On the Abstraction of a Categorical Clustering Algorithm

Mina Sheikhalishahi(✉), Mohamed Mejri, and Nadia Tawbi

Department of Computer Science, Université Laval, Québec City, Canada
mina.sheikh-alishahi.1@ulaval.ca,
{mohamed.mejri,nadia.tawbi}@ift.ulaval.ca

Abstract. Despite being one of the most common approach in unsupervised data analysis, a very small literature exists on the formalization of clustering algorithms. This paper proposes a semiring-based methodology, named Feature-Cluster Algebra, which is applied to abstract the representation of a labeled tree structure representing a hierarchical categorical clustering algorithm, named CCTree. The elements of the feature-cluster algebra are called terms. We prove that a specific kind of a term, under some conditions, fully abstracts a labeled tree structure. The abstraction methodology maps the original problem to a new representation by removing unwanted details, which makes it simpler to handle. Moreover, we present a set of relations and functions on the algebraic structure to shape the requirements of a term to represent a CCTree structure. The proposed formal approach can be generalized to other categorical clustering (classification) algorithms in which features play key roles in specifying the clusters (classes).

Keywords: Formal methods · Categorical clustering · Algebraic formalization · Clustering algorithm · Semiring · Abstraction

1 Introduction

Clustering is a very well-known tool in unsupervised data analysis, which has been the focus of significant researches in different studies, spanning from information retrieval, text mining, data exploration, to medical diagnosis [2]. Clustering refers to the process of partitioning a set of data points into groups, in a way that the elements in the same group are more similar to each other rather than to the ones in other groups. The problem of clustering becomes more challenging when data are described in terms of *categorical attributes*, for which, differently from numerical attributes, it is hard to establish an ordering relationship. Although being vastly used in the literature, a very few works exist to express categorical clustering algorithms and the related issues in terms of formal methods.

This research has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

In present work, we provide an abstract representation of the clusters through the formal definition of a categorical clustering algorithm, named CCTree. This abstract representation facilitates the analysis of cluster properties, while getting rid of confronting a large amount of data in each cluster. The proposed formal scheme can also be used to formalize challenging tasks in categorical clustering algorithms, e.g. parallel clustering, feature selection.

CCTree (Categorical Clustering Tree) [11] has a decision tree-like structure, which iteratively divides the data of a node on the base of an attribute, or domain of features, yielding the greatest entropy. The division of data is represented with edges coming out from a parent node to its children, where the edges are labeled with the associated features. A node which respects the specified stop conditions is considered as a leaf. The leaves of the tree are the desired clusters. Features play a key role in the CCTree algorithm. In fact, the feature structure is used to uniquely identify each cluster.

Abstraction is a mathematical concept which maps a representation of a problem, which is called the ground (semantic), onto a new representation, which is called the abstract (syntax) representation. Through abstraction, it is possible to deal with the problem in the new space by preserving certain desirable properties and in a simpler way to handle, since it is constructed from ground representation by removing unwanted details [4].

To abstract the CCTree representation, we propose a semiring-based algebraic structure, named "*Feature-Cluster Algebra*". The elements of the proposed algebraic structure are called *terms*. We show that the proposed approach, under some conditions, is able to fully abstract the labeled tree structure. The full abstraction guarantees that the semantic and syntax forms of a problem can be used alternatively. Furthermore, we present a set of functions and relations on the feature-cluster algebra, which is used to present the conditions for a term to represent a CCTree structure.
The contributions of the present work can be summarized as follows:

- A semiring-based formal approach, named *Feature-Cluster Algebra*, is proposed to *abstract* the representation of feature-based clusters. We call the elements of the proposed algebra as *term*.
- We show that a specific term in the feature-cluster algebra, under some conditions, fully abstracts a labeled tree structure, and specifically CCTree. The full abstraction allows to apply the desired calculation on simple abstract form, and ensure that the result is also satisfied in the main semantic representation.
- We present the conditions and requirements for a term in the feature-cluster algebra to represent a CCTree structure.

The paper is organized as follows. In Section 2, we present a review of the literature on formalization methods applied in feature-based problems. In Section 3, the preliminary background notions are provided. We construct the feature-cluster algebra, the semiring based methodology for abstracting CCTree in Section 4. The process of transforming a tree structure to a term and vice versa is presented in Section 5. The relations on feature-cluster algebra are introduced in Section 6 which are used to identify the CCTree term in proposed algebraic system. We conclude and point to future directions in Section 7.

2 Related Work

In the following we present some work on feature models and the associated formal approaches.

Feature models, in computer science, were first defined as information models where a set of products, e.g. software products or DVD player products, are represented as hierarchically arrangement of features, with different relationships among features [1]. Feature models are used in many applications as the result of being able to model complex systems, being interpretable, and ability to handle both ordered and unordered features [10]. Benavides et. al. [1] argue that designing a family of software system in terms of features, makes it easy to be understood by all stakeholders, rather than the time they are expressed in terms of objects or classes. Representing feature models as a tree of features, was first introduced by Kang et. al. [9] to be used in software product line. Some studies [3] show that tree models combined with ensemble techniques, lead to an accurate performance on variety of domains. In feature model tree, differently from CCTree, the root is the desired product, the nodes are the features, and different representation of edges demonstrates the mandatory or optional presence of features. Hofner et. al. [8] [7], were the first who applied idempotent semiring as the basis for the formalization of tree models of products, called it *feature algebra*. The concept of semiring is used to answer the needs of product family abstract form of expression, refinements, multi-view reconciliation, and product development and classification. The elements of semiring in proposed methodology are the sets of products, or product families. We import the idea of exploited in the work to abstract the clustering algorithm.

To the best of our knowledge, we are the first to apply an algebraic structure to abstract a clustering algorithm and formalize the associated issues.

3 Background

In this section we present the preliminary notions and definitions exploited by our proposed methodology.

Categorical Clustering Tree (CCTree). The CCTree [11] is constructed iteratively through a decision tree-like structure, where the leaves of the tree are the desired clusters. The root of the CCTree contains all the elements to be clustered. Each element is described through a set of *categorical* attributes. Being categorical, each attribute may assume a finite set of discrete values, constituting its domain. At each step, a new level of the tree is generated by splitting the nodes of the previous levels, when they are not homogeneous enough. *Shannon Entropy* is used both to define a homogeneity measure called *node purity*, and to select the attribute used to split a node. In particular non-leaf nodes are divided

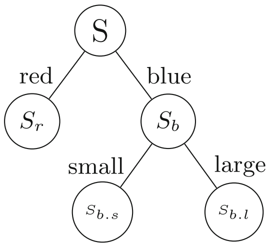

Fig. 1. A small CCTree: the root contains all the data desired to be clustered and the leaves are the desired clusters

on the base of the attribute yielding the maximum value for Shannon entropy. The separation is represented through a branch for each possible outcome of the specific attribute. Each branch or edge extracted from parent node is labeled with the selected feature which directs data to the child node [12].
Figure 1 depicts a simple CCTree.

Graph Theory Preliminaries. In graph theory [5], a *tree* is an undirected graph in which any two vertices are connected by exactly one path. A *leaf* is a vertex of degree 1. A tree is called a *rooted tree* if one vertex has been designated the root, which means that the edges have a natural orientation, towards or away from the root [5]. A tree is a labeled tree if the edges of the tree are labeled. A *branch* of a tree, refers to the path between the root and a leaf in a rooted tree [5]. A *tree structure*, in our context, is a triple (Σ, Q, δ) where Σ represents the set of edge labels; Q is the set of states or nodes; δ is the set of state (node) transition function as $\delta : Q \times \Sigma \rightarrow Q$, such that there is no cycle in transitions. The transitions connect each parent node to its children moving from root to leaves. This means that there exists a state where no transition enters it, which is the root, and there are states for them no transition exits, which are the leaves of the tree. For a transition as $\delta(s_1, f) = s_2$, we show it as the triple (s_1, f, s_2). *Graph homomorphism* ζ from a graph $G = (V, E)$ to a graph $G' = (V', E')$, written as $\zeta : G \rightarrow G'$, is a mapping $\zeta : V \rightarrow V'$ from the vertex set of G to the vertex set of G' such that $\{u, v\} \in E$ implies $\{\zeta(u), \zeta(v)\} \in E'$ [6]. If the homomorphism $\zeta : G \rightarrow G'$ is bijection whose inverse function is also a graph homomorphism, then ζ is a *graph isomorphism.* In our context, except having the same graphical structure, it is important that both $\{u, v\} \in E$ and $\{\zeta(u), \zeta(v)\} \in E'$ have the same edge label. Under this condition, we say that two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic*, we denote it as $G \approx G'$.

Abstraction Theory Preliminaries. The process of *abstraction* is related to the process of extracting from a representation of an object or subject an "*abstract*" representation that consists of a brief sketch of the original representation [4]. More precisely, the abstraction is the process of mapping a representation of a problem, called the "ground"(semantic) representation, onto a new representation, called the "abstract"(syntax) representation, such that it helps to deal with the problem in the original space by preserving certain desirable properties and is simpler to handle as it is constructed from the ground representation by " not considering the details" [4]. An abstraction can formally be written as a function $[[.]] : \mathcal{X} \to \mathcal{Y}$ from the ground representation of an object to its abstract form. We say $[[.]]$ *adequately abstracts* $\mathcal{X}$ if from the equivalence of two elements of semantic forms, we get the equivalence of their counterpart syntax forms. Formally, let show the equivalence of elements in $\mathcal{X}$ with $\simeq$ and the equivalence of elements in $\mathcal{Y}$ with $\cong$, then for adequate abstraction we have $[[X_1]] \cong [[X_2]] \ \Rightarrow\ X_1 \simeq X_2$. We say $[[.]]$ *abstracts* $\mathcal{X}$ if we have $X_1 \simeq X_2 \ \Rightarrow\ [[X_1]] \cong [[X_2]]$. In the case that the above relations respect simultaneously, we say $[[.]]$ *fully abstracts* $\mathcal{X}$, i.e. we have $[[X_1]] \cong [[X_2]] \ \Leftrightarrow\ X_1 \simeq X_2$.

4 Feature-Cluster Algebra

A cluster in a CCTree can uniquely be identified with the set of elements respecting a set of features. To construct a semiring which also contains the clusters, we firstly propose two semirings on the set of features and the set of elements, respectively. The proofs of some theorems are provided in Appendix.

(I) Semiring of Features. Let a *set of disjoint sorts*, denoted as $\mathcal{A}$, is given, where the *carrier set* of each sort $A_i \in \mathcal{A}$ is denoted by $\mathcal{V}_{A_i}$. In our context, we call the given set of sorts as the set of *attributes*, and we call the *union* of sorts, denoted as $\mathbb{V} = \bigcup_{A_i \in \mathcal{A}} \mathcal{V}_{A_i}$ as the set of *values* or *features.* For example, we may consider the set of attributes as $\mathcal{A} = \{color, size\}$, where the carrier set of each attribute can be considered as $\mathcal{V}_{color} = \{red, blue\}$ and $\mathcal{V}_{size} = \{small, large\}$. In this case, we have $\mathbb{V} = \{red, blue, small, large\}$.

Definition 1 (Sort). *We define the* sort *function which gets a set of features and returns a set of the associated sorts of received feature as follows:*

$$
\begin{aligned}
&sort : \mathcal{P}(\mathbb{V}) \to \mathcal{P}(\mathbb{V})\\
&sort(\{f\}) = \mathcal{V}_A \qquad\qquad for \quad f \in \mathcal{V}_A\\
&sort(V_1 \cup V_2) = sort(\{V_1\}) \cup sort(\{V_2\})
\end{aligned}
$$

Consider $\mathbb{F} = \mathcal{P}(\mathcal{P}(\mathbb{V}))$ be the power set of the power set of $\mathbb{V}$, whilst we denote $1 = \{\emptyset\}$ and $0 = \emptyset$. We define the operations "+" and "·", as *choice* and *composition* operators on $\mathbb{F}$ as the following:

$$
\begin{aligned}
&\cdot : \mathbb{F} \times \mathbb{F} \to \mathbb{F} && + : \mathbb{F} \times \mathbb{F} \to \mathbb{F}\\
&F_i \cdot F_j = \{X_s \cup Y_t \ : X_s \in F_i\,,\, Y_t \in F_j\} && F_i + F_j = F_i \cup F_j
\end{aligned}
$$

We say that F belongs to $\mathbb{F}$, if it respects one of the following syntax forms $F \ := 0 \,|\, \{\{f\}\} \,|\, F \cdot F \,|\, F + F \,|\, 1$, where $f \in \mathbb{V}$.

Example 1. Some elements of $\mathbb{F}$ on $\mathbb{V} = \{red, blue, small, large\}$ are $F_1 = \{\{red, large\}, \{blue\}\}$, $F_2 = \{\{small\}\}$, $F_1 \cdot F_2 = \{\{red, large, small\}, \{blue, small\}\}$, and $F_1 + F_2 = \{\{red, large\}, \{blue\}, \{small\}\}$.

Definition 2. *Lets consider* $|.|$ *returns the number of elements in a set. Then, we say* $F \in \mathbb{F}$ *belongs to the set* $\mathbb{F}_n$*, if* $|F| = n$*. Under this definition,* $\mathbb{F}_1$*, i.e. the subset of* $\mathbb{F}$*, where each element contains just one dataset of features, is the desired one according to our problem. In this case, for* $F \in \mathbb{F}_1$*, we remove the brackets and separate the features belonging to the same set by multiplication. Hence, we consider* $F \in \mathbb{F}_1$ *if it can be written as one of the syntax forms as:* $0 \,|\, f \,|\, F_1 \cdot F_2 \,|\, 1$*, for* $f \in \mathbb{V}$*.*

It is noticeable when two elements of $\mathbb{F}_1$ are added or multiplied, they follow the same properties following the main semiring defined on $\mathbb{F}$.

Example 2. We simplify the elements of Example 1 according to Definition 2, as $F_1 = \{\{red, large\}, \{blue\}\} = \{\{red, large\}\} + \{\{blue\}\} = red \cdot large + blue$, $F_2 = \{\{small\}\} = small$, $F_1 \cdot F_2 = \{\{red, large, small\}, \{blue, small\}\} = \{\{red, large, small\}\} + \{\{blue, small\}\} = red \cdot large \cdot small + blue \cdot small$, $F_1 + F_2 = \{\{red, large\}, \{blue\}, \{small\}\} = \{\{red, large\}\} + \{\{blue\}\} + \{\{small\}\} = red \cdot large + blue + small$.

(II) Semiring of Elements. Let us consider that the set of the sorts, or the set of attributes $\mathcal{A}$ with an order among attributes, is given. Suppose $|A| = k$, and without loss of generality $A_1, A_2, \ldots, A_k$ are the ordered sorts which range over $\mathcal{A}$. We say s belongs to *the set of elements* $\mathbb{S}$, if $s \in \mathcal{V}_{A_1} \times \mathcal{V}_{A_2} \times \ldots \times \mathcal{V}_{A_k} \times \mathbb{N}$, where $\mathbb{N}$ is the set of *natural* numbers. Hence, $\mathbb{S} \subseteq \mathcal{V}_{A_1} \times \mathcal{V}_{A_2} \times \ldots \times \mathcal{V}_{A_k} \times \mathbb{N}$, i.e. $s \in \mathbb{S}$ can be written as $s = (x_1, x_2, \cdots, x_k, n)$, where $x_i \in \mathcal{V}_{A_i}$ for $1 \le i \le k$, and $n \in \mathbb{N}$ represents the *ID* of an element. For the sake of simplicity, we may use the alternative representation $x_i \in A_i$ instead of $x_i \in \mathcal{V}_{A_i}$. We formally define two operations "+" and "·" as *union* and *intersection* of elements of $\mathcal{P}(\mathbb{S})$ (the power set of $\mathbb{S}$) as follows:

$$\begin{aligned} \cdot : \mathcal{P}(\mathbb{S}) \times \mathcal{P}(\mathbb{S}) \to \mathcal{P}(\mathbb{S}) \qquad\qquad & + : \mathcal{P}(\mathbb{S}) \times \mathcal{P}(\mathbb{S}) \to \mathcal{P}(\mathbb{S}) \\ S_i \cdot S_j = S_i \cap S_j \qquad\qquad & S_i + S_j = S_i \cup S_j \end{aligned}$$

Formally, we say S belongs to the *power set of the elements* $\mathcal{P}(\mathbb{S})$, if it respects one of the form as $S := \emptyset \,|\, S' \,|\, S + S \,|\, S \cdot S \,|\, \mathbb{S}$, where $S' \subseteq \mathbb{S}$. The quintuple $(\mathcal{P}(\mathbb{S}), +, \cdot, \emptyset, \mathbb{S})$ is an idempotent commutative semiring.

Note: It should be noted that the operations "+" and "·" are overloaded to the kind of elements that they are applied on. This means that if the operation "+" is used between two sets of elements, it refers to the addition operation in semiring of elements, and when the operation "+" is applied between two sets of features, it refers to the addition operation in semiring of features. The same property satisfies for multiplication operation "·".

(III) Semiring of Terms. In what follows, we construct the semiring of terms with the use of previous semrirings, which will be used to abstract the tree structure. In the rest of the chapter, we use the same notions and symbols introduced in previous sections. Recalling that a cluster in CCTree can uniquely be identified by a set of elements respecting a set of features, we define the *satisfaction* relation to formally express the concept of cluster.

Definition 3 (Satisfaction Relation $\diamond$). *Recalling that when the elements of $\mathbb{F}$ contain just one dataset of features we remove the brackets (Definition 2), we define* satisfaction *relation, denoted with $\diamond$, as the following:*

$$\begin{aligned}
&\diamond : \mathbb{F} \times \mathcal{P}(\mathbb{S}) \rightarrow \mathcal{P}(\mathbb{S})\\
&\diamond(f, \{(x_1, x_2, \cdots, x_k)\}) = \{(x_1, x_2, \cdots, x_k)\} \quad if \quad \exists i,\ 1 \leq i \leq k,\ s.t\, x_i = f\\
&\diamond(f, \{(x_1, x_2, \cdots, x_k)\}) = \emptyset \quad if \quad \nexists i,\ 1 \leq i \leq k,\ s.t\ x_i = f\\
&\diamond(f, S_1 \cup S_2) \ = \ \diamond(f, S_1) \ \cup \ \diamond(f, S_2)\\
&\diamond(F_1 \cdot F_2, S) \ = \ \diamond(F_1, S) \ \cap \ \diamond(F_2, S)
\end{aligned}$$

In the case that $\diamond(F, S) \neq \emptyset$, we say that S satisfies *F, and we apply the alternative representation $F \diamond S$ instead of $\diamond(F, S)$.*
We define that the multiplication "$\cdot$" and "$+$" over $\diamond$ to respect the following properties:

$$(F_1 \diamond S_1) \cdot (F_2 \diamond S_2) = (F_1 \cdot F_2) \diamond S_2 \qquad if \quad S_1 \cdot S_2 = S_2 \tag{1}$$

$$(F_1 \diamond S_1) + (F_2 \diamond S_2) = (F_1 + F_2) \diamond S_2 \qquad if \quad S_1 + S_2 = S_2 \tag{2}$$

where $S_1 \cdot S_2 = S_2$ means $S_2 \subseteq S_1$, and $S_1 + S_2 = S_2$ means $S_1 \subseteq S_2$. In the case neither set is a subset of the other, the multiplication and addition return the received elements unchanged. It should be noted that "$\cdot$" and "$+$" are overloaded to their own definition for the semiring of features and the semiring of elements when they are applied between two sets of features and two sets of elements, respectively. In our context, the property 1 is applied to address the concept of division of a cluster to new smaller clusters, where each new cluster satisfies the features of the main cluster, plus more restricted features. Moreover, the property 2 is used to get the simpler form of clusters according to Definition 2.

Proposition 1. *For $F_1, F_2 \in \mathbb{F}$ and $S \in \mathcal{P}(\mathbb{S})$, the symbol "$\diamond$" satisfies the following properties with respect to "$+$" and "$\cdot$":*

$$\begin{aligned}
(F_1 \cdot F_2) \diamond S &= (F_1 \diamond S) \cdot (F_2 \diamond S)\\
(F_1 + F_2) \diamond S &= (F_1 \diamond S) + (F_2 \diamond S)
\end{aligned}$$

Proof. The proof is straightforward from the properties 1 and 2, since we have $S \cdot S = S$ and $S + S = S$. The above equations express how we can transform the different forms of $F \in \mathbb{F}$ to the form of $F \in \mathbb{F}_1$.

Example 3. The following equation shows the transformation of relations of Proposition 1 to a set of features as $F \in \mathbb{F}_1$ as Definition 2.

$$\{\{f_1, f_2\}, \{f_3\}\} \diamond S = \{\{f_1, f_2\}\} \diamond S + \{\{f_3\}\} \diamond S = f_1 \cdot f_2 \diamond S + f_3 \diamond S.$$

The form of $F \in \mathbb{F}_1$ is a particular desired representation of the set of features which will be used in our context. Hence, we attribute a specific name to it.

Definition 4 (Feature-Cluster (Family) Term). *The set of* feature-cluster family terms *on $\mathbb{V}$ and $\mathbb{S}$ denoted as $\mathbb{FC}_{\mathbb{V},\mathbb{S}}$ (or simply $\mathbb{FC}$) is the smallest set containing elements satisfying the following conditions:*

$$\begin{array}{llll} if & S \subseteq \mathbb{S} & then & S \in \mathbb{FC} \\ if & F \in \mathbb{F}_1,\, S \subseteq \mathbb{S} & then & F \diamond S \in \mathbb{FC} \\ if & \tau_1 \in \mathbb{FC}, \tau_2 \in \mathbb{FC} & then & \tau_1 + \tau_2 \in \mathbb{FC} \end{array}$$

In this case, we call S and $F \diamond S$ a "feature-cluster term" *and the addition of one or more feature-cluster terms is called* "feature-cluster family term". *We may simply use "$\mathbb{FC}$-term" to refer to a feature-cluster (family) term.*
We define the block *function, which receives a feature-cluster family term and returns the set of its blocks. Formally, block* : $\mathbb{FC} \to \mathcal{P}(\mathbb{FC})$*, such that:*
$block(S) = \{S\}, \quad block(F \diamond S) = \{F \diamond S\}, \quad block(\tau_1 + \tau_2) = block(\tau_1) \cup block(\tau_2)$

In the case that no feature specifies S directly, it is called an *atomic* term. The set of all atomic terms is denoted as $\mathscr{A}$.

Definition 5 ($\mathbb{FC}$-Term Comparison). *We say two $\mathbb{FC}$-terms τ_1 and τ_2 are* equal*, denoted by $\tau_1 \equiv \tau_2$, if for different representations of $\mathbb{FC}$-terms, it satisfies the following relations:*

$$\begin{array}{rl} S_1 \equiv S_2 & \Leftrightarrow S_1 = S_2 \\ F_1 \diamond S_1 \equiv F_2 \diamond S_2 & \Leftrightarrow S_1 = S_2\,,\, F_1 = F_2 \\ \tau \equiv \tau' & \Leftrightarrow\ block(\tau) = block(\tau'). \end{array}$$

Definition 6 (Term). *We call τ a* term*, if it respects one of the syntax forms as "$\tau := S \mid F \diamond S \mid \tau + \tau \mid \tau \cdot \tau$", where S and F come from $\mathcal{P}(\mathbb{S})$ and $\mathbb{F}$, respectively. Furthermore, we consider the operations "+" and "·" be commutative and associative among terms, whilst "·" left and right distributes over "+".* The set of terms *on $\mathcal{P}(\mathbb{S})$ and $\mathbb{F}$ is shown with $\mathbb{C}_{\mathcal{P}(\mathbb{S}),\mathbb{F}}$, or abbreviated as $\mathbb{C}$ where it is known from the context.*

Example 4. Some examples of terms on $\mathbb{V} = \{red, blue, small, large\}$ and dataset S, can be considered as "$red \cdot small \diamond S$", "$red \cdot small \diamond S + blue \diamond S'$", "$(red \cdot small \diamond S) \cdot (blue \diamond S')$", "$\{\{red, large\}, \{blue\}\} \diamond S$".

Theorem 1. *Two identity elements of $\mathbb{C}$ with respect to "+" and "·" are $0 \diamond \emptyset$ and $1 \diamond \mathbb{S}$, respectively.*

Theorem 2. *The quantiple $(\mathbb{C}, \text{“}+\text{”}, \text{“}\cdot\text{”}, 0 \diamond \emptyset, 1 \diamond \mathbb{S})$ is an idempotent commutative semiring.*

Proof. The proof is straightforward from the semrirng definition , semiring of features, semiring of elements, and the properties mentioned in 1 and 2.

Definition 7 (Feature-Cluster Algebra). *The semiring $(\mathbb{C}, \text{“}+\text{”}, \text{“}\cdot\text{”}, 0 \diamond \emptyset, 1 \diamond \mathbb{S})$ is called a* feature-cluster algebra.

It is noticeable that in present work our terms in the following sections, mostly, belong to the set of feature-cluster family terms $\mathbb{FC} \subseteq \mathbb{C}$. This means that as the elements of the semiring $\mathbb{C}$, they follow the same operation and properties among the elements of the proposed feature-cluster algebra.

5 Feature-Cluster (Family) Term Abstraction

In this section, we explain how tree structure and feature-cluster family term can be transformed to each other. To this end, we first present the "*meaning*" relation to transform a feature-cluster family term to a labeled graph structure. Afterwards, we present a function to get a feature-cluster family term from a labeled tree structure. Then, we prove in a theorem that if two labeled trees are equivalent, they return equal terms. However, we show that the two equal feature-cluster family terms do not necessarily return two equivalent graph structures. We prove that under the condition of considering a fixed order among the features, the latter requirement will also be respected. In the provided examples, attributes $Color = \{r(ed), b(lue)\}$, $Size = \{s(mall), l(arge)\}$, and $Shape = \{c(ircle), t(riangle)\}$ are used to describe the terms. To avoid the confusion of different representations of an $\mathbb{FC}$-term, in what follows we present the definitions of *factorized* and *non factorized* terms.

Definition 8 (Factorized Term). *We define the* factorization *rewriting rule through an attribute $A \in \mathcal{A}$, denoted as $\xrightarrow{A}$, from an $\mathbb{FC}$-term to its factorized form as "$f \cdot \tau_1 + f \cdot \tau_2 \xrightarrow{A} f \cdot (\tau_1 + \tau_2)$" for $f \in A$.*

We denote the normal form of applying the factorization rewriting rule on term τ through attribute A as $\tau \downarrow_A$, and the set of factorized forms of the terms of $\mathbb{FC}$ is denoted by $\mathbb{FC} \downarrow$. A term after factorization is called a factorized term.

Definition 9 (Defactorization). *We define the* defactorized *rewriting rule on an $\mathbb{FC}$-term as "$f \cdot (\tau_1 + \tau_2) \rightarrow_d f \cdot \tau_1 + f \cdot \tau_2$".*

A normal term resulted from defactorized rewriting rule is called a non factorized term. *A non factorized form of the term τ is denoted as $\tau \uparrow$. The set of non factorized terms of $\mathbb{FC}$ are denoted by $\mathbb{FC} \uparrow$.*

Example 5. For factorization we have $(r{\cdot}s{\diamond}S{+}r{\cdot}c{\diamond}S{+}b{\cdot}s{\diamond}S) \downarrow_{color}= r{\cdot}(s{\diamond}S{+}c{\diamond}S){+}b{\cdot}s{\diamond}S$ and for defactorizatio $r{\cdot}(s{\diamond}S{+}c{\diamond}S){+}b{\cdot}s{\diamond}S \rightarrow_d r{\cdot}s{\diamond}S{+}r{\cdot}c{\diamond}S{+}b{\cdot}s{\diamond}S$.

From Feature-Cluster Family Term to Forest Structure. Applying the same notions presented in previous sections, in what follows we define a relation to get a forest structure from $\mathbb{FC}$-terms.

Definition 10 (Meaning Relation). *Considering $\mathcal{G}_{\mathbb{V},\mathbb{FC}}$ as the set of all possible forest structures on the set of edge labels $\mathbb{V}$ and nodes $\mathbb{FC}$, the* meaning *relation, denoted as $[[.]]$, receives a feature-cluster family term and returns a triple as following:*

$$\begin{aligned}
&[[.]] : \mathbb{FC} \uparrow \rightarrow \mathcal{G}_{\mathbb{V},\mathbb{FC}}\\
&[[S]] = (\emptyset, \{S\}, \emptyset)\\
&[[f \diamond S]] = (\{f\}, \{f \diamond S, S\}, \{(S, f, f \diamond S)\})\\
&[[f \cdot F \diamond S]] = (\{f\}, \{f \cdot F \diamond S\}, \{(F \diamond S, f, f \cdot F \diamond S)\}) + [[F \diamond S]]\\
&[[\tau_1 + \tau_2]] = [[\tau_1]] + [[\tau_2]]
\end{aligned}$$

Example 6. In what follows, we show how a feature-cluster family term is transformed to its equivalent tree structure according to above rules:

$$[[r \diamond S + b \cdot l \diamond S + b \cdot s \diamond S]] = (\{b,\ r,\ l,\ s\}, \{S,\ r \diamond S,\ b \diamond S,\ b.l \diamond S,\ b.s \diamond S\}, \\ \{(S,\ r,\ r \diamond S),\ (b \diamond S,\ l,\ b \cdot l \diamond S),\ (S,\ b,\ b \diamond S),\ (b \diamond S,\ s,\ b \cdot s \diamond S)\})$$

From Tree Structure to a Feature-Cluster Family Term. We define the function *root*, denoted as $r : \mathcal{T}_{\mathbb{V},\mathbb{FC}} \rightarrow Q$, which gets a tree and returns the root of the tree, as $r(T) = \{s \mid \cup_{s_i \in Q} \{(s_i, f, s)\} = \emptyset\}$, where $\mathcal{T}_{\mathbb{V},\mathbb{FC}}$ is the set of rooted trees on $\mathbb{V}$ and $\mathbb{FC}$. Moreover, we define the set of edge labels of the children of $r(T)$ as $\delta(T) = \{f \mid \exists\, s' \in Q \ s.t.\ (r(T), f, s') \in \omega\}$. Furthermore, in a tree T, the *descendant tree* directly after edge f, as the *derivative tree* of T following edge f, is denoted by $\partial_f(T)$.

With the use of above notations, we define the Ψ function which gets a tree structure T, and returns the features as $\Psi(T) = \sum_{f \in \delta(T)} f \cdot \Psi(\partial_f(T))$, where $\Psi(T) = 1$ when $\delta(T) = 1$, and $f \cdot 1$ is denoted as f.

We define the *transform* function, denoted by ψ, which gets a set of k labeled trees (forest) and returns an $\mathbb{FC}$-term as follows:

$$\psi : \mathcal{G}_{\mathbb{V},\mathbb{FC}} \rightarrow \mathbb{FC}$$
$$\psi(\emptyset) = 0 \quad , \quad \psi(T_1 \cup T_2) = \Psi(T_1) \diamond r(T_1) + \Psi(T_2) \diamond r(T_2)$$

Example 7. Suppose the following tree $M = (\{f_1, f_2\}, \{s, s_1, s_2\}, \{(s, f_1, s_1), (s, f_2, s_2)\})$ is given. Then, the only state to which there is no transition (the root) is the node s. Hence, we have: $\Psi(M) = f_1 \cdot \Psi(\emptyset, \{s_1\}, \emptyset) + f_2 \cdot \Psi(\emptyset, \{s_2\}, \emptyset) = f_1 \cdot 1 + f_2 \cdot 1 = f_1 + f_2$. Hence, the resulting term is equal to $\psi(M) = \Psi(M) \diamond s$.

Definition 11. *A term resulting from a CCTree structure, or equivalently transformable to a tree structure representing a CCTree, is called a* CCTree term.

Example 8. Suppose the CCTree of Figure 1 is given. The tree structure of this CCTree can be written as $(\{red, blue, small, large\}, \{S, S_r, S_b, S_{b \cdot s}, S_{b \cdot l}\}, \{(S, red, S_r), (S, blue, S_b), (S_b, small, S_{b.s}), (S_b, large, S_{b.l})\})$. Hence, the CCTree term resulting from this CCTree equals to :

$$red \diamond S + blue \cdot small \diamond S + blue \cdot large \diamond S$$

Theorem 3. *The* meaning *relation* $[[.]]$ adequately abstracts *a graph structure resulting from a feature-cluster (family) term on* $\mathbb{V}$ *and the same fixed dataset of elements* $S \subseteq \mathbb{S}$. *This means that for two non factorized* $\mathbb{FC}$*-terms* τ *and* τ' *we have* $[[\tau]] \approx [[\tau']] \ \Rightarrow\ \tau \equiv \tau'$.

Intuitively, the above relation expresses that if two forest structures resulting from two $\mathbb{FC}$-terms are equal, by certain the original terms were equal as well. In other words, if $\tau \not\equiv \tau'$ then we can conclude that $[[\tau]] \not\approx [[\tau']]$. However, the following example contradicts the satisfiability of the relation of Theorem 3 from right to left.

Example 9. The two following feature-cluster family terms are equivalent in terms of term comparison (Definition 5), i.e. we have:

$$f_1 \cdot f_2 \diamond S + f_1 \cdot f_3 \diamond S \quad \equiv \quad f_2 \cdot f_1 \diamond S + f_3 \cdot f_1 \diamond S$$

but their equivalent tree representation are not equivalent, since we have:

$$[[f_1 \cdot f_2 \diamond S + f_1 \cdot f_3 \diamond S]] = (\{f_1, f_2, f_3\}, \{S, f_1 \diamond S, f_1 \cdot f_2 \diamond S, f_1 \cdot f_3 \diamond S\},$$
$$\{(f_1 \diamond S, f_2, f_1.f_2 \diamond S), (f_1 \diamond S, f_3, f_1.f_3 \diamond S), (S, f_1, f_1 \diamond S)\})$$
$$[[f_2 \cdot f_1 \diamond S + f_3 \cdot f_1 \diamond S]] = (\{f_1, f_2, f_3\}, \{S, f_2 \diamond S, f_3 \diamond S, f_2.f_1 \diamond S, f_3.f_2 \diamond S\},$$
$$\{(f_2 \diamond S, f_1, f_2.f_1 \diamond S), (S, f_2, f_2 \diamond S), (f_1 \diamond S, f_3, f_1.f_3 \diamond S), (S, f_1, f_1 \diamond S)\})$$

where the first one contains five nodes, whilst the second one contains four nodes.

This example shows that commutativity of "$\cdot$" is not an appropriate property for *full abstraction.* In what follows, we will show that the reverse of this relation is satisfied if an order of features is identified on the set of features, which solves the problem of multiplication ("$\cdot$") commutativity.

Definition 12 (Ordered Features). *We say that the set of features* $\mathbb{V}$ *is an* "ordered set of features" *if there is an order relation* "$<$" *on* $\mathbb{V}$*, such that* $(\mathbb{V}, <)$ *is a total ordered set. This means that for any* $f_1, f_2 \in \mathbb{V}$ *we either have* $f_1 < f_2$ *or* $f_2 < f_1$*. We say* $F_1 \in \mathbb{F}_1$ *is* exactly equal to $F_2 \in \mathbb{F}_1$*, denoted by* $F_1 \cong F_2$*, if considering the order of features in multiplication representation, they are equal.*

Definition 13 (Order Rewriting Rule). *Let an ordered set of features* $(\mathbb{V}, <)$ *be given. We say an* $\mathbb{FC}$*-term is an* ordered $\mathbb{FC}$-term *on* $(\mathbb{V}, <)$*, if it is the normal form of applying the following rewriting rule:*

$$f_1 \cdot f_2 \diamond S \rightarrow_O f_2 \cdot f_1 \diamond S \qquad if \qquad f_1 < f_2 \quad \forall\, f_1, f_2 \in \mathbb{V}$$

Moreover, we define a rewriting rule which orders the features of an $\mathbb{FC}$*-term based on an attribute* $A \in \mathscr{A}$ *as* $f_2 \cdot f_1 \diamond S \xrightarrow{A}_O f_1 \cdot f_2 \diamond S$ *if* $f_1 \in A$*.*
We represent the normal for of a term τ *applying above rewriting rule, based on attribute* A*, as* $\tau \Downarrow_A$*.*

Example 10. Suppose that the set of features $V_1 = \{red, blue, small, large\}$ is given. Without loss of generality, fixing a strict order "$<$" among them as "$red < blue < small < large$" results in having $(V_1, <)$ as a total ordered set. The following terms show how ordered $\mathbb{FC}$-terms on $\mathbb{V}_1$ are obtained by applying the order rewriting rule:

$$red \cdot small \diamond S + blue \cdot large \diamond S \rightarrow small \cdot red \diamond S + large \cdot blue \diamond S$$

Moreover "$red \cdot small \ncong small \cdot red$", whilst "$red \cdot small \cong red \cdot small$".

Definition 14 (Ordered $\mathbb{FC}$-term Comparison). *We say two ordered* $\mathbb{FC}$*-terms on* $(\mathbb{V}, <)$ *are* exactly equal*, denoted by* $\sim$*, as the smallest relation for which the terms respect one of the following relations:*

$$\begin{array}{llcr}
if & S_1 = S_2 & then & S_1 \sim S_2 \\
if & S_1 = S_2 \wedge F_1 \cong F_2 & then & F_1 \diamond S_1 \sim F_2 \diamond S_2 \\
if & \forall \tau_i \in block(\tau) \;\; \exists \tau_j \in block(\tau') \;\; s.t \;\; \tau_i \sim \tau_j, & & \\
 & \forall \tau_j \in block(\tau') \;\exists \tau_i \in block(\tau) \;\; s.t \;\; \tau_j \sim \tau_i & then & \tau \sim \tau'
\end{array}$$

Theorem 4. *Let* $(\mathbb{V}, <)$ *be a total ordered set of features and* $S \subseteq \mathbb{S}$*. The* meaning *relation* $[[.]]$ abstracts *the forest (tree) structure resulted from the ordered non factorized* $\mathbb{FC}$*-terms on* $\mathbb{V}$ *and* S*. This means considering* τ *and* τ' *be two arbitrary ordered non factorized* $\mathbb{FC}$*-terms on* $(\mathbb{V}, <)$ *and* $S \subseteq \mathbb{S}$*, we have:*

$$\tau \sim \tau' \;\Rightarrow\; [[\tau]] \approx [[\tau']]$$

Theorem 5 (Main Theorem). *Let the ordered set of features* $(\mathbb{V},<)$*, the set of elements* $S \subseteq \mathbb{S}$ *are given. The* meaning *function* $[[.]]$ fully abstracts *the ordered feature-cluster family terms on* $(\mathbb{V},<)$ *and* S*. This means that for two arbitrary ordered feature-cluster family terms* τ *and* τ' *on* $\mathbb{V}$ *and* S*, we have:*

$$[[\tau]] \approx [[\tau']] \Leftrightarrow \tau \sim \tau'$$

Proof. The proof is straightforward from the proofs of Theorems 3 and 4.

6 Relations on Feature-Cluster Algebra

In this section, we define several relations on feature-cluster algebra and discuss the properties of the proposed relations.

Definition 15 (Attribute Division). Attribute division *($\mathscr{D}_{\mathcal{A}}$) is a function from* $\mathcal{A} \times \mathbb{FC}$ *to* $\{True, False\}$*, which gets an attribute and a non factorized* $\mathbb{FC}$*-term as input; it returns "True" or "False" as follows:*

$$\begin{array}{lll}
\mathscr{D}_{\mathcal{A}} : \mathcal{A} \times \mathbb{FC} \uparrow\rightarrow \{True, False\} & & \\
\mathscr{D}_{\mathcal{A}}(A, S) = False & & \\
\mathscr{D}_{\mathcal{A}}(A, f \diamond S) = True & if & f \in A \\
\mathscr{D}_{\mathcal{A}}(A, f \diamond S) = False & if & f \notin A \\
\multicolumn{3}{l}{\mathscr{D}_{\mathcal{A}}(A, f \cdot F \diamond S) = \mathscr{D}_{\mathcal{A}}(A, f \diamond S) \vee \mathscr{D}_{\mathcal{A}}(A, F \diamond S)} \\
\multicolumn{3}{l}{\mathscr{D}_{\mathcal{A}}(A, \tau_1 + \tau_2) = \mathscr{D}_{\mathcal{A}}(A, \tau_1) \wedge \mathscr{D}_{\mathcal{A}}(A, \tau_2)}
\end{array}$$

The concept of attribute division is used to order the attributes presented in a term, which will be discussed later.

Example 11. In the following we show how attribute division performs:
$\mathscr{D}_{\mathcal{A}}(color, r \cdot s \diamond S + r \cdot c \diamond S + b \cdot s \diamond S) =$
$\mathscr{D}_{\mathcal{A}}(color, r \cdot s \diamond S) \wedge \mathscr{D}_{\mathcal{A}}(color, r \cdot c \diamond S) \wedge \mathscr{D}_{\mathcal{A}}(color, b \cdot s \diamond S) = True$

Definition 16 (Initial). *We define the* initial *(δ) function from* $\mathcal{P}(\mathbb{FC}\uparrow)$ *to* $\mathcal{P}(\mathbb{F})$*, i.e.* $\delta : \mathcal{P}(\mathbb{FC}\uparrow) \rightarrow \mathcal{P}(\mathbb{F})$*, which gets a set of ordered non factorized terms on* $(\mathbb{V},<)$ *and returns a set of the first features of each term as follows:*

$$\delta(\emptyset) = \{0\}, \qquad \delta(\{S\}) = \{1\}, \qquad \delta(\{f \cdot F \diamond S\}) = \{f\}$$
$$\delta(\{\tau_1 + \tau_2\}) = \delta(\{\tau_1\}) \cup \delta(\{\tau_2\}), \qquad \delta(\{\tau_1, \tau_2\}) = \delta(\{\tau_1\}) \cup \delta(\{\tau_2\})$$

with the property $\delta(\{X, Y\}) = \delta(X) \cup \delta(Y)$*, where* $X, Y \in \mathcal{P}(\mathbb{FC}\uparrow)$*.*
In the case that the input set contains just one term, we remove the brackets, i.e. $\delta(\{\tau\}) = \delta(\tau)$ *when* $|\{\tau\}| = 1$*. Moreover, when the output set also contains just one element, for the sake of simplicity we remove the brackets, i.e.* $\delta(X) = \{f\} = f$ *for* $X \in \mathcal{P}(\mathbb{FC}\uparrow)$*.*

Definition 17 (Derivative). *We define the* derivative*, denoted by ∂, as a function which gets an ordered non factorized* $\mathbb{FC}$*-term on* $(\mathbb{V},<)$ *and returns the term (set of terms) by cutting off the first features as follows:*

$$\begin{array}{ll}
\partial : \mathbb{FC}\uparrow\rightarrow \mathcal{P}(\mathbb{FC}) & \\
\partial(S) = \emptyset, & \partial(f \diamond S) = \{S\}
\end{array}$$

$$\partial(f \cdot F \diamond S) = \{F \diamond S\}, \qquad \partial(\tau_1 + \tau_2) = \partial(\tau_1) \cup \partial(\tau_2)$$

Note that the functions initial *(δ) and* derivative *(∂) are overloaded to the input, depending to the input that if it is a tree or a term.*

Definition 18 (Order of Attributes). *We say attribute B is smaller or equal to attribute A on the non factorized term $\tau \in \mathbb{FC} \uparrow$, denoted as $B \preceq_\tau A$, if the number of blocks of τ that B divides, is less than (equal to) the number of blocks that A divides. Formally, $B \preceq_\tau A$ implies that:*

$$|\{\tau_i \in block(\tau) \mid \mathscr{D}_{\mathcal{A}}(B, \tau_i) = True\}| \leq |\{\tau_i \in block(\tau) \mid \mathscr{D}_{\mathcal{A}}(A, \tau_i) = True\}|$$

Given a set of attributes $\mathcal{A}$ and a term τ, the set $(\mathcal{A}, \preceq_\tau)$ is a lattice. We denote the upper bound *of this set as $\sqcap_{\mathcal{A},\tau}$. This mean we have $\forall\, A \in \mathcal{A} \Rightarrow A \preceq_\tau \sqcap_{\mathcal{A},\tau}$.*

Example 12. In the following we show how the order of attributes of a term is identified. Suppose that the term $\tau = r \cdot s \diamond S + r \cdot c \diamond S + b \cdot s \diamond S$ is given. We have $block(\tau) = \{r \cdot s \diamond S, r \cdot c \diamond S, b \cdot s \diamond S\}$. Consequently,

$$\begin{aligned} &|\{\tau_i \in block(\tau) \mid \mathscr{D}_{\mathcal{A}}(shape, \tau_i) = True\}| = 1 \\ \leq\ &|\{\tau_i \in block(\tau) \mid \mathscr{D}_{\mathcal{A}}(size, \tau_i) = True\}| = 2 \\ \leq\ &|\{\tau_i \in block(\tau) \mid \mathscr{D}_{\mathcal{A}}(color, \tau_i) = True\}| = 3 \end{aligned}$$

which means that we have $shape \preceq_\tau size \preceq_\tau color$.

Recalling that not having the predefined order among features creates a problem in full abstraction of terms. To this end, here we propose a way to order the set of features which is appropriate to our problem. First of all, given a feature-cluster family term τ, we find the order of attributes according to definition 18, whilst if for two arbitrary attributes A and A', we have $A = A'$, without loss of generality, we choose a strict order among them, say $A \prec A'$. Then in each attribute we arbitrarily order the features. It is important that the features of smaller attribute be always smaller than the features of greater attribute. For example, if $size \prec color$, we consider the order of features as $small < large < blue < red$, whilst all the features of *color* are greater than all the features of *size*.

Definition 19 (Ordered Unification). Ordered unification *($\mathcal{F}$) is a partial function from $\mathcal{P}(\mathcal{A}) \times \mathbb{FC} \uparrow$ to $\mathbb{FC} \downarrow$, which gets a set of attributes and a non factorized term; it returns the normal form of applying rewriting rule $\xrightarrow{A}_O$ introduced in Definition 13, iteratively, based on the order of attributes on received term as follows:*

$$\begin{aligned} &\mathcal{F}(\emptyset, \tau \uparrow) = \tau \\ &\mathcal{F}(\{A\}, \tau \uparrow) = \tau \Downarrow_A \\ &\mathcal{F}(\mathcal{A}, \tau) = \mathcal{F}(\sqcap_{\mathcal{A},\tau}, \mathcal{F}(\mathcal{A} - \{\sqcap_{\mathcal{A},\tau}\}, \tau \uparrow)) \end{aligned}$$

The normal form of ordered unification is called a unified term. *By $\mathcal{F}^*(\tau)$ we mean that $\mathcal{F}$ is performed iteratively on the set of ordered attributes on τ to get the unified term.*

Example 13. To find the unified form of $\tau_1 = r \cdot s \diamond S + r \cdot c \diamond S + b \cdot s \diamond S$, we have "$\mathcal{F}^*(\tau_1) = \mathcal{F}(\{shape, color, size\}, \tau_1 \uparrow) = \mathcal{F}(color, \mathcal{F}(size, \mathcal{F}(shape, \tau_1))) = r \cdot s \diamond S + r \cdot c \diamond + b \cdot s \diamond S$".

Definition 20 (Component relation). *Given two ordered non factorized* $\mathbb{FC}$*-terms* τ_1 *and* τ_2 *on* $(\mathbb{V}, <)$*, we define the* component *relation, denoted by* $\sim_1$*, as the first level comparison of the terms as* $\tau_1 \sim_1 \tau_2 \;\Leftrightarrow\; \delta(\tau_1) = \delta(\tau_2)$.

Proposition 2. *The component relation is an equivalence relation on the set of ordered non factorized* $\mathbb{FC}$*-terms.*

Definition 21 (Component). *Let consider that the ordered term* $\tau \in \mathbb{FC} \uparrow$ *on* $(\mathbb{V}, <)$ *is given. The equivalence class of* $\tau' \in block(\tau)$ *is called a* component *of* τ*, and it is formally defined as* $[\tau']_\tau = \{\tau_i \in block(\tau) \,|\, \tau' \sim_1 \tau_i\}$*. The set of all components of the term* τ *through the equivalence relation* $\sim_1$*, is denoted by* $block(\tau)/\sim_1$ *or simply* $\tau/\sim_1$*, i.e. we have "*$\tau/\sim_1 = \{[\tau_i]_\tau \mid \tau_i \in block(\tau)\}$*".*

Definition 22 (Component Order). *Let* X *and* Y *be two sets of of ordered non factorized* $\mathbb{FC}$*-terms on* $(\mathbb{V}, <)$*. We say* X *is smaller than* Y*, denoted as* $X < Y$*, if* $\forall f' \in \delta(X)$ *and* $\forall f'' \in \delta(Y)$ *we have* $f' < f''$*. Specifically, let* τ *be an ordered non factorized* $\mathbb{FC}$*-term on* $(\mathbb{V}, <)$*. We order the components of* τ *according to the order of features in* $\mathbb{V}$ *as what follows:*

$$[\tau']_\tau < [\tau'']_\tau \quad \Leftrightarrow \quad (\forall f' \in \delta([\tau']) \ , \ \forall f'' \in \delta([\tau'']) \Rightarrow f' < f'')$$

It is noticeable that $|\delta([\tau'])| = |\delta([\tau''])| = 1$*, for all* $\tau', \tau'' \in block(\tau)$*, since the first features of all elements in a component are equal.*

Definition 23 (Well formed term). Well formed function*, denoted as* W*, is a binary function from* $\mathbb{FC} \uparrow$ *to* $\{True, False\}$*, which gets a unified non factorized* $\mathbb{FC}$*-term; it returns* $True$ *if the set of first features of its components is equal to a sort of* $\mathcal{A}$ *to which these features belong; it returns* $False$ *otherwise. Formally:*

$$W(\tau) = \begin{cases} True \ \ if \qquad \delta(\tau/\sim_1) = sort(\delta([\tau_i]_\tau)) \qquad \forall \tau_i \in block(\tau) \\ False \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad otherwise \end{cases}$$

where $\delta(\tau/\sim_1) = sort(\delta([\tau_1]_\tau))$ *means that the the set of the first features of the components of the term* τ *is equal to the attribute that the first feature belongs to. A unified term* τ *is called a* well formed term*, if* $W(\tau) = True$*. An atomic term is a* well formed term.

Example 14. The unified term of Example 13, $\tau = r\cdot s\diamond S + r\cdot c\diamond S + b\cdot s\diamond S$ is a well formed term, since we have "$\delta(\tau/\sim_1) = \delta(\{\{r\cdot s\diamond S\ , \ r\cdot c\diamond S\}, \{b\cdot s\diamond S\}\}) = \{r, b\}$" and "$sort(\delta([r\cdot s\diamond S])) = sort(\{r\cdot s\diamond S\ , \ r\cdot c\diamond S\}) = \{r, b\}$", i.e. $W(\tau) = True$.

It is noticeable that in an ordered CCTree term all first features belong to the same attribute. Hence, in what follows we exploit the concept of well formed term to identify whether a term represents a CCTree term or not. The knowledge of knowing which feature-cluster family term represents a CCTree term, provides us with the opportunity to iteratively use the rules on CCTree terms.

Theorem 6. *A unified term represents a CCTree term, or it is transformable to a CCTree structure, if and only if, it can be written in the form* $\mathcal{F}^*(\tau) = \sum_i f_i \cdot \tau_i$*, such that "* $W(\mathcal{F}^*(\tau)) = True$*", i.e. the unified form of the received term is a well formed term; and the unified form of each* τ_i *is a well formed term as well (*$W(\tau_i) = True$ *) and it also respects the above formula by itself.*

7 Discussion and Future Directions

In this paper, a semiring-based formal method, named Feature-Cluster Algebra, is proposed to abstract the representation of a tree structure representing a categorical clustering algorithm in general, and specifically CCTree. We proved that the proposed approach, under some conditions, fully abstracts the tree structure. Furthermore, we presented a set of functions and relations on feature-cluster algebra, which is used to shape the requirements for a term to represent a CCTree. The abstraction concept proposed in this work, as a novel methodology, establishes an algebraic structure on clusters, which can be considered as a formal technique exploited to get conclusion on a data mining algorithm.

As future work, we plan to apply the proposed technique to abstract a broad category of feature-based data mining algorithms, e.g. additional categorical clustering algorithms, classification, and association rule mining. Moreover, we expect to use the result of abstraction to address the data mining associated issues, e.g. parallel clustering, feature selection.

References

1. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: A literature review. Inf. Syst. **35**(6), 615–636 (2010)
2. Berkhin, P.: A survey of clustering data mining techniques. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping Multidimensional Data, pp. 25–71. Springer, Heidelberg (2006)
3. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 161–168. ACM, NY (2006)
4. Giunchiglia, F., Walsh, T.: A theory of abstraction. Artif. Intell. **57**(2–3), 323–389 (1992)
5. Gross, J.L., Yellen, J.: Graph Theory and Its Applications. Discrete Mathematics and Its Applications, 2nd edn. Chapman & Hall/CRC (2005)
6. Hell, P., Nesetil, J.: Graphs and homomorphisms. Oxford lecture series in mathematics and its applications. Oxford University Press, Oxford, New York (2004)
7. Höfner, P., Khedri, R., Möller, B.: Feature algebra. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006. LNCS, vol. 4085, pp. 300–315. Springer, Heidelberg (2006)
8. Höfner, P., Khédri, R., Möller, B.: An algebra of product families. Software and System Modeling **10**(2), 161–182 (2011)
9. Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: Form: A feature-oriented reuse method with domain-specific reference architectures. Ann. Softw. Eng. **5**, 143–168 (1998)
10. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: Planet: Massively parallel learning of tree ensembles with mapreduce. Proc. VLDB Endow. **2**(2), 1426–1437 (2009)
11. Sheikhalishahi, M., Mejri, M., Tawbi, N.: Clustering spam emails into campaigns. In: Library, S.D. (ed.) 1st International Conference on Information Systems Security and Privacy (2015)
12. Sheikhalishahi, M., Saracino, A., Mejri, M., Tawbi, N., Martinelli, F.: Fast and effective clustering of spam emails based on structural similarity. In: Garcia-Alfaro, J., et al. (eds.) FPS 2015. LNCS, vol. 9482, pp. 195–211. Springer, Heidelberg (2016). doi:10.1007/978-3-319-30303-1_12

Appendix

Proof of Theorem 1

Proof. The proof is resulted from the properties 1 and 2 as the following:

$$(1 \diamond \mathbb{S}) \cdot (F \diamond S) = (1 \cdot F) \diamond S = F \diamond S$$
$$(0 \diamond \emptyset) \cdot (F \diamond S) = (0 \cdot F) \diamond \emptyset = 0 \diamond \emptyset$$
$$(0 \diamond \emptyset) + (F \diamond S) = (0 + F) \diamond S = F \diamond S$$

For the other elements of $\mathbb{C}$, the proof is straightforward from above equations, and the properties 1 and 2.

Proof of Theorem 3

Proof. Suppose that the left hand side of the relation is satisfied. Hence, we have:

$$[[\tau]] \approx [[\tau']] \Rightarrow \Theta(\tau) = \Theta(\tau'),\, \Phi(\tau) = \Phi(\tau'),\, \Delta(\tau) = \Delta(\tau') \tag{3}$$
$$\Rightarrow block(\tau) = block(\tau') \Rightarrow \tau \equiv \tau' \tag{4}$$

where 3 is resulted from the equivalent graph structures of τ and τ', and 4 is satisfied from $\Phi(\tau) = \Phi(\tau')$ and the fact that two main terms were originated from the same dataset.

Proof of Theorem 4

Proof. Suppose the left side of the relation satisfies. This means that for each feature-cluster term $\tau_i \in \tau$ there exists a feature-cluster term $\tau_j \in \tau'$ such that τ_i and τ_j are exactly equal. This property causes that the set of transitions of $[[\tau_i]]$ to be equal to the set of transitions of $[[\tau_j]]$. Hence, we have:

$$\tau \sim \tau' \Rightarrow \forall \tau_i \in block(\tau) \exists \tau_j \in block(\tau')\ s.t\ \tau_i \sim \tau_j (\Rightarrow [[\tau_i]] \approx [[\tau_j]]), \tag{5}$$
$$\forall \tau_j \in block(\tau') \exists \tau_i \in block(\tau)\ s.t\ \tau_j \sim \tau_i (\Rightarrow [[\tau_i]] \approx [[\tau_j]])$$
$$\Rightarrow [[\tau]] \approx [[\tau']] \tag{6}$$

Proof of Proposition 2

Proof. For ordered non factorized $\mathbb{FC}$-terms τ_1, τ_2 and τ_3, we have:

$$\begin{array}{lll} if\ \ \tau_1 \sim_1 \tau_1 & iff & \delta(\tau_1) = \delta(\tau_1) \\ if\ \ \tau_1 \sim_1 \tau_2 \ \wedge\ \tau_2 \sim_1 \tau_1 & iff & \delta(\tau_1) = \delta(\tau_2) \wedge \delta(\tau_2) = \delta(\tau_1) \\ if\ \ \tau_1 \sim_1 \tau_2\,,\ \tau_2 \sim_1 \tau_3\ then\ \tau_1 \sim_1 \tau_3 & iff & \delta(\tau_1) = \delta(\tau_2), \delta(\tau_2) = \delta(\tau_3)\ then\ \delta(\tau_1) = \delta(\tau_3) \end{array}$$

this means that $\sim_1$ is an equivalence relation.

Proof of Theorem 6

Proof. First we show that a unified term obtained from a CCTree structure satisfies the equation 6. In a CCTree, the attribute used for division in the root, has the greatest number of occurrence in non factorized CCTree term (all blocks of CCTree term contain one of the features of this attribute). According to 5, for transforming the tree to a term, the first features of components are specified from $\delta(T) = \{f \,|\, \exists\, s' \in Q \ s.t. \ (s_T, f, s') \in \omega\}$, where in CCTree all belong to the same sort, i.e. we have:

$$\delta(T) = \{f \,|\, \exists\, s' \in Q \ s.t. \ (s_T, f, s') \in \omega\} = sort(\{f\}) \Rightarrow W(\psi(T)) = True$$

we call the tree following a child of the root as a *new tree*. It is noticeable each new tree is a CCTree by itself; hence, it respects 6. By considering the tree following the new tree as new trees themselves, the aforementioned process is iteratively repeated for all new trees, due to the iterative structure of CCTree, i.e. from 5, we have:

$$W(\psi(\partial_f(T))) = True \qquad \forall\ f \in \delta(T)$$

this means that if the input tree structure is a CCTree, then the obtained term respects the above formula.

On the other hand, a unified term that respects equation 6 can be converted to a CCTree structure. To this end, τ_i's are the components of τ separating their first features (f_i's). The set of the first features of components of the term, constitute the transitions of the first division from the root of CCTree, i.e.:

$$\Omega\Big(\sum_{[\tau_i]\in\tau/\sim_1} \delta([\tau_i]) \cdot \sum_{\tau_k\in[\tau_i]} \partial(\tau_k)\Big) = \bigcup_{[\tau_i]\in\tau/\sim_1} \{(S, \delta([\tau_i]), \sum_{\tau_k\in[\tau_i]} \partial(\tau_k))\}$$

where S is the main dataset the term is originated from.
Since the term is well formed, it guarantees that the label of children belong to the same sort, as required by CCTree. Due to the iterative rule for successive components, iteratively the structure of CCTree is constructed. Note that the condition of equivalence of the first features of components to a sort, guarantees that in the process of transforming the term to its equivalent tree structure, all the features of a selected attribute exist.