2 d(L^{(0)}, L^*) \leq -N,$$

which is equivalent to

$$-mn \leq -N - \log_2 d(L^{(0)}, L^*).$$

Multiplying by -1, we get

$$mn \geq N + \log_2 d(L^{(0)}, L^*).$$

 doi:10.20944/preprints202604.0952.v1

Therefore,

$$n \geq \frac{N + \log_2 d(L^{(0)}, L^*)}{m}$$

is a sufficient condition.

If $L^{(0)} = \emptyset$, then $d(L^{(0)}, L^*) \leq 1$, so

$$\log_2 d(L^{(0)}, L^*) \leq 0.$$

Hence

$$N + \log_2 d(L^{(0)}, L^*) \leq N,$$

and therefore

$$n \geq \frac{N}{m}$$

is also a sufficient condition. □

Theorem 5. *Let Σ be a finite alphabet, let $S \subseteq \Sigma^*$, and let*

$$G = \{(u_r, v_r) : r = 1, \ldots, p\} \subseteq \Sigma^* \times \Sigma^*$$

be a finite set of guards with guard length at least $m \geq 1$. Let

$$T = T_{S,G} : \mathcal{L} \to \mathcal{L}$$

be the wrapping operator generated by the seed S and the guards G. Let L^* be the fixed point of T. If

$$d(L, L^*) \leq 2^{-N},$$

then L and L^* coincide on all words of length strictly less than N.

Proof. If $d(L, L^*) = 0$, then $L = L^*$, and the conclusion is immediate.

Assume that $d(L, L^*) > 0$. By Definition 3, there exists an integer $k \geq 0$ such that

$$d(L, L^*) = 2^{-k},$$

where k is the smallest length at which L and L^* differ. Since

$$d(L, L^*) \leq 2^{-N},$$

it follows that

$$2^{-k} \leq 2^{-N},$$

hence

$$k \geq N.$$

Therefore the first disagreement between L and L^* can occur only at a word of length at least N. Equivalently, L and L^* coincide on all words of length strictly less than N. □

4. Examples

Example 1. *Let $\Sigma = \{a, b\}$ and consider the operator $T : \mathcal{L} \to \mathcal{L}$ given by*

$$T(L) = \{\varepsilon\} \cup aLb.$$

Here $S = \{\varepsilon\}$, $p = 1$, $u_1 = a$, $v_1 = b$, and $|u_1| + |v_1| = 2$, hence T is guarded with guard length $m = 2$.

 doi:10.20944/preprints202604.0952.v1

Starting with $L^{(0)} = \emptyset$, the first iterates are

$$\begin{aligned} L^{(1)} &= T(\emptyset) = \{\varepsilon\}, \\ L^{(2)} &= T(\{\varepsilon\}) = \{\varepsilon, ab\}, \\ L^{(3)} &= T(\{\varepsilon, ab\}) = \{\varepsilon, ab, aabb\}, \\ L^{(4)} &= T(\{\varepsilon, ab, aabb\}) = \{\varepsilon, ab, aabb, aaabbb\}. \end{aligned}$$

By induction one easily verifies that

$$L^{(n)} = \{a^k b^k : \ 0 \le k < n\} \qquad (n \ge 1).$$

Consequently, the limit (and hence the unique fixed point of T) is

$$L^* = \bigcup_{n \ge 1} L^{(n)} = \{a^k b^k : \ k \ge 0\}.$$

Moreover, since $m = 2$, Lemma 4 gives contractivity with constant 2^{-2}, and Theorem 4 yields convergence of the Picard iteration to the unique fixed point.

Example 2. *Let $\Sigma = \{0, 1\}$ and consider two operators $T_1, T_2 : \mathcal{L} \to \mathcal{L}$ defined by*

$$\begin{aligned} T_1(L) &= \{\varepsilon\} \cup 0L1 \cup 00L11, \\ T_2(L) &= \{\varepsilon\} \cup 00L11 \cup 000L111. \end{aligned}$$

Both operators are guarded, but with different guard lengths:

$$m_1 = 2 \quad \text{for } T_1 \qquad \text{and} \qquad m_2 = 4 \quad \text{for } T_2.$$

Starting with $L^{(0)} = \emptyset$, the first iterates for T_1 are

$$\begin{aligned} L_1^{(1)} &= \{\varepsilon\}, \\ L_1^{(2)} &= \{\varepsilon,\ 01,\ 0011\}, \\ L_1^{(3)} &= \{\varepsilon,\ 01,\ 0011,\ 000111,\ 00001111\}. \end{aligned}$$

For T_2 we obtain

$$\begin{aligned} L_2^{(1)} &= \{\varepsilon\}, \\ L_2^{(2)} &= \{\varepsilon,\ 0011,\ 000111\}, \\ L_2^{(3)} &= \{\varepsilon,\ 0011,\ 000111,\ 00001111,\ 0000011111,\ 000000111111\}. \end{aligned}$$

The fixed points can be described explicitly. For T_1 one checks that

$$L_1^* = \{0^n 1^n : n \ge 0\}.$$

Indeed, T_1 allows wrapping by $(0, 1)$ and by $(00, 11)$, hence every application increases the exponent by 1 or by 2, and all $n \ge 0$ are reachable.

For T_2 the allowed wrappings are $(00, 11)$ and $(000, 111)$, so the exponent increases by 2 or 3. Since the semigroup generated by 2 and 3 contains all integers $n \ge 2$, we obtain

$$L_2^* = \{\varepsilon\} \cup \{0^n 1^n : n \ge 2\}.$$

 doi:10.20944/preprints202604.0952.v1

In particular, $L_2^* \subset L_1^*$.

Finally, the guard length controls the certified correctness depth: the contraction constants are

$$c_1 = 2^{-m_1} = \frac{1}{4}, \qquad c_2 = 2^{-m_2} = \frac{1}{16},$$

so the error $d(L_i^{(n)}, L_i^*)$ *decays faster for* T_2 *than for* T_1.

Example 3. *Let* $\Sigma = \{x, y\}$ *and define* $T : \mathcal{L} \to \mathcal{L}$ *by*

$$T(L) = xLx \cup yLy.$$

This operator is guarded with seed $S = \varnothing$ *and guard length* $m = 2$, *since each wrapping adds one symbol on the left and one symbol on the right. Hence, by Lemma 4,* T *is a contraction on* $(\mathcal{L}, d)$ *with constant* 2^{-2}, *and therefore it has a unique fixed point.*

Starting from $L^{(0)} = \{\varepsilon\}$ *we obtain*

$$L^{(1)} = \{xx, yy\}, \qquad L^{(2)} = \{xxxx, xyyx, yxxy, yyyy\},$$

and, in general, $L^{(n)}$ *consists of palindromes of length exactly* $2n$.

Although the set-theoretic union $\bigcup_{n\geq 0} L^{(n)}$ *is the set of all even-length palindromes, the Picard iteration converges in the ultrametric to the empty language. Indeed, the shortest words in* $L^{(n)}$ *have length* $2n$, *so the first disagreement between* $L^{(n)}$ *and* $\varnothing$ *occurs at length* $2n$, *and therefore*

$$d\big(L^{(n)}, \varnothing\big) = 2^{-2n} \xrightarrow[n\to\infty]{} 0.$$

Since $T(\varnothing) = \varnothing$, *the limit* $\varnothing$ *is a fixed point; by Banach's theorem it is the unique fixed point of* T.

This example shows that convergence in the length-based ultrametric means stabilization of membership on all words up to any fixed length, and does not coincide with convergence in terms of set-theoretic inclusion.

5. Applications in Computer Science

The results of the previous sections have natural applications in computer science, especially in recursively defined languages, data structures, and validation schemes. The basic idea is that recursion is well controlled whenever each recursive occurrence is separated from the outer context by a nonzero guard. In this case, the corresponding wrapping operator is a contraction in the metric space under consideration; that is, it reduces the distance between languages by a factor smaller than 1. From a computer-science perspective, this means that the recursive specification can be viewed not only as a formal description, but also as a computational process in which, for inputs of bounded length, there exists a provable bound beyond which the result is guaranteed to be correct.

5.1. Recursive Data Structures and Formats

The first direct application of the theory under consideration is to recursive data structures and formats, in which complex objects are constructed from simpler ones by adding a new outer structural layer. Such situations arise in XML (Extensible Markup Language) and HTML (Hypertext Markup Language) documents with nested tags [6], in Lisp-like expressions, in serialized trees, and in JSON (JavaScript Object Notation) objects with recursively nested substructures [7]. In this setting, the set of valid objects can be described by an operator that adds new correct wrappers around already valid substructures to the basic admissible cases. The fixed point of this operator then represents the language of all valid objects that can be generated through a finite number of construction steps. From a computer science perspective, the sequence of iterations $L^{(0)}, L^{(1)}, L^{(2)}, \ldots$ can be viewed as a series of progressively refined approximations of the valid format. The initial iterations cover only the simplest structures, while subsequent ones allow increasingly deeper nesting. This corresponds to the

 doi:10.20944/preprints202604.0952.v1

way parsers and validators for recursive formats operate in practice: they process the structure layer by layer rather than as a single indivisible object. Therefore, fixed-point theory provides both a formal description and a clear model for stepwise computation and validity checking.

Table 1. Examples of recursive data structures and formats with their computer science interpretation.

Example	Sample Recursive Construction	Computer Science Interpretation
HTML/XML documents	`<tag>doc</tag>`	Each new layer of tags adds an outer syntactic context; $L^{(n)}$ describes documents with nesting depth up to n.
S-expressions	`(f expr)`	An appropriate model for parsing nested Lisp-like terms, where each new expression contains an already valid subexpression.
JSON tree structure	`{"value": x, "children": [node]}`	Describes recursively nested objects; $L^{(n)}$ corresponds to trees of depth up to n.
Linked list	`{"head": x, "tail": list}`	Each new object adds one more node to the list; the iterations describe lists of length up to n.
Serialized abstract syntax tree	`Add(expr,num)`, `Neg(expr)`	Each constructor builds a more complex expression from an already valid subexpression; $L^{(n)}$ describes trees with bounded syntactic depth.
Constructions in a Domain-Specific Language (DSL)	`Wrap(cmd)`	Each operator adds a new syntactic layer around an already valid command or expression.

5.2. Recursive Data Types, Abstract Syntax Trees, and Serialization

A particularly important application of the theory under consideration arises in recursive data types and abstract syntax trees (ASTs). In compilers, interpreters, and static analysis systems, more complex constructions are built from simpler ones through constructors that add a new outer layer around an already valid subexpression [8]. Thus, a valid structure can be viewed as the result of successive construction rather than merely as a formally specified object.

For example, in a language of arithmetic expressions with the constructors `Num`, `Neg(expr)`, and `Add(expr, Num)`, the corresponding operator may be written schematically as

$$T(L) = \{\,\texttt{Num}\,\} \cup \texttt{Neg}(L) \cup \texttt{Add}(L, \texttt{Num}),$$

which describes how new valid expressions are obtained from already valid ones.

In this case, the fixed point L^* is the set of all well-formed serialized expressions, while $L^{(n)}$ contains the expressions of syntactic depth at most n. The same logic applies to classical data structures. A binary tree can be represented by `Leaf` and `Node(left, right)` [12], and a singly linked list by `Cons(head, tail)` [8]. In serialization, for example in JSON [7], this leads to forms such as `{"head": value, "tail": List}` or `{"value": x, "children": [Tree, ...]}`. In both cases, the recursive part appears inside a clearly defined outer context. This wrapper plays the role of a guard and makes the step-by-step construction and verification of valid structures possible.

5.3. Validation under Limited Resources: The Length of the Guard as a Computational Parameter

One of the most practically relevant consequences of the theory is its application to validation under limited resources. In real systems, it is usually not necessary to determine validity for all possible unfoldings of recursion, but only for inputs up to a certain size—for example, a JSON document of fixed length [7], a syntax tree up to a given depth, or a query of limited volume. Therefore, the

 doi:10.20944/preprints202604.0952.v1

quantitative result of the paper has direct algorithmic meaning: if the minimum length of the outer wrapper is m, then after sufficiently many iterations the approximation $L^{(n)}$ already coincides with the fixed point on all inputs of length below N.

In the paper itself, this is formulated as an explicit bound of the form $n \geq N/m$ for correctness on words below a given length. From a computer science perspective, this means that the convergence theorem can be used as a formal stopping criterion. Instead of introducing an arbitrary recursion bound, the number of required steps can be derived from the structure of the specification itself. For a validator of nested JSON objects [11,14], a parser for Lisp-like expressions, or verification of serialized trees, this means that the necessary processing depth is not chosen heuristically, but is determined by the parameter m. Thus, the length of the guard becomes a measure not only of the structure of recursion, but also of the cost of its reliable computation.

Among the most important computer science interpretations of the theory under consideration is that the length of the guard can be viewed as a computational parameter. Mathematically, m is the minimum length of the mandatory outer wrapper added at each recursive step. From a computer science perspective, this is the amount of syntactic context that separates one recursive level from the next. The greater this value, the more quickly the iterations stabilize on inputs of limited length, since the contraction coefficient is 2^{-m}. This also gives the result a clear practical meaning. A more explicit outer structure not only facilitates reading of the data, but also makes formal validation more predictable. For example, a JSON object with fields `"head"` and `"tail"` contains more mandatory structural context than an array in which the recursive element is placed directly. For this reason, a validator for a list may reach a correct result in fewer steps than a validator for a more general recursive array structure on the same input.

5.4. Restricted Fragment of Recursive JSON Schemas

The most direct practical application of the theory under consideration is to recursive JSON schemas [9–11]. Here, however, it must be clearly emphasised that the approach does not cover JSON Schema in its full generality, but rather a restricted recursive fragment in which the recursive occurrence is located within a clearly defined outer context—for example, inside an array, in an object with fixed keys [7], or within a nested property. In such cases, the language of valid JSON documents can be regarded as the result of step-by-step construction: one starts from the simplest admissible values and then gradually adds increasingly deeply nested structures. It is precisely this class of schemas that corresponds to the model of wrapping operators introduced in this paper.

A particularly noteworthy example is the model of nested comments in a social network. A comment may contain the fields `"author"`, `"text"`, and `"replies"`, where `"replies"` is an array of new comments with the same structure. The initial approximation then describes comments without replies, the next one describes comments with one level of nested replies, and each subsequent step permits one more level of nesting. In the same way, a JSON representation of a linked list or a tree structure can be viewed as a serialization of classical recursive data types [8,12]. In this sense, the theory provides a clear model for the step-by-step construction and validation of recursive JSON structures when the recursion is placed within a sufficiently explicit structural wrapper.

5.5. Scope and Limitations

The proposed model is strongest for positive, structurally guarded recursive specifications that can be reduced to a finite wrapping operator. This includes formats with nested tags [6], abstract syntax trees [8], linked lists, tree-shaped JSON structures, and a limited but practically relevant fragment of recursive JSON schemas [9,11]. Under these conditions, the results of the previous sections provide, at once, unambiguous semantics, iterative convergence, and a formal validation bound for inputs of limited length.

At the same time, the model should not be regarded as a complete semantics for all possible schema languages with richer compositional operators [6,10]. The most accurate conclusion is that the theory under consideration provides a rigorous foundation for guarded recursive validation and

 doi:10.20944/preprints202604.0952.v1

the step-by-step unfolding of recursive structures, but it does not offer a definitive solution for all constructive possibilities of industrial schema standards [10,11].

Author Contributions: The mentioned authors participated equally to the study and are arranged in alphabetical order as follows: conceptualization, H.H., A.I., H.K., and B.Z; methodology, H.H., A.I., H.K., and B.Z; investigation, H.H., A.I., H.K., and B.Z; writing—original draft preparation, H.H., A.I., H.K., and B.Z.; writing—review and editing, H.H., A.I., H.K., and B.Z. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: The authors would like to express their gratitude for the support provided by the European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project DUECOS BG-RRP-2.004-0001-C01.

Funding: This research received no external funding.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Banach, S. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundam. Math.* **1922**, *3*, 133–181.
2. de Bakker, J.W.; Zucker, J.I. Processes and the denotational semantics of concurrency. *Inf. Control* **1982**, *54*, 70–120. https://doi.org/10.1016/S0019-9958(82)91250-5
3. America, P.; Rutten, J.J.M.M. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.* **1989**, *39*, 343–375. https://doi.org/10.1016/0022-0000(89)90027-5
4. Salomaa, A. *Formal Languages*; Academic Press: New York, NY, USA, 1973; ISBN 978-0-12-615750-5.
5. Eilenberg, S. *Automata, Languages, and Machines, Vol. A*; Academic Press: New York, NY, USA, 1974; ISBN 978-0-12-234001-7.
6. Martens, W.; Neven, F.; Schwentick, T.; Bex, G.J. Expressiveness and complexity of XML Schema. *ACM Trans. Database Syst.* **2006**, *31*, 770–813. https://doi.org/10.1145/1166074.1166076
7. Bray, T. (Ed.) The JavaScript Object Notation (JSON) Data Interchange Format. *RFC 8259*; IETF: Fremont, CA, USA, 2017. https://doi.org/10.17487/RFC8259
8. Bird, R. *Introduction to Functional Programming using Haskell*, 2nd ed.; Prentice Hall Europe: Hemel Hempstead, UK, 1998; ISBN 0-13-484346-0.
9. Pezoa, F.; Reutter, J.L.; Suarez, F.; Ugarte, M.; Vrgoč, D. Foundations of JSON Schema. In *Proceedings of the 25th International Conference on World Wide Web (WWW 2016)*; ACM: New York, NY, USA, 2016; pp. 263–273. https://doi.org/10.1145/2872427.2883029
10. Attouche, L.; Baazizi, M.-A.; Colazzo, D.; Ghelli, G.; Sartiani, C.; Scherzinger, S. Validation of Modern JSON Schema: Formalization and Complexity. *Proc. ACM Program. Lang.* **2024**, *8* (POPL), Article 55. https://doi.org/10.1145/3632891
11. Wright, A.; Andrews, H.; Hutton, B.; Dennis, G. JSON Schema: A Media Type for Describing JSON Documents. *IETF Internet-Draft* draft-bhutton-json-schema-01; 2020. https://json-schema.org/draft/2020-12/json-schema-core.html
12. Meertens, L. First Steps Towards the Theory of Rose Trees. Working Paper 592 ROM-25, IFIP Working Group 2.1; CWI: Amsterdam, The Netherlands, 1988.
13. Gajić, L. On ultrametric spaces. *Novi Sad J. Math.* **2001**, *31*, 69–71.
14. Ajv JSON Schema Validator, version 8.x; 2023. Available online: https://ajv.js.org (accessed on 1 January 2024).