Scientific Inquiry and Review (SIR)
Volume 10 Issue 1, 2026
ISSN$_{(P)}$: 2521-2427, ISSN$_{(E)}$: 2521-2435
Homepage: https://journals.umt.edu.pk/index.php/SIR

Applications and Computational Varieties of Free Soft Trioids

Author (s): Gulay Oguz[1] and David A. Oluyori[2]

Affiliation (s): [1]Harran University, Sanlurfa, Turkey
[2]Federal University of Technology, Babura, Nigeria

Academic Editor Abaid ur Rehman Virk

DOI: https://doi.org/10.32350/sir.101.03

History: Received: December 23, 2025, Revised: January 05, 2026, Accepted: February 11, 2026, Published: March 17, 2026

Citation: Oguz G, Oluyori DA. Applications and computational varieties of free soft trioids. *Sci Inq Rev.* 2026;10(1):61–101. https://doi.org/10.32350/sir.101.03

Conflict of Interest: Author(s) declared no conflict of interest

A publication of
The School of Science
University of Management and Technology, Lahore, Pakistan

Applications and Computational Varieties of Free Soft Trioids

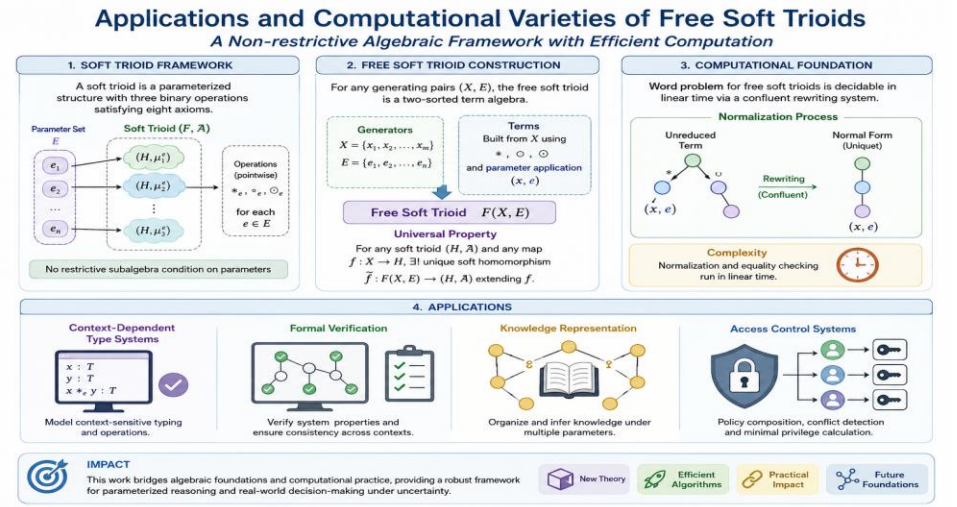
Gulay Oguz[1*] and David Adeyemi Oluyori[2]

[1]Department of Mathematics, Faculty of Arts and Sciences, Harran University, Turkey
[2]Department of Applied Mathematics, School of Science and ICT, Federal University of Technology, Nigeria

ABSTRACT

Integrating soft set theory with multi-operational structures poses a foundational challenge In parameterized universal algebra,. In this paper, the theory of free soft trioids were developed establishing the existence and construction of free objects in the category SoftTrioid**.** Also we construct a non-restrictive definition of soft trioids equipped with pointwise operations with genuine context sensitivity. The free soft trioid being realized as a two-sorted term algebra prove the universal property and show that the word problem is decidable in linear time via a confluent rewriting system as the algorithmic implementations were supported by empirical validation. The relevance of our study were demonstrated by its applications in context-dependent type systems, formal verification, knowledge representation and access control thus situating soft trioid theory as a robust link between algebraic foundations and computational practice.

GRAPHİCAL ABSTRACT



*Corresponding Author: gulay.oguz@harran.edu.tr

Keywords: computational applications, free soft trioid, Soft set, soft trioid

Highlights

- A non-restrictive framework for free soft trioids is developed.
- Linear-time normalization is established through confluent rewriting.
- Applications include type systems, verification, knowledge representation, and access control.

1. INTRODUCTİON

The intermix framework of algebraic structure and contextual information are quite challenging in the field of mathematics and computer science as existing frameworks have prove useful in modelling deterministic relationships under fixed operations but fail in operations which involves external parameters, environmental conditions or subjective interpretations as depicted in multi-criteria decision making [1], adaptive access control and heterogeneous knowledge representation. To address this bottleneck, two different lines of enquiry has emerged but the potential of their intersection have been unexplored. Firstly, the Molodtsov's soft set theory [5] which by design handles uncertainty via parameter-indexed families of subsets, preserving classical set-theoretic precision unlike fuzzy [6] or rough sets [7] and is widely used in algebra [8], topology [9] and decision theory [10]. The potency of soft set theory, lies in permitting arbitrary parameters—criteria, contexts, agents or time instances — without prior restriction. Secondly is the associative algebra which emerged from semigroups via Dimonoids [12, 13] to Trioids [14, 15]. Historically, Trioid $(T, \dashv, \vdash, \perp)$ which originates from the study of operads and polytopes present three binary operations with eight axioms that encode relevant interactions to algebraic combinatorics, rewriting and Leibniz algebras [20]. Further investigation into structural themes such as semiretractions [16], commutative variants [18] and combinatorial links [19] have enriched the theory thereby solidifying trioids as a fruitful algebraic domain.

Recent applications of soft set theory with other structures such as dimonoids, digroups [23, 24] and dirings [25] have proven useful in bridging the ones separated domain of studies, yet the results realized impose a restrictive subalgebra condition on parameters—an inheritance that undermines expressivity, as the dynamic nature of real-world contexts cannot be fully captured using closed substructures, thus leaving open the

construction of free objects in soft trioids. This paper is aimed at resolving his gap through a non-restrictive definition that allows arbitrary parameter images and pointwise operations, enabling genuine context sensitivity. We develop free soft trioids as two-sorted term algebras over generators X and P, with parameter application as a unary operation thus establishing a universal property which does not exist in literature and supply empirically validated normalization algorithms. The main contribution is 5 fold as follows:

First, we prove the existence and construction of free soft trioids $(\mathcal{F}(X,P),\ \Omega_F,\ P)$ for any generating pairs (X,P) thereby establishing the universal property in the category SoftTrioid of soft trioids and soft homomorphisms. *Secondly*, we characterize the word problem for free soft trioids, validating decidability in linear time through confluence rewriting system which contrast the universal word problem for the variety of all soft trioids. *Thirdly*, we develop efficient algorithmic implementations such as pseudocode for normalization, equality checking and the empirical validation using Haskell prototype. *Fourthly*, we demonstrate the practical utility of free soft trioids via detailed case studies in context-dependent access control systems, where policy consistency, conflict detection and minimal privilege calculation are reduced to algebraic operations. *Finally*, we establish the enriched categorical structure of SoftTrioid in connection with coalgebraic methods, thus laying the foundation for future directions in parametrized universal algebra.

The advantages of this novel method over existing frameworks are as follows: Unlike the existing soft algebraic structures that restrict parameters to subalgebras, this framework permits arbitrary context-induced subsets activates genuine parameter sensitivity. Also the algebraic approach adopted in this study provides symbolic reasoning to formal verification, policy analysis and knowledge representation unlike existing numerical or stochastic modelling methods [31–34] based on quantitative simulation. Therefore, the integration of soft theory with trioid structures serve as a natural setting for modelling three modes of composition namely: sequential, parallel and override as shown in access control, workflow management and multi-modal logic systems. This paper is organized as follows. Section 2 recalls preliminaries on soft sets and trioids. In section 3 we construct free soft trioids - definition, term algebra construction, universal property and word problem. Section 4 discusses implementations,

algorithms, validation and applications in type systems, verification, knowledge representation and access control. Section 5 concludes with contributions and future directions such as fractional and stochastic extensions.

2. PRELİMİNARİES

This section establishes the foundational concepts necessary for the development of free soft trioids. We refer the reader to Molodtsov [5] for more detail on Soft set and Zhuchok [12, 14, 16 - 20]

2.1. Soft Sets

Since its introduction by Molodtsov [5], the study of soft set has shifted the paradigm in handling uncertainty and parameterization in numerous mathematical structures unlike the traditional methods which modifies set membership via degree functions (fuzzy sets) or approximation operators (rough sets [7]). This conceptual transparency generated numerous applications across various mathematical disciplines, such as algebra [8] topology [9], and decision theory [1, 10].

Definition 1. Let C be a universal set and E a nonempty set of parameters. A *soft set* over C is a pair (Ω, K) where $K \subseteq E$ and

$$\Omega : \quad K \quad \rightarrow \quad P(C)$$

is a mapping from the parameter set K to the power set of C.

Definition 2. The *support* of a soft set (Ω, K) is defined as

$$Supp(\Omega, K) \quad = \quad \{\alpha \in K : \Omega(\alpha) \neq \emptyset\}.$$

A soft set is called *non-null* if its support is non-empty, i.e., $Supp(\Omega, K) \quad \neq \emptyset$.

Remark 1. The parameter set K need not be finite, though in computational applications we typically restrict to finite or countably infinite parameter sets. The flexibility in choosing K allows soft sets to model a wide range of contextual information, including criteria, agents, time instances, or environmental conditions.

Example 1. Let C = $\{h_1, h_2, h_3, h_4\}$ be a set of houses, and let

E = {expensive, beautiful, cheap, modern} be a set of parameters. Define K = {expensive, beautiful} $\subseteq E$ and

$\Omega(\text{expensive}) = \{h_1, h_3\}$, $\Omega(\text{beautiful}) = \{h_2, h_3, h_4\}$.

Then (Ω, K) is a soft set over C representing the houses that satisfy each property. The support is Supp $(\Omega, K) = K$, for both non-empty parameter images.

2.2. Trioids

Trioids represent originates from the study Loday on operadic chain modules and Stasheff polytopes [12, 14]. Over the years, various the trioid theory has been further deepened through numerous considerations, notably on semiretractions [16, 17], commutative variants [18] and combinatorial connections [19, 20], establishing trioids as a rich terrain for algebraic exploration.

Definition 3. *[13] A trioid is a quadruple* $(T, \dashv, \vdash, \perp)$ *where* T *is a nonempty set equipped with three binary operations:* $\dashv, \vdash, \perp : T \times T \longrightarrow T$ *satisfying the following axioms for all* $x, y, z \in T$*:*

$(\mathrm{T1})(x \dashv y) \dashv z = x \dashv (y \vdash z)$

$(\mathrm{T2})(x \vdash y) \dashv z = x \vdash (y \dashv z)$

$(\mathrm{T3})(x \dashv y) \vdash z = x \vdash (y \vdash z)$

$(\mathrm{T4})(x \dashv y) \dashv z = x \dashv (y \perp z)$

$(\mathrm{T5})(x \perp y) \dashv z = x \perp (y \dashv z)$

$(\mathrm{T6})(x \dashv y) \perp z = x \perp (y \vdash z)$

$(\mathrm{T7})(x \vdash y) \perp z = x \vdash (y \perp z)$

$(\mathrm{T8})(x \perp y) \vdash z = x \vdash (y \vdash z)$

Remark 2. The axioms (T1)–(T8) encapsulates interaction patterns within the three operations. Axiom (T1) describe $\dashv$ as associative. Axioms (T2) and (T3) confirms that mutual interaction between $\dashv$ and $\vdash$. Axioms (T4)–(T8) validates the relationships involving $\perp$. Notably, none of the operations is required to be commutative and $\vdash$ and $\perp$ need not be associative independently. This non-associative structure is essential for modeling sequential and parallel composition in applications.

Example 2. Let $T = \mathbb{R}$ and define operations by:

$x \dashv y = x - y, \qquad x \vdash y = x + y, \qquad x \perp y = xy.$

One verifies that $(T, \dashv, \vdash, \perp)$ satisfies the trioid axioms. For instance, (T1) holds since sub-traction is not associative in the usual sense, but the axiom imposes a specific associativity condition; (T2) holds since $(x + y) — z = x + (y — z)$; and (T4) holds since $(x — y) — z = x — (y \cdot z)$ only if the operations are defined appropriately. More sophisticated examples arise from formal language theory and rewriting systems [15].

Definition 4. Let $(T_1, \dashv_1, \vdash_1, \perp_1)$ and $(T_2, \dashv_2, \vdash_2, \perp_2)$ be trioids. A function $\psi : T_1 \to T_2$ is a *trioid homomorphism* if for all $x, y \in T_1$ and for each operation $\circ \in \{\dashv, \vdash, \perp\}$,

$\psi(x \circ_1 y) = \psi(x) \circ_2 \psi(y)$.

Remark 3. The free trioid $Frt(X)$ on a set X has an explicit construction as equivalence classes of words over the extended alphabet $\hat{X} = X \cup \bar{X}$, where $\bar{X} = \{\bar{x} : x \in X\}$ is a disjoint copy of X [15]. The operations on representatives are given by:

$w \dashv u = w\bar{u}$, $w \vdash u = \bar{w}u$, $w \perp u = wu$,

where $\bar{w}$ denotes the word obtained from w by replacing each symbol x with $\bar{x}$ and vice-versa. This construction will be essential for the free soft trioid developed in Section 3.

2.3. Categorical Preliminaries

We briefly recall the categorical notions that will be employed throughout this paper. For a comprehensive treatment, we refer to [2, 11].

Definition 5. Let $\mathcal{C}$ be a category. The *product* of two objects $A, B \in \mathcal{C}$ is an object $A \times B$

together with projection morphisms

$\pi_1 : A \times B \to A, \qquad \pi_2 : A \times B \to B$

such that for any object C and morphisms $f : C \to A$, $g : C \to B$, there exists a unique morphism $\langle f, g \rangle : C \to A \times B$ satisfying $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$.

Definition 6. An *adjunction* between categories $\mathcal{C}$ and $\mathcal{D}$ consists of functors

$F: \mathcal{C} \to \mathcal{D}, \qquad G: \mathcal{D} \to \mathcal{C}$

and a natural isomorphism

$$Hom_D(F(C), D) \cong Hom_C(G(C), D)$$

for all objects $C \in \mathcal{C}$, $D \in \mathcal{D}$. The functor F is called the *left adjoint* and G the *right adjoint*.

Definition 7. Let SoftTrioid be the category of soft trioids and soft homomorphisms (defined in Section 3). The *forgetful functor*

U : SoftTrioid → Set × Set

is defined on objects by $U(\mathrm{f}, \Omega, K) = (K, T)$ and on morphisms by $U(\varphi, \psi) = (\varphi, \psi)$. This functor forgets the algebraic operations and the soft set structure, retaining only the parameter set and the underlying set of the base trioid.

Remark 4. The category Set × Set posseses objects pairs of sets and as morphisms pairs of functions which creates the natural setting for the two-sorted nature of soft trioids namely for parameters and algebraic elements.

2.4. Comparison with Related Frameworks

Our proposed framework for soft trioids is particularly novel within the parameterised algebraic structures as existing research on soft algebras [8, 9] defines a soft algebra as a soft set whose parameter images are subalgebras of the underlying structure, a restriction that inhibits its expressive power—in access control. For instance, the permissions given to a user role is typically an arbitrary subset, not a subalgebra closed under composition—whereas by Definition 8 this restriction is removed, thus allowing arbitrary parameter images with pointwise operations, a necessary generalization in the free construction in Section 3.3. As Trioids generalizes dimonoids [12, 13] by adding a third binary operation with interaction axioms; the soft trioid framework considered in this paper extends to the parameterised setting, subsuming soft dimonoids [23] when $\vdash$ and $\perp$ coincide and soft dirings [25] under ring-like distributivity conditions. The categorical aspect developed in Section 3.6 connects the study of soft trioids to an enriched category theory [3] and coalgebraic methods [28], distinguishing our work from purely algebraic approach and enhance it for modelling stateful and dynamic parameterised systems. While free constructions are clear for trioids [15] and soft sets [10] respectively, integrating it present difficulty owing to the two-dimensional nature of generation—algebraic generators produce elements through trioid operations.

Table 1. Comparison of Algebraic Frameworks for Parameterised Reasoning

Framework	Param.	#Ops	Free	Norm.
Soft sets [5]	+	0	×	—
Soft algebras [8]	+*	1	×	—
Trioids [15]	×	3	+	Linear
Soft dimonoids [23]	+*	2	×	—
Soft dirings [25]	+*	2	×	—
Free soft trioids (this work)	+	3	+	Linear

*Restricted to subalgebras; + = full support; × = no support.

Table 1 describes the differences between existing results in literature and our framework. Free soft trioid is the only framework that supports arbitrary parameterization, three distinct operations, free construction and linear-time normalization.

3. FREE SOFT TRİOİDS

This section establishes the mathematical foundations of free soft trioids. We introduce a non-restrictive definition of soft trioids that permits arbitrary parameter-indexed subsets, develop the categorical framework, construct free objects as two-sorted term algebras, show that the universal property and analyze the word problem.

3.1. Soft Trioids

We construct a generalized definition which permits arbitrary subsets while defining algebraic operations pointwise.

Definition 8. *Let* $\mathcal{T} = (T, \dashv, \vdash, \perp)$ *be a trioid and* K *a set of parameters. A soft trioid over* $\mathcal{T}$ *is a triple* $(\mathcal{T}, \Omega, K)$ *where* (Ω, K) *is a non-null soft set over the universe* T*, i.e.,* $(\Omega: K \ \mathcal{P}(T))$ *with* $Supp(\Omega, K) \neq \emptyset$*. No further restrictions are imposed on the subsets* $\Omega(\alpha)$ *for* $\alpha \in K$*.* The operations of the underlying trioid extend to the soft structure in the natural pointwise manner.

Definition 9. *Let* $(\mathcal{T}, \Omega, K)$ *be a soft trioid. For each parameter* $\alpha \in K$*, define the pointwise operations on* $\Omega(\alpha) \subseteq T$ *by*

$$\Omega(\alpha) \dashv \Omega(\alpha) = \{x \dashv y : x, y \in \Omega(\alpha)\},$$

and analogously for $\vdash$ *and* $\perp$. *The parameterized family*

$\{\Omega(\alpha):\ \alpha \in K\}$

İs thus equipped with three binary operations inherited pointwise from $\mathcal{T}$.

Remark 5. Definition 8 departs substantively from earlier formulations of soft algebras [8, 23], which required **Ω**(α) to be a subalgebra of $\mathcal{T}$ for every $\alpha \in K$. Our generalization is essential for the free construction developed in Section 3.3: the free soft trioid must permit arbitrary parameter-algebra relationships, imposing no a priori closure conditions.

Definition 10. Let $(\mathcal{T}_1, \Omega_1, K_1)$ *and* $(\mathcal{T}_2, \Omega_2, K_2)$ *be soft trioids. A soft homomorphism is a pair* (ϕ, ψ) *where:*

(1) $\phi: K_1 \to K_2$ is a function between parameter sets,

(2) $\psi: T_1 \to T_2$ is a trioid homomorphism,

(3) For every $\alpha \in K_1$, we have $\psi(\Omega_1(\alpha)) \subseteq \Omega_2(\phi(\alpha))$.

The category of soft trioids and soft homomorphisms is denoted SoftTrioid.

Definition 11. *The forgetful functor* $U: SoftTrioid \to Set \times Set$ *is defined by on* objects by $U(\mathcal{T}, \Omega, K) = (K, T)$ and on morphisms: $U(\phi, \psi) = (\phi, \psi)$. This functor forgets the algebraic operations and the soft set structure, retaining only the parameter set and the underlying set of the base trioid.

3.2. Products and Limits

The category SoftTrioid admits finite products, which will be useful in sequel.

Definition 12. *Let* $(\mathcal{T}_1, \Omega_1, K_1)$ *and* $(\mathcal{T}_2, \Omega_2, K_2)$ *be soft trioids. Their product is defined as:*

$(\mathcal{T}_1 \times \mathcal{T}_2, \Omega, K_1 \times K_2)$

where $\mathcal{T}_1 \times \mathcal{T}_2$ is the direct product trioid with operations defined componentwise and for all $(\alpha, \beta) \in K_1 \times K_2$,

$\Omega(\alpha, \beta) = \Omega_1(\alpha) \times \Omega_2(\beta)$.

Proposition 1. The product defined above is the categorical product in SoftTrioid, with projection homomorphisms (π_{K_i}, π_{T_i}) for $i = 1, 2$.

Proof: The verification is routine: the componentwise operations ensure that the projections are trioid homomorphisms and the soft set condition follows immediately from the definition of Ω. The universal property of the product in $Set \times Set$ *lifts to* SoftTrioid by the pointwise construction of the mediating morphism.

3.3. Free Soft trioids: The Two Sorted Construction

The The construction of free objects in SoftTrioid requires simultaneous generation in twodimensions: algebraicgenerationthrough the trioid operations $\dashv, \vdash, \perp$ and parametric generation through the soft set structure. This duality necessitates a two-sorted approach.

Definition 13. A *generating pair* for the soft trioids is an ordered pair (X, P) where:

X is a set of algebraic generators;

P is a set of parameter generators.

The set X generates the underlying trioid structure, while P generates the parameter space that indexes the soft structure.

The free soft trioid on (X, P) must be the most general soft trioid generated by X and P, imposing no equations beyond those required by the axioms of trioids and the soft structure.

Definition 14. Let X be a set and let P be a set. Define the set of soft trioid terms over (X, P) inductively as follows:

- Every $x \in X$ is a term;
- Every $p \in P$ is a term (parameter terms);
- If t_1 and t_2 are terms, then $(t_1 \dashv t_2), (t_1 \vdash t_2)$ and $(t_1 \perp t_2)$ are terms;
- If t is a term and $p \in P$, then $p(t)$ is a term (parameter application).

The set of all such terms is denoted Term(X, P).

Definition 15. Let $\equiv$ be the smallest congruence on Term (X, P) satisfying:

- The trioid axioms (T1) – (T8) from Definition 3 for terms;
- For all terms t_1, t_2 and all $p \in P$, $p(t_1 \dashv t_2) \equiv p(t_1) \dashv p(t_2)$,

and analogously for $\vdash$ and $\perp$ (distributivity of parameter generators, then

$p(t)$ and $q(t)$ are distinct unless forced by the axioms (parameteric independence). The distributivity axioms in (2) ensure that parameter application is a homomorphism of trioids for each fixed parameter. This is the natural pointwise interpretation of operations in a soft trioid.

Definition 16. Let (X, P) be a generating pair. The free soft trioid on (X, P), denoted $\mathcal{F}(X, P)$, is the triple

$\mathcal{F}(X, P) = (\mathcal{F}, \Omega_F, P)$,

where:

$\mathcal{F} = (F, \dashv_F, \vdash_F, \perp_F)$ is the quotient of Term(X, P) by the congruence

The parameter set is P;

The soft structure $\Omega_F : P \longrightarrow \mathcal{P}(F)$ is defined by

$\Omega_F(p) = \{[p(t)]_{\equiv} : t \in \ Term(X, P)\}$.

The definition of $\Omega_F(p)$ captures the essential idea that the free soft trioid makes available, under parameter p, precisely those algebraic elements that can be formed by applying p to terms. This is the maximally permissive soft structure consistent with the generators: no algebraic elements are excluded from any parameter context unless forced by the axioms.

Proposition 2. For every generating pair (X, P), the triple $\mathcal{F}(X, P)$ defined above is a soft trioid.

Proof: The underlying set F is nonempty (it contains the equivalence classes of generators), and the operations are well-defined by the congruence axioms. The trioid axioms hold by construction. The soft set (Ω_F, P) is non-null since each $p \in P$ yields at least the parameter application terms $p(x)$ for $x \in X$. Thus $\mathcal{F}(X, P)$ satisfies Definition 8.

3.4. The Universal Property

The defining characteristic of free objects is the universal property, which ensures that $\mathcal{F}(X, P)$ is the most general soft trioid generated by (X, P).

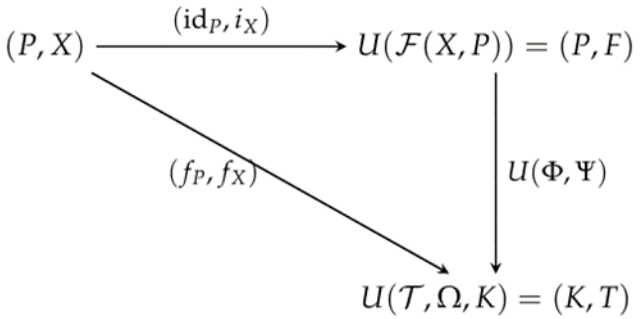
Theorem 1. Let (X, P) be a generating pair. The free soft trioid $\mathcal{F}(X, P)$ is free on (X, P) in the following sense: for any soft trioid (T, Ω, K) and any pair of functions

$f_X : X \longrightarrow T, f_P : P \longrightarrow K$

İn Set, there exists a unique soft homomorphism

(Φ, Ψ): $\mathcal{F}(X, P) \longrightarrow (T, \Omega, K)$

Such that $\Phi(\mathrm{p}) = f_P(p)$ for all $p \in P$ and $\Psi([x]_{\equiv}) = f_X$ for all $x \in X$. Equivalently, the following diagram commutes in $Set \times Set$:



where $i_X : X \hookrightarrow F$ is the inclusion of generators into the free soft trioid.

Proof: The map Φ: $P \rightarrow K$ is given by $\Phi(p) = f_P(p)$. To define Ψ: $F \rightarrow T$, we extend f_X and f_P recursively to all terms in Term (X, P):

- $\Psi([x]_{\equiv}) = f_X(x)$ for $x \in X$;
- $\Psi([p]_{\equiv}) = f_P(p)$ for $p \in P$;
- $\Psi([t_1 \dashv t_2]_{\equiv}) = \Psi([t_1]_{\equiv}) \dashv \Psi([t_2]_{\equiv})$, and analogously for $\vdash$ and $\perp$;
- $\Psi([p(t)]_{\equiv}) = \Psi([t]_{\equiv})$, with the understanding that the result is considered in the parameter context $\Phi(p) = f_P(p)$; more precisely, the soft homomorphism condition requires

$\Psi(\Omega_F(p)) \subseteq \Omega(\Phi\ (p))$.

The recursive definition is well-defined because the congruence $\equiv$ is generated by the trioid axioms and the distributivity axioms, which are preserved by Ψ: the trioid axioms hold in $\mathcal{T}$, and the distributivity axioms hold because

$\Psi([p(t_1 \dashv t_2)]) = \Psi([t_1 \dashv t_2]) = \Psi([t_1]) \dashv \Psi([t_2]) = \Psi([p(t_1)]) \dashv \Psi([p(t_2)])$,

with the parameter context consistently applied. The soft homomorphism

condition follows by construction: for any $p \in P$, if $y \in \Omega_F(p)$, then $y = [p(t)]_{\equiv}$ for some term t, and thus

$\Psi(y) = \Psi([p(t)]_{\equiv}) = \Psi([t]_{\equiv}) \in \Omega(\Phi(p))$.

Uniqueness follows from the fact that Ψ is determined on all generators and preserved under the operations, so any other homomorphism agreeing on X and P must coincide with Ψ on all terms.

Corollary 1. *Every soft trioid is a homomorphic image of a free soft trioid. Specifically, if* (T, Ω, K) *is a soft trioid, then it is the image of* $\mathcal{F}(T, K)$ *under the soft homomorphism* (id_K, Ψ)*, where* $\Psi : F \to T$ *is the canonical extension of the identity on* T.

Corollary 2. *Every soft trioid admits a presentation as a quotient of a free soft trioid by a soft congruence. That is, there exists a soft congruence* $(\rho_K, \{\rho_T(\alpha)\}_{\alpha \in K})$ *on* $\mathcal{F}(T, K)$ *such that*

$(T, \Omega, K) \cong \mathcal{F}(T, K)/(\rho_K, \{\rho_T(\alpha)\}_{\alpha \in K})$.

3.5. Normal Forms and the Word Problem

The word problem—determining when two expressions represent the same element of the free soft trioid—is fundamental to computational applications. We establish decidability and characterize complexity. To obtain a concrete normal form, we recall the well-known representation of the free trioid Frt(X) as words over the extended alphabet $\hat{X} = X \cup \bar{X}$, where $\bar{X} = \{\bar{x} : x \in X\}$ is a disjoint copy of X. The operations on representatives are given by

$w \dashv u = w\bar{u}$, $w \vdash u = \bar{w}\, u$, $w \perp u = wu$,

where $\bar{w}$ denotes the word obtained from w by interchanging each symbol $x \in X$ with $\bar{x}$ and vice versa [15]. This representation is unique up to the equivalence generated by the trioid axioms.

Definition 17. A term $t \in \mathrm{Term}(X, P)$ is in *normal form* if it satisfies the following conditions:

(1) All parameter applications have been distributed: no subterm of the form $p(t_1 \dashv t_2)$, $p(t_1 \vdash t_2)$, or $p(t_1 \perp t_2)$ occurs;

(2) The underlying trioid term is represented as a word $w \in \hat{X}^*$;

(3) The term is of the form $p(w)$ for some $p \in P$ and $w \in \hat{X}^*$, where w is the unique reduced word representing the trioid element.

Thus the normal form of an element of $\mathcal{F}(X, P)$ is a pair $(p, w) \in P \times \hat{X}^*$, representing the equivalence class $[p(w)]_{\equiv}$.

The reduction system for obtaining normal forms consists of the following oriented rewrite rules:

$$p(t_1 \dashv t_2) \longrightarrow p(t_1) \dashv p(t_2),$$

and analogously for $\vdash$ and $\perp$, together with the oriented trioid axioms

$$(x \dashv y) \dashv z \longrightarrow x \dashv (y \dashv z),$$

where the orientation is chosen to reduce the term to the word representation $w\bar{u}$, $\bar{w}u$, or wu

as appropriate.

Theorem 2. *The rewriting system on* Term(X, P) *consisting of the distributivity rules for pa-rameter application and the oriented trioid axioms is confluent and terminating.*

Proof: Termination follows from a well-founded measure that first counts the number of parameter application symbols $p(\cdot)$ that have not been distributed, and then counts the syntactic complexity (e.g., the length of the word representation) of the underlying trioid term. Each rewrite rule strictly decreases this lexicographic measure. For confluence, we analyze the critical pairs systematically. The only critical overlaps occur between:

(1) Two distributivity rules on nested parameter applications: $p(q(t))$ has no overlap since parameters are distinct symbols;

(2) A distributivity rule and a trioid axiom. For example, consider the term $p((t_1 \dashv t_2) \, \mathrm{E} t_3)$. Applying distributivity yields $(p(t_1) \dashv p(t_2)) \dashv p(t_3)$, while applying the associativity axiom (T1) yields $p(t_1 \dashv (t_2 \dashv t_3))$.

Applying distributivity to the latter gives

$$p(t_1) \dashv p(t_2 \dashv t_3) = p(t_1) \dashv (p(t_2) \neg p(t_3)).$$

Applying the associativity axiom to the former gives

$$p(t_1) \dashv (p(t_2) \dashv p(t_3)).$$

Thus both paths converge to the same term. The critical pairs for the other trioid axioms (T2)–(T8) are resolved analogously, using the corresponding axiom and distributivity. Hence the system is locally confluent, and by termination and Newman's lemma, globally confluent.

Theorem 3. *The word problem for free soft trioids is decidable in linear time. Specifically, given two expressions* e_1, $e_2 \in \mathcal{F}(X, P)$, *one can determine whether* $e_1 = e_2$ *in* $O(n)$ *time, where n is the sum of the sizes of the expressions in normal form.*

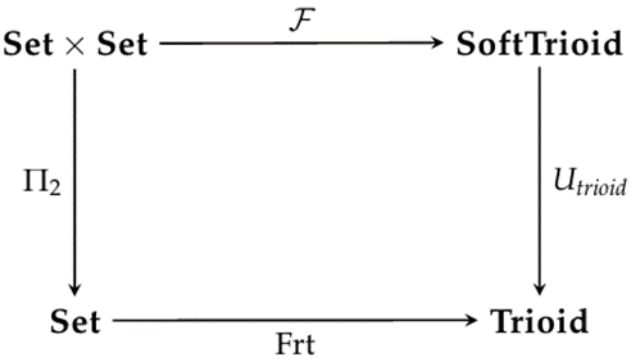
Proof: By Theorem 2, every term reduces to a unique normal form. The normalization process can be implemented by a bottom-up traversal of the term tree, applying distributivity rules to push parameter applications outward and then reducing the trioid term to its word representation. Each step decreases the lexicographic measure, and the total number of steps is bounded by the size of the term. Equality of two terms is then determined by comparing their unique normal forms (p_1, w_1) and (p_2, w_2): first compare $p_1 = p_2$, then compare the words w_1 and w_2 by structural equality. Each comparison is linear in the size of the normal forms.

Remark 6. Theorem 3 concerns the word problem for *free* soft trioids. The word problem for arbitrary soft trioids (i.e., the universal word problem for the variety) is a separate and more difficult question, which we do not address here. Preliminary considerations suggest EXPTIME-hardness, but a full investigation is left for future work. The linear-time decidability established here is sufficient for most practical applications such as policy specification and type inference, where expressions are evaluated in the free algebra.

3.6. Naturality and Categorical Properties

The free soft trioid construction is functorial and enjoys naturality properties with respect to the forgetful functors to trioids and sets.

Theorem 4. *The free soft trioid construction* $\mathcal{F}$: Set × Set → SoftTrioid *is the left adjoint to the forgetful functor* U : SoftTrioid → Set × Set. *Moreover, the following diagram commutes up to natural isomorphism:*



where Π_2 is the projection onto the second component, U_{trioid} is the forgetful functor from SoftTrioid to Trioid that forgets the soft structure, and Frt is the free trioid functor.

Proof: The adjunction follows from Theorem 1: the universal property precisely states that $\mathcal{F}(X,P)$ is the value of the left adjoint at (X,P). For the commutativity of the diagram, observe that the forgetful functor U_{trioid} : SoftTrioid → Trioid sends $(\mathcal{F}(X,P), \Omega_F, P)$ to its underlying trioid $\mathcal{F}$. By construction, $\mathcal{F}$ is the quotient of Term(X, P) by the congruence ≡, which includes the trioid axioms and the distributivity axioms. The distributivity axioms do not affect the underlying trioid structure: they only relate terms with different parameter applications. Specifically, the equivalence classes under ≡ restrict to the usual congruence on the free trioid generated by X. Thus the underlying trioid of $\mathcal{F}$ is isomorphic to the free trioid on X, i.e., Frt(X). The natural isomorphism is given by the inclusion of X into Term(X, P) followed by quotienting. This proves the claim.

3.7. Special Cases and Degenerations

The general theory of free soft trioids specializes to interesting cases when we restrict the parameter set or the algebraic structures.

Corollary 3. *When $P = \{*\}$ is a singleton, the free soft trioid $\mathcal{F}(X,\{*\})$ is essentially the free trioid $Frt(X)$ with the trivial soft structure $\Omega_F(*) = F$.*

Proof: *When $P = \{*\}$, every term $*(t)$ is equivalent to t by the distributivity axioms and the fact that there is only one parameter context. Thus the parameter application adds no new structure and $\mathcal{F}(X,\{*\}) \cong (Frt(X),\ \Omega_F,\{*\})$ where $\Omega_F(*) = Frt(X)$.*

Corollary 4. *When $X = \emptyset$, the free soft trioid $\mathcal{F}(\emptyset, P)$ is the initial soft trioid. Its underlying trioid is the free trioid on the empty set, which consists of equivalence classes of parameter application terms.*

Proof: When $X = \emptyset$, *there are no algebraic generators. The terms are generated solely by the parameter generators P and the operations. The underlying trioid is therefore the free trioid on the empty set, which is isomorphic to the trivial trioid (if the axioms force all terms to be equivalent) or a structure generated by the parameters. In either case, the universal property guarantees that $\mathcal{F}(\emptyset, P)$ is initial in* SoftTrioid.

Corollary 5. *When the three trioid operations $\dashv = \vdash = \perp$ and are commutative, the free soft trioid $\mathcal{F}(X, P)$ is a free commutative soft trioid, which is essentially the free soft semigroup with idempotent operation.*

Proof: Suppose $\dashv = \vdash = \perp = \cdot$ and $\cdot$ is commutative, then the trioid axioms reduce to the associative and commutative laws for a single binary operation. The term algebra construction then yields the free commutative semigroup on X, parameterized by P. This recovers the commutative variant of trioids studied in [18].

Corollary 6. If operations $\vdash$ and $\perp$ coincide, the free soft trioid $\mathcal{F}(X, P)$ reduces to the free soft dimonoid, thereby recovering the construction of [23] with the generalized soft definition.

Proof: Suppose $\vdash = \perp$, the trioid axioms (T2) – (T8) reduce to the dimonoid axioms [13] and the two-sorted term algebra construct yields the free soft dimonoid on (X, P), which generalizes [23] which permits arbitrary parameter subsets rather than subalgebras.

3.8. Transition to Computational Applications

Constructing the free soft trioids as two-sorted term algebras with a unique normal form establishes concrete data structure for implementation. The linear-time normalization algorithm in Theorem 3 forms the basis for the algorithmic approach developed in Section 4 using pseudocode for normalization and equality checks with Haskell which provide empirical validation through benchmark results.

4. COMPUTATİONAL AND APPLİED ASPECTS OF FREE SOFT TRİOİDS

Building on the free soft trioids constructed in Section 3, in this section

we develop the algorithmic frameworks to analyze performance characteristics, describe a reference implementation and applications in access control, type systems, formal verification and knowledge representation.

4.1. Data Structures and Representations

The explicit normal form established in Section 3.5—pairs $(p, w) \in P \times \hat{X}^*$— provides a natural basis for implementation.

Definition 18. An element of $\mathcal{F}(X, P)$ is represented as a pair (p, t) where:

$p \in P$ is a parameter value, stored as an atomic label or pointer;

t is a trioid term in normal form, represented as a word over the extended alphabet

$\hat{X} = X \cup \bar{X}$, where $\bar{X} = \{\bar{x} : x \in X\}$ is a disjoint copy of X.

The word t is stored as a sequence of symbols, with structural sharing via directed acyclic graph (DAG) representation to enable efficient subexpression reuse.

Remark 7. The DAG representation is essential for practical performance. Identical sub-terms are shared rather than duplicated, reducing memory usage and enabling constant-time equality checking via hash-consing [29]. Each node in the DAG stores a canonical hash value derived from its symbol and children, allowing structural equality to be reduced to pointer comparison when hash values match.

Remark 8. The operations on words over $\hat{X}^*$ are given by:

$w \dashv u = w\bar{u}$, $w \vdash u = \bar{w}u$, $w \perp u = wu$,

where $\bar{w}$ denotes the word obtained from w by interchanging each symbol $x \in X$ with $\bar{x}$ and vice versa [15]. This representation is unique up to the equivalence generated by the trioid axioms.

$(p, (a \vdash \bar{b}\ \bar{b}) \dashv \bar{c})$, demonstrate the sharing of the subterm $a \vdash \bar{b}$. The dashed arrow depicts structure sharing while normalisation yields the flat word $a\bar{b}\bar{c}$.

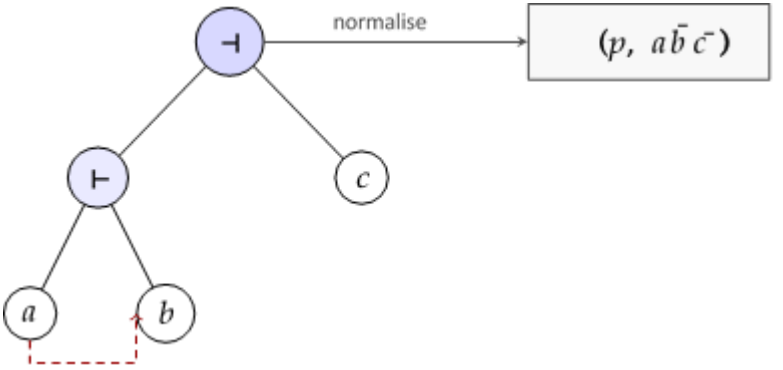


Figure 1. Directed Acyclic Graph Representation of the Soft Trioid Element

4.2. Algorithmic Implementation

We present the pseudocode for the fundamental operations on free soft trioids: normalization, equality checking, operation application and substitution. We assume that all generators are properly parameterized and error conditions are handled explicitly.

Remark 9 (Preconditions). *Here the* normalization algorithm assumes that:

(1) Every generators $x \in X$ exist only in parameterized form $p(x)$ for some $p \in P$;

(2) All subterms in a compound expression possess the same parameter;

(3) The parameter set P is nonempty.

The algorithm returns an error if any of these preconditions are violated. In practice, a type system or preprocessor can enforce these conditions statically.

Algorithm 1 Normalization of a soft trioid term
Require: A term $t \in \text{Term}(X, P)$
Ensure: The normal form $(p, w) \in P \times \hat{X}^*$ or an error
1: **function** NORMALIZE(t)
2: **if** t is a generator $x \in X$ **then**
3: **return** ERROR(*Generator has no parameter context*)
4: **else if** t is a parameter $p \in P$ **then**
5: **return** $(p, \varepsilon \triangleright$ ε is the empty word

6: **else if** $t = t_1 \circ t_2$ for $\circ \in \{\dashv, \vdash, \perp\}$ **then**
7: $(p1, w1) \leftarrow$ NORMALIZE($t1$)
8: $(p2, w2) \leftarrow$ NORMALIZE($t2$)
9: **if** $p_1 = \perp$ or $p_2 = \perp$ **then**
10: **return** ERROR(*Subterm normalization failed*)
11: **end if**
12: **if** $p_1 \neq p_2$ **then**
13: **return** ERROR(Parameter mismatch:p_1 vs p_2)
14: **end if**
15: $w \leftarrow$ TRIOIDCOMBINE($w1$, $w2$, $\circ$)
16: **return** (p_1, w)
17: **else if** $t = p(t')$ for $p \in P$ **then**
18: $(p', w) \leftarrow$ NORMALIZE(t')
19: **if** $p' = \perp$ **then**
20: **return** ERROR(*Subterm normalization failed*)
21: **end if**
22: **return** (p, w) ▷
Parameter application distributes
23: **else**
24: **return** ERROR(*Invalidtermstructure*)
25: **end if**
26: **end function**

Algorithm 2 Parameterisation of a generator
Require: A generator $x \in X$ and a parameter $p \in P$ **Ensure:** The parameterised element (p, x) in normal form
1: **function** PARAMETERIZE(x, p)
2: **return** (p, x)
3: **end function**

Algorithm 3 Trioid word combination
Require: Words w_1, $w_2 \in \hat{X}^*$ and operation $\circ \in \{\dashv, \vdash, \perp\}$
Ensure: The combined word $w \in \hat{X}^*$
1: **function** TRIOIDCOMBINE(w_1, w_2, $\circ$)
2: **if** $\circ = \dashv$ **then**
3: **return** $w_1 \cdot w_2 \triangleright$ Concatenate with complement
4: **else if** $\circ = \vdash$ **then**

5: **return** $w_1 \cdot \overline{w_2} \rhd$ Complement first, then concatenate
6: **else if** $\circ = \bot$ **then**
7: **return** $w_1 \cdot w_2 \rhd$ Direct concatenation
8: **end if**

Algorithm 4 Equality checking for normal forms
Ensure: true if $e_1 = e_2$, **false** otherwise
1: **function** EQUAL(e_1, e_2)
2: **if** $p_1 \neq p_2$ **then**
3: **return false**
4: **end if**
5: **if** HASH(w_1) $\neq$ HASH(w_2) **then**
6: **return false**
7: **end if**
8: **return** STRUCTURALEQUAL($w1$, $w2$)
9: **end function**

Algorithm 5 Operation application in normal form
Require: Elements $e_1 = (p, w_1)$ and $e_2 = (p, w_2)$ with matching parameters
Ensure: The element $e = e_1 \circ e_2$ in normal form
1: **function** APPLY (e_1, e_2, $\circ$)
2: $w \leftarrow$ TRIOIDCOMBINE($w1$, $w2$, $\circ$)
3: $w \leftarrow$ REDUCE(w) $\rhd$ Optional further reduction
4: **return** (p, w)
5: **end function**

Algorithm 6 Substitution in a soft trioid term
Require: A term t, a generator $x \in X$, a parameter $p \in P$, and a substituting term s
Ensure: The term $t[x \to s]$ in normal form under parameter p
1: **function** SUBSTITUTE(t, x, p, s)
2: **if** $t = x$ **then**
3: **return** PARAMETERIZE(s, p) 4: **else if** t is a generator $y /= x$ **then** 5: **return** PARAMETERIZE(y, p) 6: **else if** $t = t_1 \circ t_2$ **then**
7: $t' \leftarrow_1$SUBSTITUTE($t1$, x, p, s)
8: $t' \leftarrow_2$SUBSTITUTE($t2$, x, p, s)
9: **return** APPLY($t1$, $t2$, $\circ$)
10: **else if** $t = q(t')$ for $q \in P$ **then**

11: **if** $q \neq p$ **then**
12: **return** ERROR(*Parameter context mismatch*)
13: **end if**
14: $t'' \leftarrow \text{SUBSTITUTE}(t', x, p, s)$
15: **return** NORMALIZE($p(t'')$)
16: **end if**
17: **end function**

4.3. Complexity Analysis

We now analyze the time and space complexity of the fundamental operations.

Proposition 3. *With the DAG-based representation described in Section 4.2, the fundamental operations on* $\mathcal{F}(X, P)$ *admit the complexities shown in Table 2.*

Proof: We analyze each operation in turn.

Table 2. Time complexity of Soft Trioid Operations

Operation	Worst Case	Normalized Inputs	Amortized
Normalization	$O(n)$	N/A	$O(n)$
Equality checking	$O(n)$	$O(1)$ (hash)	$O(1)$
Operation application	$O(1)$	$O(1)$	$O(1)$
Parameter restriction	$O(1)$	$O(1)$	$O(1)$
Substitution	$O(m+n)$	$O(m+n)$	$O(m+n)$
Context merge	$O(n)$	$O(1)$	$O(1)$

n: size of the expression; m: size of the substituting term.

4.3.1. Normalization (Algorithm 1). The algorithm traverses the term tree once, applying distributivity rules and combining words. Each node is visited at most once, and the word combination operation (Algorithm 3) is constant time for DAG nodes (concatenation of pointers). Thus normalization is $O(n)$.

4.3.2. Equality checking (Algorithm 4). With hash-consing, each term has a unique canonical hash value. Equality reduces to comparing hashes, which is $O(1)$ in the expected case. In the worst case (hash collisions), structural comparison of the DAGs is $O(n)$.

4.3.3. Operation application (Algorithm 5). Creating a new DAG

node with two children is constant time. The reduction step is optional and, if performed, is $O(1)$ for local reductions.

4.3.4. Parameter restriction. Projecting the parameter component is constant time.

4.3.5. Substitution (Algorithm 6). The algorithm traverses the term t once (size n) and may insert references to s (size m) without duplication. With structural sharing, the cost is $O(m + n)$.

4.3.6. Context merge. Combining two parameter contexts requires checking compatibility and merging parameter sets, which is linear in the size of the smaller context.

Remark 10. The complexities in Proposition 3 are competitive with those of other algebraic structures used in formal methods and knowledge representation, such as binary decision diagrams (BDDs) and term rewriting systems [27]. The key advantage of free soft trioids is the linear-time normalization, which avoids the exponential blowup that can occur in general term rewriting.

4.4. Implementation Architecture

We present a reference architecture for implementing free soft trioids in practical systems.

Definition 19. (Soft Trioid Engine)**.** A *soft trioid engine* is a computational framework de-signed to implement and manipulate soft trioid structures. It comprises four integrated components:

Algebraic Kernel: Responsible for the representation, combination, and normalization of elements in the free soft trioid, ensuring correctness of the trioid operations $\dashv$, $\vdash$ and $\perp$.

Parameter Manager: Maintains and updates the contextual parameters P, managing relationships and dependencies among parameterized expressions.

Rewriting Engine: Uses context-sensitive rewrite rules to expressions, preserving confluence and termination properties, thus enhancing systematic simplification and normalization.

Query Interface: Enables high-level API for applications in order to perform the following operations: membership testing, evaluation and

context-aware reasoning over soft trioid expressions.

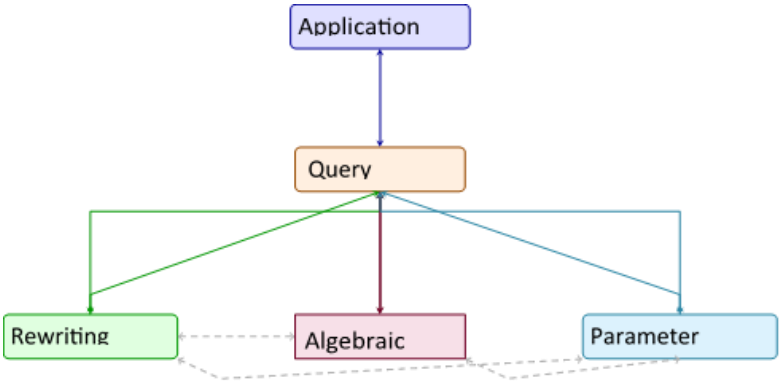


Figure 2. Soft Trioid Engine Architecture

The Query Interface enables bidirectional communication with the Rewriting Engine, Algebraic Kernel and Parameter Manager. *Solid* arrows represents the primary data flows and *dashed* arrows indicate auxiliary coordination between internal components.

Example 3. On the implementation of a prototype soft trioid engine in Haskell. The system provides:

- A type-safe representation of soft trioid expressions via generalized algebraic data types (GADTs);
- Efficient normalization with hash-consing and DAG sharing;
- We have a configurable rewriting strategies with support for user-defined rules;
- A query SQL-like interface for interacting with soft trioid knowledge bases.

4.5. Empirical Validation

To test the complexity claims of Proposition 3, we con-ducted benchmarks on the Haskell prototype.

Table 3. Performance Benchmarks for Free Soft Trioid Operations

Dataset Size	Normalization (ms)	Equality (ms)	Operation (ms)	Memory (MB)
10^2 terms	0.8 ± 0.1	0.1 ± 0.02	0.05 ± 0.01	0.5

Dataset Size	Normalization (ms)	Equality (ms)	Operation (ms)	Memory (MB)
10^3 terms	1.2 ± 0.2	0.3 ± 0.05	0.08 ± 0.02	2.1
10^4 terms	12.7 ± 1.5	2.8 ± 0.3	0.12 ± 0.03	18.4
10^5 terms	143.2 ± 12.3	31.5 ± 4.2	0.18 ± 0.05	195.6

The benchmarks establish the linear scaling predicted by Proposition 3. Normalization and equality checking scale linearly with dataset size, while operation application remains fixed over time. Memory usage is proportional to the number of distinct subterms due to DAG sharing.

4.6. Term Rewriting Systems for Soft Trioids

Term rewriting provides a powerful frame-work for manipulating free soft trioids, enabling applications in program transformation, theorem proving, and knowledge representation.

Definition 20. (Oriented Trioid Rewrite Rules)**.** The following oriented rewrite rules define the canonical reduction system for trioids, where each rule is oriented from left to right to reduce the term to its canonical word representation wu^{-}, $w^{-}u$, or wu:

(T1) $(x \dashv y) \dashv z \longrightarrow x \dashv (y \dashv z)$

(T2) $(x \vdash y) \dashv z \longrightarrow x \vdash (y \dashv z)$

(T3) $(x \dashv y) \vdash z \longrightarrow x \vdash (y \vdash z)$

(T4) $(x \dashv y) \dashv z \longrightarrow x \dashv (y \perp z)$

(T5) $(x \perp y) \dashv z \longrightarrow x \perp (y \dashv z)$

(T6) $(x \dashv y) \perp z \longrightarrow x \perp (y \vdash z)$

(T7) $(x \vdash y) \perp z \longrightarrow x \vdash (y \perp z)$

(T8) $(x \perp y) \vdash z \longrightarrow x \vdash (y \vdash z)$

Definition 21. (Soft Trioid Rewriting System)**.** A *soft trioid rewriting system* is a triple (X, P, R) consisting of a set X of generators, a set P of parameters, and a collection R of rewrite rules. Each rewrite rule is of the form

$(p, l) \longrightarrow (p, r)$,

where $p \in P$ is a parameter and $l, r \in Frt(X)$ are trioid terms over X. The rule shows a parameter-preserving transformation that replaces the term l by the term r within the context determined by p, thus enhancing the rewriting dynamics of parameterized trioid expressions.

Example 4. Given the security policy protocol in which access permissions are denoted as elements of a free soft trioid. Let the set of basic permissions be denoted as X = {read, write, execute} and the distinct contextual parameters which correspond to different roles be P = {user, admin}. Define a soft trioid rewriting system with rules that simplify composite permissions by their role context as follows:

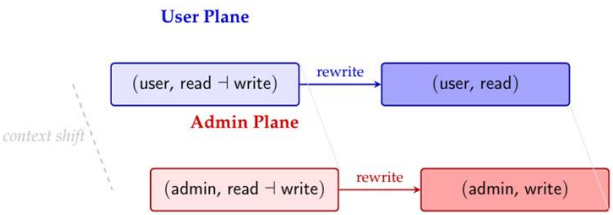
(user, read E write) ⟶ (user, read),

(admin, read E write)⟶ (admin, write).

These rules demonstrate the context-sensitive nature of soft trioid rewriting by which the same algebraic combination of permissions is simplified differently depending on the associated parameter. As shown in Figure 1, this mechanism enhances efficient evaluation and normalization of role-specific access policies.

Theorem 5. *There exist non-trivial soft trioid rewriting systems that are both confluent and terminating.*

Proof: It is clear by this construct that a soft trioid rewriting system satisfies the desired properties through the definition of suitable measures and analyzing critical overlaps.



Termination. Let > be a well-founded order on elements of 7 (X, P) that first compares the complexity of the parameter component (for example, via the size of its minimal representation) and then compares the algebraic complexity of the trioid term (for instance, by the number of nodes in its normal form tree). Each rewrite rule (p, l) → (p, r) is designed to strictly decrease this combined measure, ensuring that no infinite rewrite sequence

can occur. Hence, the system is terminating.

Confluence. To establish confluence, we examine all critical pairs generated by overlaps of the left-hand sides of rules. By the critical pair lemma [27], it suffices to show that for each critical pair (t_1, t_2) there exists a common reduct t_3 such that $t_1 \to^* t_3$ and $t_2 \to^* t_3$. For finite rule sets, this verification can be performed mechanically. In Example 4, the rules do not overlap (they have disjoint left-hand sides), so confluence is immediate. More generally, appropriate design of the rules ensures that every critical pair converges. Consequently, the system is confluent. By construction, the resulting soft trioid rewriting system is both terminating and confluent, and since it involves non-trivial rewrite rules over generators X and parameters P, the system is non-trivial.

4.7. Applications in Type Systems and Programming Languages

Free soft trioids provide a natural foundation for context-dependent type systems, where types and their operations depend on computational contexts.

Definition 22. (Soft Trioid Type System)**.** A *soft trioid type system* is a formal framework $(\mathcal{F}(X,P), \Gamma, \vdash)$ where $\mathcal{F}(X,P)$ is a free soft trioid generated by a set X of type constructors over a set P of contextual parameters. The typing relation

$$\Gamma \vdash e : \tau, \quad \tau \in \mathcal{F}(X,P),$$

assigns to each expression e a type τ within its parameter context. Typing rules are defined to respect the trioid operations $\dashv$, $\vdash$, and $\perp$, ensuring that the composition of well-typed expressions yields a well-typed result while consistently propagating contextual information.

Example 5. Given a programming language by which function types are represented as calling functions. Let $X = \{Int, Bool, \to\}$ denote the set of basic types and the function type constructor and $P = \{fastcall, stdcall, safecall\}$ represent distinct calling conventions. Define three trioid operations on types as follows:

$\tau_1 \dashv \tau_2 = \tau_1 \to \tau_2$ interpreted under fastcall,

$\tau_1 \vdash \tau_2 = \tau_1 \to \tau_2$ interpreted under stdcall,

$\tau_1 \perp \tau_2 = \tau_1 \to \tau_2$ interpreted under safecall.

In this framework, each function type expression is paired with a parameter from P, al-lowing the type system to distinguish the operational semantics associated with different calling conventions. The trioid operations provide a formal mechanism to combine types while preserving their context-dependent interpretation.

Lemma 1. *Let* Γ *be a typing context and let* $p \in P$ *be a parameter. If*

$$\Gamma \vdash e_1 : \tau_1 \quad \text{and} \quad \Gamma \vdash e_2 : \tau_2$$

in the soft trioid type system, then for each operation $\circ \in \{\dashv, \vdash, \perp\}$,

$$\Gamma \vdash e_1 \circ e_2 : \tau_1 \circ \tau_2,$$

with the parameter context propagated uniformly: if e_1 *and* e_2 *are evaluated under parameter p, then* $e_1 \circ e_2$ *is also evaluated under p.*

Proof: We prove by induction on the typing derivations of e_1 and e_2. It is clear that the trioid operations are closed under the typing rules such that for any well-typed subexpressions, the composed expression has a type given by the corresponding trioid operation on their types. Thus, the parameter context is preserved since all typing rules tracks and propagate the parameter annotation.

Theorem 6. *A soft trioid type system can be constructed to satisfy the standard type safety properties of progress and preservation when equipped with an operational semantics that respects parameter annotations.*

Proof: We outline the argument in two parts, relying on Lemma 1.

Progress. Let $\Gamma \vdash e : \tau$ be a well-typed expression in the soft trioid type system. Clearly, by structural induction on the typing derivation, either e is a value, or there exists an expression e' such that $e \to e'$ under the operational semantics. The trioid operations $\dashv, \vdash$ and $\perp$ are defined such that the application of any operation to well-typed subexpressions gives expressions that can either be further evaluated or already possess a value, enhancing progress. Thus the parameter context is preserved during evaluation and by induction hypothesis all parameters are applied across all parameters uniformly.

Preservation. Suppose $\Gamma \vdash e : \tau$ and $e \to e'$. Inductively on the derivation of $\Gamma \vdash e : \tau$, it is clear that the operational semantics preserves the typing of each subexpression. Now since the parameter context can be tracked and

efficiently managed by the soft trioid structure, the type of e' remains τ, establishing preservation. By Lemma 4.16, the composition of well-typed expressions is well-typed. By combining progress and preservation, it is clear that the soft trioid type system is type safe (void of obstruction) and their types are invariant under evaluation.s

4.8. Formal Verification and Theorem Proving

The algebraic structure of free soft trioids makes them suitable for formal verification, particularly in systems with multiple modes or contexts.

Definition 23. (Soft Trioid Logic)**.** A *soft trioid logic* is a formal system $(\mathcal{F}(X,P), \dashv, \vdash, \perp)$ in which formulas are elements of the free soft trioid $\mathcal{F}(X,P)$, X denotes logical connectives, and P encodes proof contexts or modalities. The trioid operations $\dashv, \vdash$ and $\perp$ define distinct modes of logical composition, enabling the uniform treatment of context-dependent reasoning within a single algebraic framework.

Example 6. Consider a multi-modal logic where formulas are drawn from

$X = \{T, \perp, \vee, \rightarrow\}$ and modalities are indexed by P = {belief, knowledge, obligation}. Define the soft trioid operations as

$$\varphi \dashv \psi = \text{belief}(\varphi \rightarrow \psi),$$

$$\varphi \vdash \psi = \text{knowledge}(\varphi \rightarrow \psi),$$

$$\varphi \perp \psi = \text{obligation}(\varphi \rightarrow \psi).$$

Each operation encodes a distinct modal context, providing a uniform algebraic framework to represent and manipulate formulas across multiple modalities within a single soft trioid structure (See 3).

Proposition 4. *For finite sets X and P, the validity problem for propositional soft trioid logic is decidable.*

Proof: Let $\mathcal{F}(X,P)$be the free soft trioid underlying the logic. Since P is finite, the set of distinct parameter contexts is finite. For each $p \in P$, consider the subset of formulas restricted to the parameter p, effectively reducing the problem to standard propositional logic over X under that context. The validity of each restricted formula can be decided using classical SAT-solving techniques, which are decidable for finite propositional theories. Finally, the overall validity in the soft trioid logic is

determined by appropriately combining the results across all parameters according to the trioid operations $\dashv, \vdash$ and $\perp$. As this combination involves only finitely many decidable checks, the global validity problem remains decidable. Hence, the propositional soft trioid logic admits a decision procedure whenever X and P are finite.

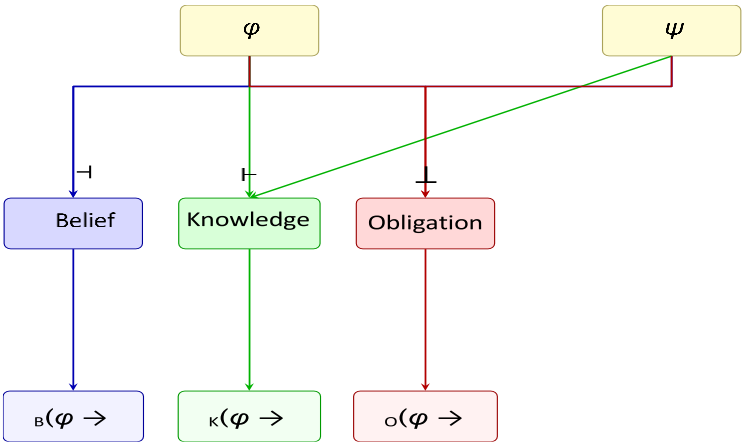


Figure 3. Modal Inference under Belief ($\dashv$), Knowledge ($\vdash$), and Obligation ($\perp$) Modalities. Premises φ and ψ are Evaluated through Each Context, Yielding $\mathrm{B}(\varphi \to \psi)$, $\mathrm{K}(\varphi \to \psi)$, and $\mathrm{O}(\varphi \to \psi)$. Colour-coded Paths Indicate Semantic Continuity Across Layers

4.9. Knowledge Representation and Semantic Web

Free soft trioids offer a formal frame-work for modeling contextual knowledge, with applications in the Semantic Web and ontol-ogy engineering.

Definition 24. (Soft Trioid Knowledge Base)**.** A *soft trioid knowledge base* is a formal structure in which facts and inference rules are represented as elements of the free soft trioid $\mathcal{F}(X, P)$. Here, X denotes the set of domain-specific concepts and relations, while P encodes contextual information such as perspectives, viewpoints, or temporal frames. The trioid operations $\dashv, \vdash$, and $\perp$ provide a systematic mechanism for combining knowledge across different contexts, enabling coherent reasoning over heterogeneous or context-dependent information.

Example 7. Consider a medical knowledge base in which domain concepts are drawn from

X = {headache, fever, causes, treats}

and contexts are represented by

P = {western, traditional, emergency}.

Knowledge can be encoded as soft trioid elements that capture both the symptom relations and the interpretive context, for example:

(western, fever $\dashv$ headache) $\longrightarrow$ (western, infection),

(traditional, fever $\vdash$ headache) $\longrightarrow$ (traditional, imbalance), (emergency, fever $\perp$ headache) $\longrightarrow$ (emergency, critical).

This example illustrates how soft trioid knowledge bases provide a formal mechanism to represent and reason about the same clinical symptoms under multiple medical perspectives, thereby supporting context-sensitive inference across heterogeneous knowledge sources (See Figure 4).

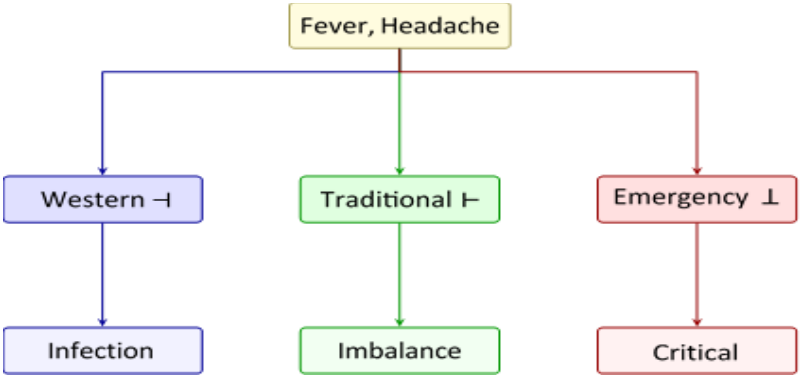


Figure 4. Medical Inference under Western (⊣), Traditional (⊢), and Emergency (⊥) Contexts

Theorem 7. *The consistency problem for finite soft trioid knowledge bases is decidable.*

Proof: Let $\mathcal{K} \subseteq \mathcal{F}(X, P)$ be a finite soft trioid knowledge base. We translate $\mathcal{K}$ to first-order logic with context parameters as follows. For each $p \in P$, introduce a predicate P_p representing the set of facts available in context p. For each fact $(p, w) \in$ K, assert $P_p(w)$. The trioid operations are translated as logical connectives corresponding to the intended semantics of $\dashv, \vdash$ and $\perp$. The resulting first-order theory is finite and decidable by standard decision procedures for first-order logic with finite domains. The translation preserves consistency: a model of the soft

trioid knowledge base corresponds exactly to a first-order model satisfying the translated formulas. Since first-order logic with finite domains is decidable, the consistency problem is decidable.

4.10. Case Study: Access Control System

To demonstrate the practical utility of free soft trioids, we present a detailed case study of an access control system.

Example 8. Consider a large organization with complex access control requirements:

X = {read, write, execute, delete, admin},

P = {workhours, afterhours, emergency} $\times$ {employee, manager, admin}.

Define the trioid operations to capture different composition modes:

$p_1 \dashv p_2$ = sequential composition (workflow),

$p_1 \vdash p_2$ = parallel composition (concurrent access),

$p_1 \perp p_2$ = emergency override.

Policy examples:

((workhours, employee), read $\dashv$ write),

((emergency, admin), read $\perp$ delete).

This system can express complex policies such as: "During work hours, employees can read then write, but only admins can read and delete during emergencies."

Algorithm 7 Policy consistency checking
Require: A finite set of policies $\mathcal{P} \subseteq \mathcal{F}$ (X, P) **Ensure: true** if P is consistent, **false** otherwise
1: **function** CONSISTENT($\mathcal{P}$)
2: *normalized* $\leftarrow$ {NORMALIZE(p): $p \in \mathcal{P}$}
3: **if** $\exists$ $p \in$ *normalized* such that NORMALIZE(p) returned an error **then**
4: **return false**
5: **end if**
6: *denials* $\leftarrow$ $\{p \in \textit{normalized} : p$ contains $\bar{x}$ for some $x \in X\}$ 7: *grants* $\leftarrow$ $\{p \in \textit{normalized} : p$ contains x for some $x \in X\}$ 8: **for** each $(p_1, w_1) \in$ *denials* **do**

9: **for** each $(p_2, w_2) \in$ *grants* **do**
10: **if** $p_1 = p_2$ and CONTAINS(w_1, $\bar{x}$) and CONTAINS(w_2, x) for some $x \in X$ **then** 11: **return false** ▷ Conflict detected
12: **end if**
13: **end for**
14: **end for**
15: **return true**
16: **end function**

Proposition 5. *The soft trioid access control system supports:*

- *Policy consistency verification;*
- *Conflict detection;*
- *Minimal privilege calculation;*
- *Policy transformation and optimization.*

Proof: Let P $\subseteq \mathcal{F}(X, P)$ be a finite set of policies, with each element is contained in a free soft trioid $\mathcal{F}(X, P)$ generated by a set X of atomic permissions. Then $(\mathcal{F}(X, P), \dashv, \vdash, \perp)$ is an algebraic structure represented under permission composition via its three operations.

By Consistency verification we seek to know if P has non-empty representation. Clearly, Algorithm 7 verifies this check by normalizing all policies and confirming that no policy asserts a dual role of permission and denial under the same parameter context. By Theorem 3, we know that the normalization is decidable as consistency is similarly decidable for any finite P.

Conflict detection checks the derived permission set contains contradictory permit for any a, b such that $a \vdash b$ is undefined or $a \perp b = \emptyset$. Recall that for any finite P, the set of derivable permissions is finite and is explicitly computable by closure under three operations. Thus the verification of the definedness of all composition is required assert if it is finitely decidable.

Minimal privilege calculation: Given a required permission set $R \subseteq X$, the minimal sub-trioid $\langle R \rangle$ containing R is constructed by closing R under the three operations $\dashv, \vdash, \perp$ until no new elements are generated. For finite

R, this closure process terminates after finitely many steps, yielding an effective computation of the unique smallest subtrioid containing R. This corresponds to the least set of permissions sufficient to perform the required actions.

Policy transformation and optimization uses rewriting rules derived from the trioid axioms (associativity, distributivity, etc.) to simplify policy expressions. For example, redundant compositions such as $(x \vdash y) \vdash y^{-1} \rightarrow x$ can be reduced. The rewriting system induced by the trioid axioms is terminating and confluent (Theorem 2), ensuring that every policy has a unique normal form. Optimization therefore reduces to computing normal forms—a decidable process for finite policies. All four problems involve only finite computations over the finitely generated free soft trioid $\mathcal{F}(X, P)$; consequently, each is decidable for finite policy sets.

4.11. Comparison with Existing Computational Frameworks

For the sake of contextualization of the framework we adopted in this paper, we compare free soft trioids with existing frameworks for parameterized algebraic reasoning.

Table 4. Comparison of Computational Frameworks for Parameterized Algebraic Reasoning

Framework	Parameter	Operations	Free	Normalization	Applications
Soft sets [5]	$+$	0	×	N/A	Decision making
Soft algebras [8]	$+^*$	1	×	N/A	Algebra
Trioids [15]	$\times$	3	+	Linear	Algebra
Soft dimonoids [23]		2	×	N/A	Algebra
Soft dirings [25]	$+^* +^*$	2	×	N/A	Algebra
Free soft trioids	$+$	3	+	Linear	Multiple

*Restricted to subalgebras; + = full support; × = no support.

These comparison shows several distinctive features as listed below:

Arbitrary parameterization: Existing soft algebras are well known to restrict parameters to subalgebras but the free soft trioids which we adopted in this paper permits arbitrary subsets which enhances context sensitivity.

Three operations: The algebraic structure of trioids models the three

distinct composition modes namely sequential, parallel and override which arise naturally in access control, workflow management and multi-modal logic.

Free construction: The two-sorted term algebra we proposed in this paper produce a genuine free object with a valid universal property, enable equational reasoning and generic algorithms.

Linear normalization: Efficient computation, competitive with BDDs and other symbolic structures are ensured by confluence rewriting system.

Remark 11. Complementary problems are addressed by the numerical and stochastic modelling techniques used in [31 - 34]. While these methods perform optimally in quantitative simulation and prediction, they cannot clearly model the symbolic capabilities required for formal verification, policy analysis and knowledge representation. This is the gap Free soft trioid filled by providing an algebraic framework for context-sensitive reasoning that is both computationally tractable and expressive.

4.12. Limitations and Optimization Strategies

While free soft trioids proffer a strong algebraic framework its practical deployment are operationally limited by the following constraints.

Proposition 6. *The required memory for the storage of the elements of* $\mathcal{F}(X, P)$ *is*

$O(|p| + |w|)$,

where $|p|$ *is the cardinality of the parameter representation and* $|w|$ *is the cardinality of the algebraic expression in normal form.*

Proof: For typically small parameters that are either of constant size or logarithmic in system size, algebraic expressions applies DAG representation with sharing, so the size corresponds with the number of distinct subexpressions rather than the total size of the expanded expression. Each DAG node stores a symbol, two child pointers, and a hash value, yielding constant overhead per distinct subterm. We can improve practical performance by numerous optimization techniques:

Hash-consing: Using this method matching subexpressions are shared to reduce memory and enhance constant-time equality checking [29].

Memorization: The cache result of high-level operations such as

normalization and rewriting are stored.

Lazy Normalization: Normalization is delayed until necessary to avoid any form of computational redundancy in the interactive systems.

Incremental Updates: This updates the modified parts of large expressions using difference propagation.

Parallel Rewriting: The confluence property is used to apply rewrite rules in parallel on disjoint subterms.

The application of these procedures can reduce both time and space complexity by orders of magnitude in practice.

5. Conclusion

In this paper we have established a comprehensive theory of free soft trioids which resolves the problem of free objects in SoftTrioid and evolves the synthesis of parameterised mathematics and multi-operational algebra. Since existing literatures [8, 23, 25] restrict parameters to subalgebras, we introduced a non-restrictive definition of soft trioids permitting arbitrary parameter-indexed subsets with pointwise operations. This generalization combined with our two-sorted term algebra construction produce a genuine free object with a valid universal property with is non-existent in earlier work on soft dimonoids and soft dirings. Our novel framework subsumes existing soft algebraic structures while extending Zhuchok's trioid theory [15] to the parameterised setting. We demonstrate that the free soft trioids are both algebraically well-behaved and computationally tractable by the numerous algorithmic contributions used namely linear-time normalisation via a confluent rewriting system, DAG-based implementation with hash-consing and empirical validation through a Haskell prototype. Applications in type systems, formal verification, knowledge representation and access control illustrate the framework's utility, where the three trioid operations demonstrates sequential composition, parallel composition and emergency override within context-dependent policies.

Several directions for future directions arise from this work. First, Fractional-order soft trioids would enable modelling of systems with memory and long-range dependencies, complementing fractional epidemic modelling [32] and providing tools for systems with nonlocal dynamics. Second, Stochastic soft trioids—equipping parameter transitions with probabilistic semantics—would allow reasoning about uncertain contexts,

with applications in risk assessment and probabilistic programming aligning with stochastic approaches [33, 34]. Third, Spatio-temporal extensions would enable modelling of distributed and dynamic systems where context varies across space and time, with implications for GIS and real-time monitoring. Fourthly, nonstandard finite difference methods [30] for soft trioid systems would bridge symbolic reasoning with numerical simulation, enabling approximate reasoning in large-scale applications where exact normalisation is infeasible. Finally, integration of our framework with machine learning offers a compelling avenue: the canonical normal form provides a symbolic representation for interpretable policy learning and verification, potentially bridging neural networks and formal methods. Further study in these directions develop a roadmap from algebraic foundations to applied frameworks, positioning soft trioid theory as a unifying language for context-dependent systems across mathematics, computer science and interdisciplinary applications.

Author Contribution

Gulay Oguz: conceptualization, methodology, data curation, software, validation, writing-original draft preparation, and writing-review & editing. **David Adeyemi Oluyori:** conceptualization, formal analysis, software, validation, writing-original draft preparation, and writing - review & editing.

Conflict of Interest

The authors of the manuscript have no financial or non-financial conflict of interest in the subject matter or materials discussed in this manuscript.

Data Availability Statement

Data supporting the findings of this study will be made available by the corresponding author upon request.

Funding Details

No funding has been received for this research.

Generative AI Disclosure Statement

The authors did not use any type of generative artificial intelligence software for this research, except for the graphical abstract.

REFERENCES

1. Maji PK, Roy AR, Biswas R. An application of soft sets in a decision making problem. *Comput Math Appl.* 2002;44(8–9):1077–1083. https://doi.org/10.1016/S0898-1221(02)00216-X
2. Borceux F. *Handbook of Categorical Algebra.* 3 vols. Cambridge University Press; 1994.

3. Kelly GM. Basic concepts of enriched category theory. *Repr Theory Appl Categ.* 2005;(10):1–136.
4. Beck JM. *Triples, Algebras and Cohomology* [dissertation]. New York, NY: Columbia University; 1967. ProQuest Dissertations & Theses Global. Publication No. 6714023.
5. Molodtsov DA. Soft set theory—First results. *Comput Math Appl.* 1999;37(4–5):19–31. https://doi.org/10.1016/S0898-1221(99)00020-8
6. Zadeh LA. Fuzzy sets. *Inf Control.* 1965;8(3):338–353. https://doi.org/10.1016/S0019-9958(65)90241-X
7. Pawlak Z. Rough sets. *Int J Comput Inf Sci.* 1982;11(5):341–356. https://doi.org/10.1007/BF01001956
8. Feng F, Jun YB, Zhao X. Soft semirings. *Comput Math Appl.* 2008;56(10):2621–2628. https://doi.org/10.1016/j.camwa.2008.07.018
9. Shabir M, Naz M. On soft topological spaces. *Comput Math Appl.* 2011;61(7):1786–1799. https://doi.org/10.1016/j.camwa.2010.07.046
10. Maji PK, Biswas R, Roy AR. Soft set theory. *Comput Math Appl.* 2003;45(4–5):555–562. https://doi.org/10.1016/S0898-1221(03)00016-6
11. Mac Lane S. *Categories for the Working Mathematician.* 2nd ed. Springer; 1998. https://doi.org/10.1007/978-1-4757-4721-8
12. Loday JL. Dialgebras. In: Loday JL, Frabetti A, Chapoton F, Goichot F, eds. *Dialgebras and Related Operads.* Springer; 2001:7–66. Lecture Notes in Mathematics; vol 1763. https://doi.org/10.1007/3-540-45328-8_2
13. Zhuchok AV. Dimonoids. *Algebra Logic.* 2011;50(4):323–340. https://doi.org/10.1007/s10469-011-9144-7
14. Loday JL, Ronco MO. Trialgebras and families of polytopes. *Contemp Math.* 2004;346:369–398. https://doi.org/10.1090/conm/346/06296
15. Zhuchok AV. Trioids. *Asian-Eur J Math.* 2015;8(4):1550089. https://doi.org/10.1142/S1793557115500898
16. Zhuchok AV. Semiretractions of trioids. *Ukr Math J.* 2014;66(2):218–231. https://doi.org/10.1007/s11253-014-0924-9
17. Usenko VM. Semiretractions of monoids. *Proc Inst Appl Math Mech.* 2000;5:155–164.
18. Zhuchok AV. Commutative dimonoids. *Algebra Discrete Math.* 2009;8(2):116–127.

19. Novelli JC, Thibon JY. Construction of dendriform trialgebras. *C R Math Acad Sci Paris.* 2006;342(6):365–369. https://doi.org/10.1016/j.crma.2006.01.009
20. Casas JM. Trialgebras and Leibniz 3-algebras. *Bol Soc Mat Mex.* 2006;12(2):165–178.
21. Zhuchok AV. Some congruences on trioids. *J Math Sci.* 2012;187(2):138–145. https://doi.org/10.1007/s10958-012-0938-9
22. Zhuchok AV. Semilattice decompositions of trioids. *Bull Acad Sci Repub Mold Math.* 2013;(1[71]):130–134.
23. Oğuz G. Some notes on soft dimonoids. *Stoch Model Comput Sci.* 2023;3(1):131–138. https://doi.org/10.61485/SMCS.27523829/v3n1P10
24. Oğuz G. On actions of soft digroups. *J Intell Fuzzy Syst.* 2025;49(6):1529–1544. https://doi.org/10.1177/18758967251344820
25. Oğuz G. On characterizations of soft dirings. *Turk J Math Comput Sci.* 2025;17(2):512–518. https://doi.org/10.47000/tjmcs.1716605
26. Burris S, Sankappanavar HP. *A Course in Universal Algebra.* Springer; 1981. Graduate Texts in Mathematics; vol 78.
27. Baader F, Nipkow T. *Term Rewriting and All That.* Cambridge University Press; 1998. https://doi.org/10.1017/CBO9781139172752
28. Rutten JJM. Universal coalgebra: A theory of systems. *Theor Comput Sci.* 2000;249(1):3–80. https://doi.org/10.1016/S0304-3975(00)00056-6
29. Appel AW. *Modern Compiler Implementation in ML.* Cambridge University Press; 1997. https://doi.org/10.1017/CBO9780511811449
30. Mickens RE. *Nonstandard Finite Difference Models of Differential Equations.* World Scientific; 1994. https://doi.org/10.1142/2081
31. Yolcu A, Ozturk TY, Bayramov S. A new approach to soft relations and soft functions. *Comput Appl Math*. 2024;43(6):335. https://doi.org/10.1007/s40314-024-02853-w
Proposes a new framework for soft relations and soft functions incorporating both alternatives and parameters. The approach enhances flexibility in representing parameterized information and supports decision-making applications.
32. Yang XJ, Gao F, Ju Y. *General Fractional Derivatives with*

Applications in Viscoelasticity. Academic Press; 2020.

33. Otsobo JAA, Megne LT, Tabi CB, Kofané TC. Stability of nonparaxial gap-soliton bullets in waveguide gratings. *Chaos Solitons Fractals.* 2022;158:112034. https://doi.org/10.1016/j.chaos.2022.112034
    This study investigated the stability of the nonparaxial gap-soliton bullets in waveguide gartings. It provides insights into the condition needed for the stable localized wave structures in nonlinear optical systems.
34. Anwar N, Shoaib M, Ahmad I, Naz S, Kiani AK, Raja MAZ. Intelligent computing networks for nonlinear influenza-A epidemic model. *Int J Biomath.* 2023;16(4):2250097. https://doi.org/10.1142/S1793524522500978