Mathematics of Digital Hyperspace

Jeremy Kepner[1,2,3], Timothy Davis[4], Vijay Gadepally[1,2], Hayden Jananthan[1,5], Lauren Milechin[6]

[1]MIT Lincoln Laboratory Supercomputing Center, [2]MIT Computer Science & AI Laboratory,
[3]MIT Mathematics Department, [4]Texas A&M, [5]Vanderbilt, [6]MIT Dept. of Earth, Atmospheric, & Planetary Sciences

Abstract—Social media, e-commerce, streaming video, e-mail, cloud documents, web pages, traffic flows, and network packets fill vast digital lakes, rivers, and oceans that we each navigate daily. This digital hyperspace is an amorphous flow of data supported by continuous streams that stretch standard concepts of type and dimension. The unstructured data of digital hyperspace can be elegantly represented, traversed, and transformed via the mathematics of hypergraphs, hypersparse matrices, and associative array algebra. This paper explores a novel mathematical concept, the semilink, that combines pairs of semirings to provide the essential operations for graph analytics, database operations, and machine learning. The GraphBLAS standard currently supports hypergraphs, hypersparse matrices, the mathematics required for semilinks, and seamlessly performs graph, network, and matrix operations. With the addition of key based indices (such as pointers to strings) and semilinks, GraphBLAS can become a richer associative array algebra and be a plug-in replacement for spreadsheets, database tables, and data centric operating systems, enhancing the navigation of unstructured data found in digital hyperspace.

Index Terms—graphs, hypergraphs, hypersparse, networks, polystore, databases, algebra

I. Introduction

Global usage of the Internet is expected to exceed 5 billion people [1]. The volume, velocity, and variety of Internet data continues to expand. Social media, e-commerce, streaming video, e-mail, cloud documents, web pages, traffic flows, and network packets fill vast digital lakes, rivers, and oceans that we each navigate daily [2]. Some of the most common manifestations of these data are in the form of spreadsheets, database tables, matrices, graphs, and networks. The resulting digital hyperspace is an amorphous flow of data supported by continuous streams of these objects that stretch standard concepts of type and dimension.

Fortunately, the unstructured data of digital hyperspace can be elegantly represented, traversed, and transformed via the mathematics of hypergraphs [3]–[6], hypersparse matrices [7]–[9], and associative array algebra [10]–[13]. These mathematics have been implemented in a variety of software libraries, including the GraphBLAS standard [14]–[17] implemented in the C/Matlab/Octave/Python/Julia languages [18]–[21] and

the RedisGraph database [22]; the C-MPI CombBLAS parallel library [23]; and the D4M associative array library in Matlab/Octave/Python/Julia languages [24]–[28] with database bindings to SciDB, Accumulo, and PostGreSQL [29]–[33]. The GraphBLAS standard has further enabled hardware acceleration of these mathematics via multithreading [34], GPUs [35], and special purpose accelerators [36]–[40].

Linearity is a key property of these mathematics utilized by the above implementations to leverage extensive linear systems theory [13]. From a performance perspective, linearity is often manifest through the distributive property

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

enabling the reordering of operations critical for effective parallel computation and distributed database query planning. From a data perspective, linearity provides the additive identity and multiplicative annihilator

$$a \oplus 0 = a \qquad a \otimes 0 = 0$$

eliminating the need to store *0* entries (an essential property for efficient sparse computations). If fact, in this context, the above properties can be used to define *0* for the relevant value set, V, which may, or may not, be the standard arithmetic 0.

Collectively, these mathematical properties are defined by mathematical *semirings* that are directly supported by the aforementioned technologies. The increasing use of semirings for the manipulation of digital data has led to frequent coupling of distinct semirings in graph analysis [8], databases [12], and machine learning computations [41]–[43]. This paper explores some of mathematical properties of coupled semirings, here referred to as *semilinks*, and offers up some potential paths forward to formalizing and applying this novel mathematics as natural extensions to existing technologies, such as the GraphBLAS standard.

The rest of this paper is organized as follows. First, some mathematical preliminaries regarding hypergraphs, hypersparse matrices, and semirings are provided. Associative arrays are then summarized. Next, some general properties of semilinks are explored and some specific possible semilinks are investigated in the context of graph analysis, databases, and machine learning. Finally, some recommendations and conclusions are provided.

II. Mathematical Preliminaries

The navigation of diverse digital data can be enhanced by a number of mathematical concepts which underpin the broader algebra of associative arrays which are briefly described in
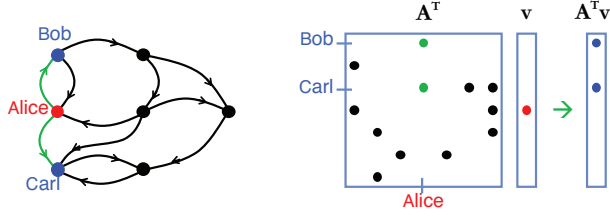
Fig. 1. **Graph Adjacency Array Duality**. Breadth-first-search performed on a graph (left) and an adjacency array (right) illustrates the deep connection between graphs and arrays.
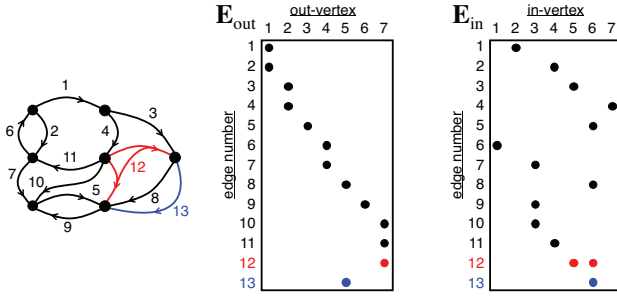

Fig. 2. **Hyper-Multi-Graph Edge Array Duality**. Incidence (or edge) arrays can capture hyper-edges (red) connecting more than two vertices and multi-edges (blue) between the same vertices.
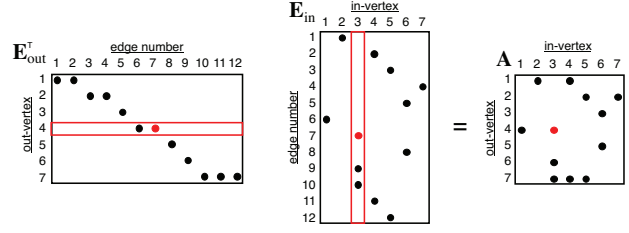

Fig. 3. **Edge Array to Adjacency Array**. Construction of an adjacency array of a graph from its incidence arrays via array multiply. The entry $\mathbf{A}(4,3)$ is obtained by combining the row $\mathbf{E}_{\text{out}}^{\mathsf{T}}(4,k)$ with the column $\mathbf{E}_{\text{in}}(k,3)$ via the array product $\mathbf{A}(4,3) = \bigoplus_{k=1}^{12} \mathbf{E}_{\text{out}}^{\mathsf{T}}(4,k) \otimes \mathbf{E}_{\text{in}}(k,3)$
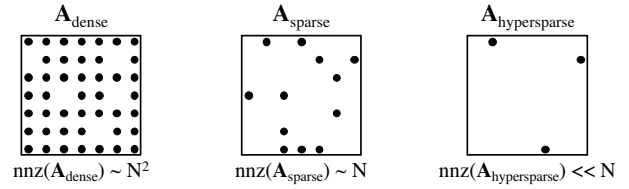

Fig. 4. **Dense, Sparse, and Hypersparse Arrays**. Sparsity concepts for an $N \times N$ array $\mathbf{A}$.

this section (see [8] for a complete description). Perhaps the most important is the graph-matrix duality illustrated in Figure 1 that links the fundamental operation of graphs (breadth-first-search) with the fundamental operation of arrays (array multiplication), where an adjacency array

$$\mathbf{A}(k_1, k_2) \neq 0$$

implies an edge from vertex k_1 to k_2. Hypergraphs extend graphs to provide a natural representation of events that connect diverse entities. Hypersparse arrays extend arrays to allow the efficient storage and operation on data that is growing without bounds. Semirings extend standard arithmetic enabling operations on diverse data to utilize the power of linear systems theory.

A. Hypergraphs

Adjacency arrays are a powerful tool for analyzing directed-weighted-graphs, but are unable to represent the diverse data that is commonly found in streaming events. These streaming events can be described as hyper-multi-weighted-directed-graphs and are best represented as incidence (or edge) arrays (see Figure 2), where

$$\mathbf{E}_{\text{out}}(k, k_1) \neq 0 \qquad \mathbf{E}_{\text{in}}(k, k_2) \neq 0$$

implies that edge k comes out of vertex k_1 and goes into vertex k_2.

The adjacency array and the edge array are strongly coupled via array multiplication (Figure 3)

$$\mathbf{A} = \mathbf{E}_{\text{out}}^{\mathsf{T}} \mathbf{E}_{\text{in}}$$

where the individual values in $\mathbf{A}$ are computed via

$$\mathbf{A}(i, j) = \bigoplus_k \mathbf{E}_{\text{out}}^{\mathsf{T}}(i, k) \otimes \mathbf{E}_{\text{in}}(k, j)$$

The adjacency array represents a projection of edge data and is often an initial step in processing diverse digital data.

B. Hypersparse

As the dimensions of digital data expand the concept of sparsity plays an increasing role. Sensor data, such as images, are well presented by dense arrays where the number on non-zero entries $nnz()$ is small. Physical networks, neural networks, mesh geometries, and other systems where the dimension of the problem is known can often be well represented by sparse arrays where $nnz()$ is on the order of the number of rows or columns in the array. Data with dimensions that are continuously increasing can be captured by hypersparse arrays where $nnz()$ is much smaller than the number of rows or columns (Figure 4).

C. Semirings

Obtaining the advantages of linear systems on diverse data involves extending addition $\oplus$ and multiplication $\otimes$ beyond standard real numbers to include sets and strings. If the set of values is denoted by V, then pairs of operations $\otimes$ and $\oplus$ that obey the distributive property on values from V will generally exhibit the desired properties of a linear system. Formally, the mathematical object with the desired mathematical properties is a semiring denoted $(V, \oplus, \otimes, 0, 1)$, where 0 is the $\oplus$ identity and 1 is the $\otimes$ identity. Some of the common combinations of addition and multiplication

264

Set	$\oplus$	$\otimes$	0	1
$\mathbb{R}$	$+$	$\times$	0	1
$\mathbb{R} \cup -\infty$	max	$+$	$-\infty$	0
$\mathbb{R} \cup +\infty$	min	$+$	$+\infty$	0
$\mathbb{R}_{\geq 0}$	max	$\times$	0	1
$\mathbb{R}_{\geq 0} \cup +\infty$	min	$\times$	$+\infty$	1
$\mathbb{V}$	$\cup$	$\cap$	$\emptyset$	$\mathcal{P}(\mathbb{V})$
$\mathbb{V} \cup -\infty$	max	min	$-\infty$	$+\infty$
$\mathbb{V} \cup +\infty$	min	max	$+\infty$	$-\infty$

operations that have proven valuable are standard arithmetic addition and multiplication $+.\times$, union and intersection $\cup.\cap$ in relational databases [44]–[46], and various tropical algebras that are important in finance [47]–[49] and neural networks [41]–[43]: max.+, min.+, max.×, min.×, max.min, and min.max. Examples of commonly used semirings are shown in Table I. For a guide to the literature on semirings and their applications see [50].

III. ASSOCIATIVE ARRAYS

The full mathematics of associative arrays and the ways they build on the mathematics of the previous section to encompass spreadsheets, database tables, matrices, graphs, networks, and higher dimension tensors are fully described in [10]–[13]. Only the essential mathematical properties of associative arrays are reviewed here. The essence of associative array algebra is three operations: element-wise addition $\oplus$, element-wise multiplication $\otimes$, and array multiplication $\oplus.\otimes$. In brief, the set of associative arrays are defined as a mapping from sets of keys to values

$$\mathbb{A} : K_1 \times K_2 \to \mathbb{V}$$

where K_1 (the set of row keys) and K_2 (the set of column keys) can be any sortable sets, such as the integers, real numbers, or strings. $\mathbb{V}$ is a set of values that forms a semiring $(\mathbb{V}, \oplus, \otimes, 0, 1)$ with addition operation $\oplus$, multiplication operation $\otimes$, additive identity/multiplicative annihilator 0, and multiplicative identity 1. The values can take on many forms, such as numbers, strings, and sets.

Associative array algebra and its specialization in the Graph-BLAS reference two main semirings. The first is the element-wise commutative semiring $(\mathbb{A}, \oplus, \otimes, 0, 1)$ built from the two commutative monoids

$$M_0 = (\mathbb{A}, \oplus, 0) \qquad M_1 = (\mathbb{A}, \otimes, 1)$$

where 0 is the array of all 0 and 1 is the array of all 1. Likewise, the array semiring $(\mathbb{A}, \oplus, \oplus.\otimes, 0, \mathbb{I})$ built from a commutative monoid and non-commutative monoid

$$M_0 = (\mathbb{A}, \oplus, 0) \qquad M_{\mathbb{I}} = (\mathbb{A}, \oplus.\otimes, \mathbb{I})$$

Property	Notation		
Construction	$\mathbf{A} = \mathbb{A}(\mathbf{k}_1, \mathbf{k}_2, \mathbf{v})$		
Extraction	$(\mathbf{k}_1, \mathbf{k}_2, \mathbf{v}) = \mathbf{A}$		
Permutation	$\mathbb{P}(\mathbf{k}_1, \mathbf{k}_2) = \mathbb{A}(\mathbf{k}_1, \mathbf{k}_2, 1)$	$\mathbf{k}_1, \mathbf{k}_2$ unique	
Identity	$\mathbb{I}(\mathbf{k}) = \mathbb{P}(\mathbf{k}, \mathbf{k})$		
Transpose	$\mathbf{A}(k_2, k_1) = \mathbf{A}^{\mathsf{T}}(k_1, k_2)$		
Row keys	$\mathbf{k}_1 = \text{row}(\mathbf{A})$	$\mathbf{k}_1$ unique	
Column keys	$\mathbf{k}_2 = \text{col}(\mathbf{A})$	$\mathbf{k}_2$ unique	
Nonzero count	$\text{nnz}(\mathbf{A})$		
Same sparsity	$|\mathbf{A}|_0 = |\mathbf{B}|_0$		
Element-wise addition	$\mathbf{C} = \mathbf{A} \oplus \mathbf{B}$ $\quad$ $\mathbf{A} \oplus 0 = \mathbf{A}$		
	$\mathbf{C}(k_1, k_2) = \mathbf{A}(k_1, k_2) \oplus \mathbf{B}(k_1, k_2)$		
Element-wise multiplication	$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ $\quad$ $\mathbf{A} \otimes 1 = \mathbf{A}$ $\quad$ $\mathbf{A} \otimes 0 = 0$		
	$\mathbf{C}(k_1, k_2) = \mathbf{A}(k_1, k_2) \otimes \mathbf{B}(k_1, k_2)$		
Array multiplication	$\mathbf{C} = \mathbf{AB} = \mathbf{A} \oplus.\otimes \mathbf{B}$ $\quad$ $\mathbf{A}\mathbb{I} = \mathbf{A}$ $\quad$ $\mathbf{A}0 = 0$		
	$\mathbf{C}(k_1, k_2) = \bigoplus_k \mathbf{A}(k_1, k) \otimes \mathbf{B}(k, k_2)$		
Commutativity	$\mathbf{A} \oplus \mathbf{B} = \mathbf{B} \oplus \mathbf{A}$		
	$\mathbf{A} \otimes \mathbf{B} = \mathbf{B} \otimes \mathbf{A}$		
	$(\mathbf{AB})^{\mathsf{T}} = \mathbf{B}^{\mathsf{T}} \mathbf{A}^{\mathsf{T}}$		
Associativity	$(\mathbf{A} \oplus \mathbf{B}) \oplus \mathbf{C} = \mathbf{A} \oplus (\mathbf{B} \oplus \mathbf{C})$		
	$(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})$		
	$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$		
Distributivity	$\mathbf{A} \otimes (\mathbf{B} \oplus \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \oplus (\mathbf{A} \otimes \mathbf{C})$		
	$\mathbf{A}(\mathbf{B} \oplus \mathbf{C}) = (\mathbf{AB}) \oplus (\mathbf{AC})$		

where $\mathbb{I}(k, k) = 1$ and 0 otherwise. Many of the properties of associative arrays that will be utilized in the semilink discussion are listed in Table II. Of particular practical importance are the large row and column key spaces typically used in associative arrays that practically eliminate the dimensional conformance rules required in matrix operations. As a result, associative arrays are typically added and multiplied with little regard for the true dimensions of their large row and column key spaces. What is more important to producing non-trivial results that are not all 0 is some overlap in the non-zero row and column keys of the constituent associative arrays.

IV. SEMIRINGS TO SEMILINKS

The overlap between three monoids and two semirings commonly used in associative arrays suggests investigating them as a potentially new mathematical concept referred to here as a *semilink*

$$(\mathbb{A}, \oplus, \otimes, \oplus.\otimes, 0, 1, \mathbb{I})$$

Among the standard (albeit somewhat rare) algebraic structures admitting three binary operations are residuated lattices [51], Poisson algebras [52], exponential fields [53], and quasi-groups [54]. The closest in flavor to our semilink that of a composition ring, though even when working over a ring or field the semilink above does not satisfy the identities required to be a composition ring. In addition to being closed under any combination of operations $\oplus$, $\otimes$, and $\oplus.\otimes$ on associative arrays, such a combination of monoids/semirings would seem to have several properties. As part of semirings the pairs of operations $(\oplus, \otimes)$ and $(\oplus, \oplus.\otimes)$ retain their properties within

each pair such as the distributive property and the additive identity is the multiplicative annihilator. Important questions with regards to a semilink are what properties might exist among the pair of operations $(\otimes, \oplus.\otimes)$ and their respective identities $\mathbb{1}$ and $\mathbb{I}$.

It is readily observable that the identities $\mathbb{1}$ and $\mathbb{I}$ preserve their properties with respect to their corresponding operations. For example

$$\mathbb{1} \otimes \mathbb{I} = \mathbb{I} \otimes \mathbb{1} = \mathbb{I} \qquad \mathbb{1}\oplus.\otimes\mathbb{I} = \mathbb{I}\oplus.\otimes\mathbb{1} = \mathbb{1}$$

$\mathbb{I}$ behaves like an identity under $\otimes$ if the array matches the sparsity structure of $\mathbb{I}$. If $|\mathbf{A}|_0 = \mathbb{I}$, then

$$\mathbf{A} \otimes \mathbb{I} = \mathbb{I} \otimes \mathbf{A} = \mathbf{A}$$

where $|\ |_0$ is the element-wise zero-norm that maps all non-zero elements to 1. More generally, if the sparsity pattern of $\mathbf{A}$ is a permutation $|\mathbf{A}|_0 = \mathbb{P}$, then

$$\mathbf{A} \otimes \mathbb{P} = \mathbb{P} \otimes \mathbf{A} = \mathbf{A}$$

In contrast, $\mathbb{1}$ with $\oplus.\otimes$ projects an array onto its rows or columns

$$\mathbf{C} = \mathbf{A}\oplus.\otimes\mathbb{1} \implies \mathbf{C}(k_1, :) = \bigoplus_{k_2} \mathbf{A}(k_1, k_2)$$

$$\mathbf{C} = \mathbb{1}\oplus.\otimes\mathbf{A} \implies \mathbf{C}(:, k_2) = \bigoplus_{k_1} \mathbf{A}(k_1, k_2)$$

Interestingly, under certain conditions, $\oplus.\otimes$ distributes over $\otimes$. Specifically, if $\mathbf{A}$ has the sparsity pattern of a permutation

$$|\mathbf{A}|_0 = |\mathbf{A}_1|_0 = |\mathbf{A}_2|_0 = \mathbb{P}$$

and $\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2$, then

$$\mathbf{A}\oplus.\otimes(\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A}_1\oplus.\otimes\mathbf{B}) \otimes (\mathbf{A}_2\oplus.\otimes\mathbf{C})$$

Similarly, a hybrid associativity does hold in the trivial case. If $\mathbf{A} = \mathbb{1}$ or $\mathbf{C} = \mathbb{I}$, then

$$\mathbf{A} \otimes (\mathbf{B}\oplus.\otimes\mathbf{C}) = (\mathbf{A} \otimes \mathbf{B})\oplus.\otimes\mathbf{C}$$

In a related result, if the non-zero entries of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ do not have sufficient overlap, then the result will be $\mathbb{0}$. Using the row() and col() functions defined in Table II, if

$$\begin{aligned}
\text{row}(\mathbf{A}) \cap \text{row}(\mathbf{B}) &= \emptyset & \text{or} \\
\text{col}(\mathbf{A}) \cap \text{col}(\mathbf{C}) &= \emptyset & \text{or} \\
\text{col}(\mathbf{B}) \cap \text{row}(\mathbf{C}) &= \emptyset
\end{aligned}$$

then

$$\mathbf{A} \otimes (\mathbf{B}\oplus.\otimes\mathbf{C}) = \mathbb{0}$$

Likewise, if

$$\begin{aligned}
\text{row}(\mathbf{A}) \cap \text{row}(\mathbf{B}) &= \emptyset & \text{or} \\
\text{col}(\mathbf{A}) \cap \text{col}(\mathbf{B}) &= \emptyset & \text{or} \\
\text{col}(\mathbf{A}) \cap \text{row}(\mathbf{C}) &= \emptyset & \text{or} \\
\text{col}(\mathbf{B}) \cap \text{row}(\mathbf{C}) &= \emptyset
\end{aligned}$$

then

$$(\mathbf{A} \otimes \mathbf{B})\oplus.\otimes\mathbf{C} = \mathbb{0}$$

which implies that if

$$\begin{aligned}
\text{row}(\mathbf{A}) \cap \text{row}(\mathbf{B}) &= \emptyset & \text{or} \\
\text{col}(\mathbf{B}) \cap \text{row}(\mathbf{C}) &= \emptyset
\end{aligned}$$

then

$$\mathbf{A} \otimes (\mathbf{B}\oplus.\otimes\mathbf{C}) = (\mathbf{A} \otimes \mathbf{B})\oplus.\otimes\mathbf{C} = \mathbb{0}$$

V. EXAMPLES

An important motivation for exploring the semilink concept is their common occurrence in practical applications. In this section several semilinks are explored in the context of graphs, databases, and neural networks.

A. Graph Analytics

The general semilink

$$(\mathbb{A}, \oplus, \otimes, \oplus.\otimes, \mathbb{0}, \mathbb{1}, \mathbb{I})$$

covers a number of important operations in graph analysis. Figure 1 illustrates the duality between the fundamental operation of graphs (breadth-first-search) and the fundamental operation of arrays (array multiplication) $\oplus.\otimes$. Figure 5 shows how element-wise addition $\oplus$ and element-wise multiplication $\otimes$ correspond to graph union and graph intersection, which are also important graph operations. In these graph operations, the essence of the calculation is topological and is determined by the presence of non-zero values in the result and not the exact value itself. Thus, the core topological aspects of graph breadth-first-search, graph union, and graph intersection operations hold for any semiring on the values of the corresponding associative array, including all the semirings listed in Table I.

B. Database Operations

Many database table operations can be mapped onto well-defined mathematical operations with known mathematical properties (see Figure 6). For example, relational (or SQL) databases [55]–[57] are described by relational algebra [44]–[46] that corresponds to the union-intersection semiring $\cup.\cap$ [58]. Triple-store databases (NoSQL) [59]–[63] and analytic databases (NewSQL) [32], [64]–[68] follow similar mathematics [12]. The table operations of these databases are further encompassed by associative array algebra, which brings the beneficial properties of array mathematics and sparse linear systems theory, such as closure, commutativity, associativity, and distributivity [13]. These mathematical properties provide strong correctness and linearity guarantees that are independent of scale and particularly helpful when trying to reason about massively parallel systems.

The full mathematics of associative arrays and the ways they encompass relational algebra are described in the afore-mentioned references [12], [13], [58]. In brief, an associative array $\mathbf{A}$ is defined as a mapping from sets of keys to values. The row keys are equivalent to the sequence ID in a relational database table. The column keys are equivalent to the column
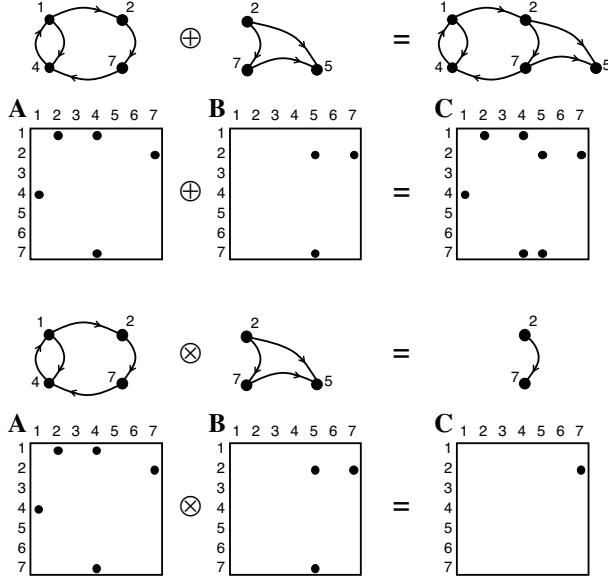
266

Fig. 5. **Graph Union and Intersection**. (top) Element-wise addition $\oplus$ of associative arrays corresponds to graph union. (bottom) Element-wise multiplication $\otimes$ of associative arrays corresponds to graph intersection.

names or record fields in a database table. Intersection $\cap$ distributing over union $\cup$ is essential to database query planning and parallel query execution over partioned/sharded database tables [69]–[75].

Perhaps the most canonical function in a relational database is the SQL select statement that returns the columns $\mathbf{k}$ of rows in a table $\mathbf{A}$ that satisfy a specific condition, such as the value in column $\mathbf{k}(i)$ is v

$$\text{select } \mathbf{k}(1), ..., \mathbf{k}(n) \text{ from } \mathbf{A} \text{ where } \mathbf{k}(i) = v$$

In terms of the associative array notation listed in Table II, the above select can be concisely written as

$$\mathbf{A}(\text{row}(\mathbf{A}(\mathbf{k}(i), :) = v), \mathbf{k})$$

For many databases, the relevant semilink is

$$(\mathbb{A}, \cup, \cap, \cup.\cap, \emptyset, \mathbb{1}, \mathbb{I})$$

where each entry in $\mathbb{1}$ is $\mathcal{P}(\mathbb{V})$ and $\mathbb{I}(k, k) = \mathcal{P}(\mathbb{V})$ and $\emptyset$ otherwise. The associative array version of the select statement can be written in terms of this semilink as

$$|((\mathbf{A} \cup.\cap \mathbb{I}(\mathbf{k}(i)) \cap v) \cup.\cap \mathbb{1}|_0 \cap \mathbf{A}$$

The term $\mathbf{A} \cup.\cap \mathbb{I}(\mathbf{k}(i))$ selects column $\mathbf{k}(i)$ from $\mathbf{A}$. The next operation $\cap v$ selects the entries corresponding to v. A mask of all the columns in these rows is constructed by $\cup.\cap \mathbb{1}$, whose values are converted to $\mathcal{P}(\mathbb{V})$ with the zero norm $| |_0$. Applying the mask with $\cap \mathbf{A}$ selects the corresponding rows.

C. Deep Neural Networks

Machine learning has been the foundation of artificial intelligence since its inception [76]–[83]. Standard machine learning applications include speech recognition [78], computer vision [79], and even board games [80], [84].

Drawing inspiration from biological neurons to implement machine learning was the topic of the first paper presented at the first machine learning conference in 1955 [76], [77] (see Figure 7). It was recognized very early on in the field that direct computational training of neural networks was computationally unfeasible with the computers that were available at that time [82]. The many-fold improvement in neural network computation and theory has made it possible to create neural networks capable of better-than-human performance in a variety of domains [85]–[88]. The production of validated data sets [89]–[91] and the power of graphic processing units (GPUs) [92]–[95] have allowed the effective training of deep neural networks (DNNs) with 100,000s of input features, N, and 100s of layers, L, that are capable of choosing from among 100,000s categories, M (see Figure 8).

The primary mathematical operation performed by a DNN network is the inference, or forward propagation, step. Inference is executed repeatedly during training to determine both the weight matrix $\mathbf{W}_\ell$ and the bias vectors $\mathbf{b}_\ell$ of the DNN. The inference computation shown in Figure 8 is given by

$$\mathbf{y}_{\ell+1} = h(\mathbf{y}_\ell \mathbf{W}_\ell + \mathbf{b}_\ell)$$

where $h()$ is a nonlinear function applied to each element of the vector. The Sparse DNN Challenge uses the standard graph community convention whereby $\mathbf{W}(i, j) \neq 0$ implies a connection between neuron i and neuron j. In this convention $\mathbf{y}_\ell$ are row vectors and left array multiplication is used to progress through the network. A commonly used function is the rectified linear unit (ReLU) given by

$$h(\mathbf{y}) = \max(\mathbf{y}, 0)$$

which sets values less than 0 to 0 and leaves other values unchanged. When training a DNN, or performing inference on many different inputs, it is usually necessary to compute multiple $\mathbf{y}_\ell$ vectors at once in a batch that can be denoted as the array $\mathbf{Y}_\ell$. In array form, the inference step becomes

$$\mathbf{Y}_{\ell+1} = h(\mathbf{Y}_\ell \mathbf{W}_\ell + \mathbf{B}_\ell)$$

where $\mathbf{B}_\ell$ is a replication of $\mathbf{b}_\ell$ along columns given by

$$\mathbf{B}_\ell = \mathbf{b}_\ell |\mathbf{Y}_\ell \mathbf{1}|_0$$

and $\mathbf{1}$ is a column array of 1's, and $| \ |_0$ is the zero norm.

If $h()$ were a linear function, then the above equation could be solved exactly and the computation could be greatly simplified. However, current evidence suggests that the non-linearity of $h()$ is required for a DNN to be effective. Interestingly, the inference computation can be rewritten as a linear function over two different semirings

$$\mathbf{y}_{k+1} = \mathbf{y}_k \mathbf{W}_k \otimes \mathbf{b}_k \oplus 0$$
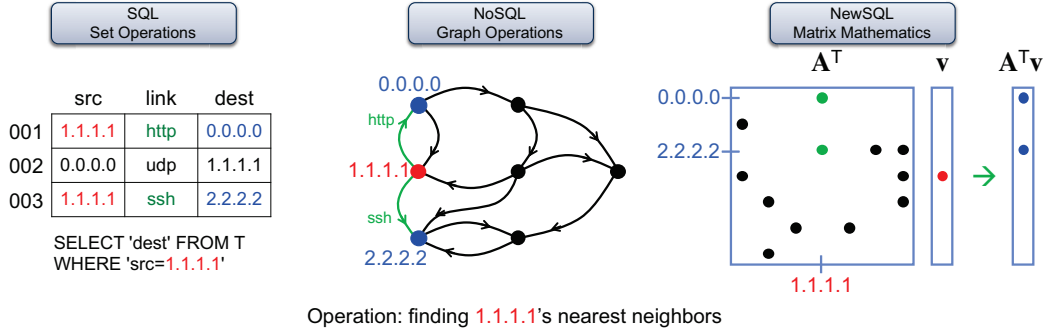
267

Fig. 6. Associative arrays combine the properties of databases, graphs, and matrices and provide common mathematics that span SQL, NoSQL, and NewSQL databases, and are ideal for analyzing networks. The diagram shows the graph operation of finding the neighbors of 1.1.1.1 in each representation.
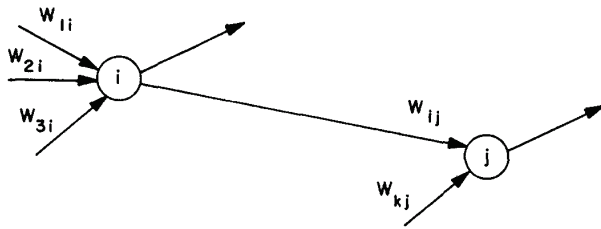

Fig. 7. Typical network elements i and j showing connection weights w (reproduced from [77])
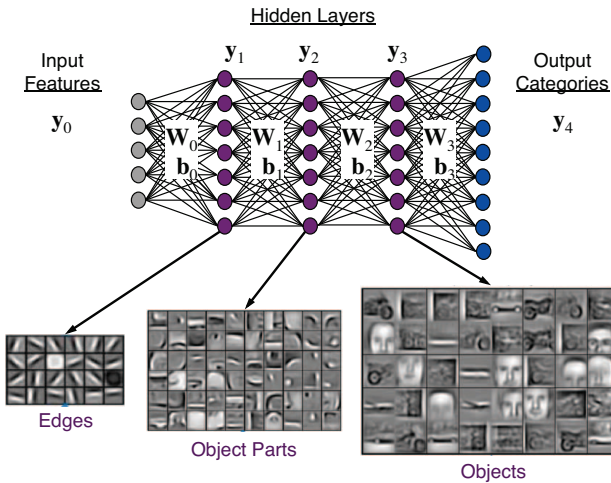

Fig. 8. Four layer ($L = 4$) deep neural network architecture for categorizing images. The input features $\mathbf{y}_0$ of an image are passed through a series of network layers $\mathbf{W}_{\ell=0,1,2,3}$, with bias terms $\mathbf{b}_{\ell=0,1,2,3}$, that produce scores for categories $\mathbf{y}_{L=4}$. (Figure adapted from [96])

or in array form

$$\mathbf{Y}_{k+1} = \mathbf{Y}_k \mathbf{W}_k \otimes \mathbf{B}_k \oplus 0$$

where the $\oplus = \max$ and $\otimes = +$. Thus, $\mathbf{y}_k \mathbf{W}_k$ and $\mathbf{Y}_k \mathbf{W}_k$ are computed over the standard arithmetic $+.\times$ semiring

$$S_1 = (\mathbb{R}, +, \times, 0, 1)$$

while the $\oplus$ and $\otimes$ operation are performed over the $\max.+$ semiring

$$S_2 = (\{-\infty \cup \mathbb{R}\}, \max, +, -\infty, 0)$$

Thus, the ReLU DNN can be written as a linear system that oscillates over two semirings S_1 and S_2. S_1 is the most widely used of semirings and performs standard correlation between vectors. S_2 is also a commonly used semiring for selecting optimal paths in graphs. Thus, the inference step of a ReLU DNN can be viewed as combining correlations of inputs to choose optimal paths through the neural network. This DNN semiring pair is is more complex than what is described in by the semilink concept and may require extending the semilink concept to encompass DNNs.

VI. CONCLUSIONS AND FUTURE WORK

The unstructured data of digital hyperspace can be elegantly represented, traversed, and transformed via the mathematics of hypergraphs, hypersparse matrices, and associative array algebra. Within this context this paper has explored a new mathematical concept, the semilink, that combines pairs of semirings to provide the essential operations for graph analytics, database operations, and machine learning. The formal mathematical specification of GraphBLAS includes monoid, semiring, and closure under element-wise addition, element-wise multiplication, and array multiplication and naturally supports linked semirings.

The specification was written from an associative array algebra perspective with intentionally minimal constraints on the internal implementation of the opaque GrB_Matrix data structure. This has allowed the GraphBLAS (in its SuiteSparse implementation) to support a myriad of different data structures: sparse, hypersparse, bitmap, and full. It uses each of them when appropriate, and switches between them automatically, with little or no involvement from the user application. In the future, this will enable distributed-memory and GPU accelerations as well. This flexibility has enabled the GraphBLAS standard to support hypergraphs, hypersparse matrices, and the mathematics required for semilinks, and seamlessly performs graph, network, and matrix operations. With the addition of key based indices (such as pointers to strings) and semilinks,

GraphBLAS can become a richer associative array algebra and be a plug-in replacement for spreadsheets, database tables, and data centric operating systems [97], [98], enhancing the navigation of unstructured data found in digital hyperspace.

From an applied mathematical perspective, the more complex pairing of operations in the DNN context is worth additional exploring. Likewise, in the context of abstract algebra, [99] considers semirings in which the multiplicative identity can be local, so that in any small part of the structure there is a multiplicative identity as far as that part of the structure is concerned. It would be worth exploring this concept in the context of infinite key spaces where identity matrices are a challenge.

ACKNOWLEDGMENTS

REFERENCES

[1] "*Cisco Visual Networking Index: Forecast and Trends, 20182023.*" https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] P. Sawadogo and J. Darmont, "On data lake architectures and metadata management," *Journal of Intelligent Information Systems*, pp. 1–24, 2020.

[3] C. Berge, "Graphs and hypergraphs," 1973.

[4] G. Ghoshal, V. Zlatić, G. Caldarelli, and M. E. Newman, "Random hypergraphs and their applications," *Physical Review E*, vol. 79, no. 6, p. 066118, 2009.

[5] J. N. Mordeson and P. S. Nair, *Fuzzy graphs and fuzzy hypergraphs*, vol. 46. Physica, 2012.

[6] J. Shun, "Practical parallel hypergraph algorithms," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 232–249, 2020.

[7] A. Buluc and J. R. Gilbert, "On the representation and multiplication of hypersparse matrices," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–11, IEEE, 2008.

[8] J. Kepner and J. Gilbert, *Graph algorithms in the language of linear algebra*. SIAM, 2011.

[9] J. Kepner, T. Davis, C. Byun, W. Arcand, D. Bestor, W. Bergeron, V. Gadepally, M. Hubbell, M. Houle, M. Jones, A. Klein, P. Michaleas, L. Milechin, J. Mullen, A. Prout, A. Rosa, S. Samsi, C. Yee, and A. Reuther, "75,000,000,000 streaming inserts/second using hierarchical hypersparse graphblas matrices," *IPDPSW GrAPL*, 2020.

[10] J. V. Kepner, "Multidimensional associative array database," Jan. 14 2014. US Patent 8,631,031.

[11] J. Kepner and V. Gadepally, "Adjacency matrices, incidence matrices, database schemas, and associative arrays," *IPDPS Graph Algorithms Building Blocks*, 2014.

[12] J. Kepner, V. Gadepally, D. Hutchison, H. Jananthan, T. Mattson, S. Samsi, and A. Reuther, "Associative array model of sql, nosql, and newsql databases," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2016.

[13] J. Kepner and H. Jananthan, *Mathematics of Big Data: Spreadsheets, databases, matrices, and graphs*. MIT Press, 2018.

[14] J. Kepner, P. Aaltonen, D. Bader, A. Buluç, F. Franchetti, J. Gilbert, D. Hutchison, M. Kumar, A. Lumsdaine, H. Meyerhenke, S. McMillan, J. Moreira, J. Owens, C. Yang, M. Zalewski, and T. Mattson, "Mathematical foundations of the GraphBLAS," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2016.

[15] A. Buluç, T. Mattson, S. McMillan, J. Moreira, and C. Yang, "Design of the graphblas api for c," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 643–652, IEEE, 2017.

[16] M. Kumar, J. E. Moreira, and P. Pattnaik, "Graphblas: handling performance concerns in large graph analytics," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pp. 260–267, 2018.

[17] T. Mattson, T. A. Davis, M. Kumar, A. Buluc, S. McMillan, J. Moreira, and C. Yang, "Lagraph: A community effort to collect graph algorithms built on top of the graphblas," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 276–284, IEEE, 2019.

[18] T. A. Davis, "Graph algorithms via suitesparse: Graphblas: triangle counting and k-truss," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2018.

[19] J. Chamberlin, M. Zalewski, S. McMillan, and A. Lumsdaine, "Pygb: Graphblas dsl in python with dynamic compilation into efficient c++," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 310–319, IEEE, 2018.

[20] J. E. Moreira, M. Kumar, and W. P. Horn, "Implementing the graphblas c api," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 298–309, IEEE, 2018.

[21] T. A. Davis, "Algorithm 1000: Suitesparse: Graphblas: Graph algorithms in the language of sparse linear algebra," *ACM Transactions on Mathematical Software (TOMS)*, vol. 45, no. 4, pp. 1–25, 2019.

[22] P. Cailliau, T. Davis, V. Gadepally, J. Kepner, R. Lipman, J. Lovitz, and K. Ouaknine, "Redisgraph graphblas enabled graph database," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 285–286, IEEE, 2019.

[23] A. Buluç and J. R. Gilbert, "The combinatorial blas: Design, implementation, and applications," *The International Journal of High Performance Computing Applications*, vol. 25, no. 4, pp. 496–509, 2011.

[24] J. Kepner, W. Arcand, W. Bergeron, C. Byun, M. Hubbell, B. Landon, A. McCabe, P. Michaleas, A. Prout, T. Rosa, *et al.*, "Massive database analysis on the cloud with d4m," *HPEC, Sep*, pp. 21–22, 2011.

[25] J. Kepner, W. Arcand, W. Bergeron, N. Bliss, R. Bond, C. Byun, G. Condon, K. Gregson, M. Hubbell, J. Kurz, A. McCabe, P. Michaleas, A. Prout, A. Reuther, A. Rosa, and C. Yee, "Dynamic distributed dimensional data model (D4M) database and computation system," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 5349–5352, IEEE, 2012.

[26] A. Chen, A. Edelman, J. Kepner, V. Gadepally, and D. Hutchison, "Julia implementation of the dynamic distributed dimensional data model," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, IEEE, 2016.

[27] L. Milechin, V. Gadepally, S. Samsi, J. Kepner, A. Chen, and D. Hutchison, "D4m 3.0: Extended database and language capabilities," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2017.

[28] L. Milechin, V. Gadepally, and J. Kepner, "Database operations in d4m.jl," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–5, IEEE, 2018.

[29] J. Kepner, C. Anderson, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, P. Michaleas, J. Mullen, D. O'Gwynn, A. Prout, A. Reuther, A. Rosa, and C. Yee, "D4M 2.0 schema: A general purpose high performance schema for the Accumulo database," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2013.

[30] J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, *et al.*, "Achieving 100,000,000 database inserts per second using Accumulo and D4M," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2014.

[31] V. Gadepally, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, L. Edwards, M. Hubbell, P. Michaleas, J. Mullen, *et al.*, "D4M: bringing associative arrays to database engines," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2015.

[32] D. Hutchison, J. Kepner, V. Gadepally, and A. Fuchs, "Graphulo implementation of server-side sparse matrix multiply in the Accumulo database," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2015.

[33] S. Samsi, L. Brattain, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, M. Jones, A. Klein, *et al.*, "Benchmarking

scidb data import on hpc systems," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–5, IEEE, 2016.

[34] M. Aznaveh, J. Chen, T. A. Davis, B. Hegyi, S. P. Kolodziej, T. G. Mattson, and G. Szárnyas, "Parallel graphblas with openmp," in *2020 Proceedings of the SIAM Workshop on Combinatorial Scientific Computing*, pp. 138–148, SIAM, 2020.

[35] X. Wang, Z. Lin, C. Yang, and J. D. Owens, "Accelerating dnn inference with graphblas and the gpu," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2019.

[36] W. S. Song, J. Kepner, H. T. Nguyen, J. I. Kramer, V. Gleyzer, J. R. Mann, A. H. Horst, L. L. Retherford, R. A. Bond, N. T. Bliss, *et al.*, "3-d graph processor," in *Workshop on High Performance Embedded Workshop (HPEC)*, MIT Lincoln Laboratory, 2010.

[37] W. S. Song, "Processor for large graph algorithm computations and matrix operations," June 10 2014. US Patent 8,751,556.

[38] W. S. Song, V. Gleyzer, A. Lomakin, and J. Kepner, "Novel graph processor architecture, prototype system, and results," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2016.

[39] Z. Jia, B. Tillman, M. Maggioni, and D. P. Scarpazza, "Dissecting the graphcore ipu architecture via microbenchmarking," *arXiv preprint arXiv:1912.03413*, 2019.

[40] M. James, M. Tom, P. Groeneveld, and V. Kibardin, "Ispd 2020 physical mapping of neural networks on a wafer-scale deep learning accelerator," in *Proceedings of the 2020 International Symposium on Physical Design*, pp. 145–149, 2020.

[41] J. Kepner, M. Kumar, J. Moreira, P. Pattnaik, M. Serrano, and H. Tufo, "Enabling massive deep neural networks with the graphblas," in *High Performance Extreme Computing Conference (HPEC)*, IEEE, 2017.

[42] M. Kumar, W. Horn, J. Kepner, J. Moreira, and P. Pattnaik, "Ibm power9 and cognitive computing," *IBM Journal of Research and Development*, 2018.

[43] T. A. Davis, M. Aznaveh, and S. Kolodziej, "Write quick, run fast: Sparse deep neural network in 20 minutes of development time via suitesparse: Graphblas," in *2019 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2019.

[44] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[45] D. Maier, *The theory of relational databases*, vol. 11. Computer science press Rockville, 1983.

[46] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*, vol. 8. Addison-Wesley Reading, 1995.

[47] P. Klemperer, "The product-mix auction: A new auction design for differentiated goods," *Journal of the European Economic Association*, vol. 8, no. 2-3, pp. 526–536, 2010.

[48] E. Baldwin and P. Klemperer, "Understanding preferences:'demand types', and the existence of equilibrium with indivisibilities," *SSRN*, 2016.

[49] B. A. Mason, "Tropical algebra, graph theory, & foreign exchange arbitrage," 2016.

[50] K. Glazek, *A guide to the literature on semirings and their applications in mathematics and information sciences: with complete bibliography*. Springer Science & Business Media, 2002.

[51] K. Blount and C. Tsinakis, "The structure of residuated lattices," *International Journal of Algebra and Computation*, vol. 13, no. 04, pp. 437–461, 2003.

[52] M. Aguiar, "Pre-poisson algebras," *Letters in Mathematical Physics*, vol. 54, no. 4, pp. 263–277, 2000.

[53] S. Kuhlmann, *Ordered exponential fields*, vol. 12. American Mathematical Soc., 2000.

[54] J. D. Smith, *An introduction to quasigroups and their representations*. CRC Press, 2006.

[55] M. Stonebraker, G. Held, E. Wong, and P. Kreps, "The design and implementation of INGRES," *ACM Transactions on Database Systems (TODS)*, vol. 1, no. 3, pp. 189–222, 1976.

[56] C. J. Date and H. Darwen, *A guide to the SQL Standard: a user's guide to the standard relational language SQL*. Addison-Wesley, 1989.

[57] R. Elmasri and S. Navathe, *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.

[58] H. Jananthan, Z. Zhou, V. Gadepally, D. Hutchison, S. Kim, and J. Kepner, "Polystore mathematics of relational algebra," in *Big Data Workshop on Methods to Manage Heterogeneous Big Data and Polystore Databases*, IEEE, 2017.

[59] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 205–220, 2007.

[60] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[61] L. George, *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*. " O'Reilly Media, Inc.", 2011.

[62] J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, L. Edwards, V. Gadepally, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, A. Rosa, C. Yee, and A. Reuther, "Lustre, hadoop, accumulo," in *2015 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–5, Sep. 2015.

[63] A. Cordova, B. Rinaldi, and M. Wall, *Accumulo: Application Development, Table Design, and Best Practices*. " O'Reilly Media, Inc.", 2015.

[64] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, *et al.*, "C-Store: a column-oriented DBMS," in *Proceedings of the 31st international conference on Very large data bases*, pp. 553–564, VLDB Endowment, 2005.

[65] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. Jones, S. Madden, M. Stonebraker, Y. Zhang, *et al.*, "H-store: a high-performance, distributed main memory transaction processing system," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1496–1499, 2008.

[66] P. Cudré-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik, "A demonstration of SciDB: a science-oriented DBMS," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1534–1537, 2009.

[67] M. Stonebraker and A. Weisberg, "The VoltDB main memory DBMS," *IEEE Data Engineering Bulletin*, vol. 36, no. 2, pp. 21–27, 2013.

[68] V. Gadepally, J. Bolewski, D. Hook, D. Hutchison, B. Miller, and J. Kepner, "Graphulo: Linear algebra graph kernels for nosql databases," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*, pp. 822–830, IEEE, 2015.

[69] G. M. Booth, "Distributed information systems," in *Proceedings of the June 7-10, 1976, national computer conference and exposition*, pp. 789–794, ACM, 1976.

[70] D. E. Shaw, "A relational database machine architecture," in *ACM SIGIR Forum*, vol. 15 #2, pp. 84–95, ACM, 1980.

[71] M. Stonebraker, "The case for shared nothing," *IEEE Database Eng. Bull.*, vol. 9, no. 1, pp. 4–9, 1986.

[72] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The google cluster architecture," *IEEE micro*, vol. 23, no. 2, pp. 22–28, 2003.

[73] C. Curino, E. Jones, Y. Zhang, and S. Madden, "Schism: a workload-driven approach to database replication and partitioning," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 48–57, 2010.

[74] A. Pavlo, C. Curino, and S. Zdonik, "Skew-aware automatic database partitioning in shared-nothing, parallel oltp systems," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 61–72, ACM, 2012.

[75] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.

[76] W. H. Ware, "Introduction to session on learning machines," in *Proceedings of the March 1-3, 1955, western joint computer conference*, pp. 85–85, ACM, 1955.

[77] W. A. Clark and B. G. Farley, "Generalization of pattern recognition in a self-organizing system," in *Proceedings of the March 1-3, 1955, western joint computer conference*, pp. 86–91, ACM, 1955.

[78] O. G. Selfridge, "Pattern recognition and modern computers," in *Proceedings of the March 1-3, 1955, western joint computer conference*, pp. 91–93, ACM, 1955.

[79] G. Dinneen, "Programming pattern recognition," in *Proceedings of the March 1-3, 1955, western joint computer conference*, pp. 94–100, ACM, 1955.

[80] A. Newell, "The chess machine: an example of dealing with a complex task by adaptation," in *Proceedings of the March 1-3, 1955, western joint computer conference*, pp. 101–108, ACM, 1955.

[81] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI magazine*, vol. 27, no. 4, p. 12, 2006.

[82] M. Minsky and O. G. Selfridge, "Learning in random nets," in *Information theory : papers read at a symposium on information theory held at the Royal Institution, London, August 29th to September 2nd*, pp. 335–347, Butterworths, London, 1960.

[83] M. Minsky, "Steps toward artificial intelligence," *Proceedings of the IRE*, vol. 49, no. 1, pp. 8–30, 1961.

[84] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[85] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[86] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[87] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[88] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[89] J. P. Campbell, "Testing with the yoho cd-rom voice verification corpus," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, pp. 341–344, IEEE, 1995.

[90] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[91] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.

[92] M. Campbell, A. J. Hoane, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[93] M. P. McGraw-Herdeg, D. P. Enright, and B. S. Michel, "Benchmarking the nvidia 8800gtx with the cuda development platform," *HPEC 2007 Proceedings*, 2007.

[94] A. Kerr, D. Campbell, and M. Richards, "Gpu performance assessment with the hpec challenge," in *HPEC Workshop 2008*, 2008.

[95] E. A. Epstein, M. I. Schor, B. Iyer, A. Lally, E. W. Brown, and J. Cwiklik, "Making watson fast," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 15–1, 2012.

[96] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, pp. 609–616, ACM, 2009.

[97] J. Kepner, R. Brightwell, A. Edelman, V. Gadepally, H. Jananthan, M. Jones, S. Madden, P. Michaleas, H. Okhravi, K. Pedretti, *et al.*, "Tabularosa: Tabular operating system architecture for massively parallel heterogeneous compute engines," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, IEEE, 2018.

[98] M. Cafarella, D. DeWitt, V. Gadepally, J. Kepner, C. Kozyrakis, T. Kraska, M. Stonebraker, and M. Zaharia, "Dbos: A proposal for a data-centric operating system," *arXiv preprint arXiv:2007.11112*, 2020.

[99] D. Wilding, *Linear algebra over semirings*. The University of Manchester (United Kingdom), 2015.