Article

Temporal Multimodal Data-Processing Algorithms Based on Algebraic System of Aggregates

Andreas Pester [1], Yevgeniya Sulema [2,*], Ivan Dychka [2] and Olga Sulema [2]

[1] Faculty of Informatics and Computer Science, The British University in Egypt, Cairo 11837, Egypt
[2] Department of Computer Systems Software, Igor Sikorsky Kyiv Polytechnic Institute, 03056 Kyiv, Ukraine; olga.sulema@pzks.fpm.kpi.ua (O.S.)
* Correspondence: sulema@pzks.fpm.kpi.ua

Abstract: In many tasks related to an object's observation or real-time monitoring, the gathering of temporal multimodal data is required. Such data sets are semantically connected as they reflect different aspects of the same object. However, data sets of different modalities are usually stored and processed independently. This paper presents an approach based on the application of the Algebraic System of Aggregates (ASA) operations that enable the creation of an object's complex representation, referred to as multi-image (MI). The representation of temporal multimodal data sets as the object's MI yields simple data-processing procedures as it provides a solid semantic connection between data describing different features of the same object, process, or phenomenon. In terms of software development, the MI is a complex data structure used for data processing with ASA operations. This paper provides a detailed presentation of this concept.

Keywords: data aggregates; temporal multimodal data; data-processing algorithms

check for updates

Citation: Pester, A.; Sulema, Y.; Dychka, I.; Sulema, O. Temporal Multimodal Data-Processing Algorithms Based on Algebraic System of Aggregates. *Algorithms* **2023**, *16*, 186. https://doi.org/10.3390/a16040186

Academic Editor: Vangelis Th. Paschos

Received: 20 December 2022
Revised: 13 March 2023
Accepted: 23 March 2023
Published: 29 March 2023

1. Introduction

Humans perceive real-world objects through the multiple senses. Semantic fusion of multi-type (multimodal) information about the object of observation received using multi-channel sensing is a natural process for the human brain. Following this natural principle, many scientific and engineering tasks also require complex semantic descriptions of an object of study based on the fusion of multimodal data received from multiple devices.

In many cases, the object is supervised over the course of time. Such timewise observation can help in understanding the dynamics of the object's behavior and is necessary in many applications. Then, such a timewise complex description requires a collection of multimodal data that are obtained from several sensors measuring certain parameters of the object's behavior, cameras recording the appearance of the object, etc., over time, not necessarily received simultaneously. This brings the problem of correct multimodal data representation based on data synchronization and aggregation.

Another aspect of using multimodality is its employment for object recognition. The latest advancement in this area concerns multimodal machine learning, which involves integrating and modeling information from multiple heterogeneous sources of data [1]. However, this approach reveals several challenges related to heterogeneity of the data.

According to [2], the first fundamental challenge is comprehensive data representation that takes into consideration the complementarity and redundancy of multiple modalities. The second challenge is multimodal data mapping that enables matching data of different modalities. The third challenge is data alignment that consists of the necessity to identify direct relations between elements from different modalities. The fourth challenge is data fusion, which yields united information of different modalities. The fifth challenge is co-learning, which consists of transferring knowledge based on different modalities. As stated in [1], there are even more challenges that include reasoning, generation, and quantification, along with data representation, alignment, and transference. All these and related issues require correct and comprehensive multimodal data representation, which is key to overcoming challenges related to different aspects of multimodal data analysis and processing.

The rest of the paper is organized as follows. Section 2 presents a related work review. Section 3 formulates the requirements of temporal multimodal data processing and explains the need to employ the Algebraic System of Aggregates (ASA) for the formal specification of an object. Section 4 explains the basic concepts of the ASA. Section 5 provides a detailed presentation of the algorithms of operations on aggregates. Section 6 presents the notion of a multi-image of an object and proposes an algorithm for multi-image formation. Section 7 offers a use case and provides the discussion of ASA operations application for multi-image formation. Finally, Section 8 concludes the paper.

2. Related Work

There are a number of papers presenting different approaches and views on the task of data fusion/data aggregation in various contexts.

Jesus et al. [3] formally defined the concept of aggregation, reviewed distributed data aggregation algorithms, and provided taxonomy of the main aggregation techniques. Comparing different techniques in their survey, the authors showed that among main data aggregation classes, the hierarchical-based approach is the cheapest option; the sketch-based approach could be considered fast but not very precise; the averaging technique gives a higher level of precision but with a much slower execution, which might not be very efficient to operate on dynamic networks, although very few approaches would actually be practical when there are dynamic settings, which presents a challenge for further research on improvements in the efficiency of dynamic networks.

Ribeiro et al. [4] proposed an algorithm for intelligent information fusion in uncertain environments, named fuzzy information fusion (FIF), which is based on multi-criteria decision making and computational intelligence. In this study, data fusion was considered a process of aggregating data from multiple sources into a single composite with higher information quality. The authors analyzed various methods of data fusion. While they considered fuzzy Set theory to be of high applicability, it needs application domain knowledge for data representation, which makes this method not fully universal and creates room for further research in this direction.

Oliveira et al. [5] presented an application of the fuzzy information fusion algorithm for the aggregation of various sources of heterogeneous information to generate value-added maps. In their work, they used operators from different classes of operators, such as algebraic, average, and reinforcement, to fuse, e.g., aggregate data. Validation with specific scenarios allowed authors to compare aggregation operators. The multiplicative FIMICA operator was identified as the most consistent operator that gave the best classification outputs for the scenarios used. It shows, however, that for certain use cases, there is a need for specialized operators that yield efficiently aggregated data depending on a specific scenario.

Lahat et al. [6] considered data fusion to be “the analysis of several data sets such that different data sets can interact and inform each other” and studied multimodality as a form of diversity. In the paper, the authors focused their attention on different aspects of multimodality, considering it in the context of various applications. They also outlined the importance of the development of single-set analysis methods for advanced data fusion.

Marinoni et al. [7] considered data fusion in the context of image pre-processing with relation to transfer learning in remote sensing. In their paper, the authors proposed metrics that quantify the maximum information extraction performance to be achieved by multimodal remote-sensing analysis. Empirical outcomes presented in the paper demonstrated how the accuracy performance of a standard classifier applied to a multimodal data set can be improved using the reliability metric. The approach proposed by the authors can be used to improve multimodal remote-sensing information extraction.

Gaonkar et al. [8] provided a study on advancements in multimodal signal processing. In the paper, focus was given to multimodal data representation and information fusion. The authors considered information fusion based on model-agnostic and model-based approaches.

Oliveira et al. [9] focused on the application of data fusion in a decision support system (DSS). The effectiveness of such a system depends on reliable fusion of data coming from multiple sources. The paper proposed high-level architecture of the DSS.

As was stated in the Introduction, one of the promising applications of multimodality is multimodal machine learning (MML). This topic stimulated a number of resent research works.

Liang et al. [10] discussed the foundational principles in multimodal research and, in particular, the principle of heterogeneity and the principle of interconnection. The authors also provided taxonomy of six core challenges in multimodal machine learning (representation, alignment, reasoning, generation, transference, and quantification) and gave their comprehensive analysis. They outlined different aspects of these challenges, and some of them are of particular interest in the context of the research purposes of this paper. Specifically, modality connections and interconnections, as an essential part of multimodal models, can show how modalities are related and how their elements interact. There is, however, a need, which the authors pointed out as one of the future directions, to formally define the core principles of heterogeneity, connections, and interactions. It requires a mathematical framework to be able to capture causal, logical, and temporal connections and interactions, which formulates a mathematical and algorithmic problem to consider.

Guo et al. [11] studied deep multimodal representation learning frameworks, including modality-specific representations, joint representation and coordinated representation, and encoder–decoder framework. They outlined one of the challenges that still exists in the context of machinery comprehension of information from multiple sensory organs, which is the heterogeneity gap in multimodal data. Aiming to narrow this gap, the authors summarized some typical models in deep multimodal representation learning, including probabilistic graphical models, multimodal autoencoders, deep canonical correlation analysis, generative adversarial networks, and attention mechanisms. Their analysis of different learning frameworks showed that one of the main disadvantages of existing frameworks and models is the difficulty of coordinating more than two modalities. It formulates a practical challenge for further research to overcome the issue of multiple modalities.

Baltrusaitis et al. [2] provided an overview of the recent advances in multimodal machine learning and presented a summary of applications enabled by multimodal machine learning. In their work, they introduced taxonomy of multimodal machine learning, which includes five core challenges: representation, translation, fusion, alignment, and co-learning. Some of these challenges have already been studied quite well. In contrast, there are a few more recent ones (e.g., representation and translation), which are now leading the creation of new multimodal algorithms and are still of interest for further research.

Kline et al. [12] provide a summary of multimodal data fusion applications for solving medical diagnostics problems. The paper shows how medical data of different modalities (e.g., text/image, EHR/genomic/time series) recorded and extracted can be used in a specific use case, giving a firm understanding of the practical importance of multimodal data processing.

3. Approach and Requirements for Temporal Multimodal Data Processing

In our research, we consider an approach to temporal multimodal data processing based on the formal specification of the objects under study. For this purpose, we need a theoretical apparatus to provide the logic of presentation and processing of temporal multimodal data at the level of mathematics, algorithms, and software, taking into account such features of this data:

- Multimodality means that the object is determined by a collection of data of a different nature; the logic of presenting and processing data about the object depends on the qualitative and quantitative composition of the data set.

- Temporality means that the elements of data collection are ordered by the time of their receiving; the order of following individual elements of the data sequence affects the result of processing the entire set of data of the object.

The description of the object and the effective processing of temporal multimodal data require specific mathematical abstractions and mechanisms to operate them. The basic requirements for a mathematical apparatus to describe and process data include the possibility of the following:

1. Presenting a data set as a structure of semantically interconnected elements.
2. Considering the sequence of data set elements when performing logical operations on them.
3. Reordering elements of the data set.

Let us consider several candidates for a mathematical concept to meet these requirements. As the data are temporal, the first possible candidate is the theory of time series (TTS) [13,14]. The TTS is one of the branches of mathematical statistics that deals with the analysis of stochastic processes. The TTS enables the analysis of data sequences, as well as the study of trends, predictions, and other similar data processing. However, time series do not allow any logical operations of the data elements. Therefore, the TTS cannot be employed for logical processing of temporal multimodal data.

Two other candidates are the theory of sets (TS) [15] and the theory of multisets (TMS) [16,17]. Both of them provide flexible mechanisms for the logical processing of data sets; however, they do not consider the sequence of elements and, accordingly, do not provide the possibility of ordering the elements in a set of temporal multimodal data.

Thus, neither of the considered theories fully correspond to the needs of temporal multimodal data processing based on the formal specification of the objects under study. However, the abovementioned requirements are satisfied by the Algebraic System of Aggregates (ASA) [18,19]. This mathematical concept enables data processing with consideration of both required data features, namely multimodality and temporality (Table 1).

Table 1. ASA comparison.

Features	TTS	TS	TMS	ASA
Ordering	yes	no	no	yes
Logical operations	no	yes	yes	yes
Fuzziness	partly	partly	partly	yes
Data aggregation	no	no	partly	yes
Temporality	yes	no	no	yes

Let us consider the main provisions and possibilities of the ASA for the presentation and processing of temporal multimodal data describing an object.

4. Basics Notions of ASA

The ASA [18,19] is an algebraic system whose carrier is a non-empty set of objects, which we call aggregates.

Definition 1. *An aggregate A is an ordered finite collection of elements defined by a tuple of sets $\{A\}$ and a tuple of tuples of elements $\langle A\rangle$, and the elements a_i^j of each tuple of elements $\left\langle a_i^j \right\rangle_{i=1}^{n_j} \in \langle A\rangle$ belong to the corresponding set $M_j \in \{A\}$, $j = [1 \ldots N]$, which specifies a one-to-one relationship between the sequence of sets and the sequence of tuples of elements:*

$$A = \left[\!\left[M_j \mid \left\langle a_i^j \right\rangle_{i=1}^{n_j} \right]\!\right]_{j=1}^{N} = [\![\{A\} \mid \langle A\rangle]\!], \tag{1}$$

where $\{A\}$ *is a tuple of sets* M_j*;* $\langle A\rangle$ *is a tuple of tuples of elements* $\left\langle a_i^j \right\rangle_{i=1}^{n_j}$*; and* a_i^j *is a separate element (value) or a composite element (a tuple of homogeneous values or a tuple of heterogeneous values),* $a_i^j \in M_j$*.*

The defining features of the aggregate, which distinguish this mathematical abstraction from others, are the following:

- An aggregate is a complex mathematical object, all of whose components are ordered;
- Elements of the tuples can be individual values or tuples of values, and the tuples of values can consist of both values of the same type and values of different types, while each tuple contains elements belonging to one set.

According to (1), the elements of the first tuple belong to the first set, the elements of the second tuple belong to the second set, etc. The sequence of sets in an aggregate determines how operations on the aggregate are performed. Sets in a tuple of sets can be repeated; this means that the aggregate includes several tuples consisting of elements of the same type. In software code, an aggregate can be defined as a data structure presented in Listing 1.

Listing 1. Interface of a data structure for representing an aggregate.

```
Aggregate structure:
    collection ← ordered collection of tuples
    length ← number of tuples in the aggregate

    Initialize Aggregate
        define length of the aggregate
        create collection of size length
        initialize tuples in collection

    Insertion (entity) ← add an entity, i.e., a set of values from different tuples
    Extraction (entity) ← delete the specific entity
    TupleInsertion (position) ← add a new tuple to the specific position in the aggregate
    TupleExtraction (position) ← delete the specific tuple
    AscendingSorting (primaryTuple) ← sort tuples in the aggregate by the defined primary
        tuple in asceding order
    DecendingSorting (primaryTuple) ← sort tuples in the aggregate by the defined primary
        tuple in descending order
    Singling (primatyTuple) ← remove all non-unique entities by the defined primary tuple
    SetsOrdering (orderOfTuples) ← reorder tuples by the defined order
```

As the order of elements in the aggregate is a crucial point in the ASA, the result of carrying out operations on aggregates depends on the aggregates' compatibility.

Definition 2. *Aggregates* A_1 *and* A_2 *are called compatible* ($A_1 \doteqdot A_2$) *if they have the same length, and the type and sequence of sets in them match, that is, the conditions are met:*

$$\begin{cases} \mid A_1 \mid = \mid A_2 \mid \\ \{A_1\} \equiv \{A_2\} \end{cases}. \tag{2}$$

For example, the aggregates defined by (3) and illustrated by Figure 1 are compatible.

$$\begin{aligned} A_1 &= [\![M_1, M_2, M_3 \mid \left\langle a_1^{1,1}, a_2^{1,1}, a_3^{1,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2} \right\rangle, \left\langle a_1^{1,3}, a_2^{1,3}, a_3^{1,3}, a_4^{1,3} \right\rangle]\!], \\ A_2 &= [\![M_1, M_2, M_3 \mid \left\langle a_1^{2,1}, a_2^{2,1} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2}, a_3^{2,2}, a_4^{2,2}, a_5^{2,2} \right\rangle, \left\langle a_1^{2,3}, a_2^{2,3} \right\rangle]\!]. \end{aligned} \tag{3}$$

In Figure 1, the color of an element represents data modality (elements of set M_1 are blue, elements of set M_2 are brown and elements of set M_3 are green), the first value in the element's number designates the aggregate (A_1 or A_2), and the second value in the element's number is an ordering number of the element in the tuple belonging to a certain set. For example, the blue circle, which contains the numbers **1-1**, represents the element $a_1^{1,1}$ that belongs to the set M_1 from the definition of A_1 and the green circle, which contains the numbers **2-1**, represents the element $a_1^{2,3}$ that belongs to the set M_3 from the definition of A_2.

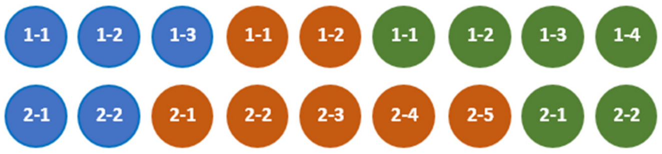


Figure 1. An example of two compatible aggregates.

Definition 3. *Aggregates A_1 and A_2 are called quasi-compatible ($A_1 \doteq A_2$) if the type and sequence order of the sets in them partially coincide, while there is no requirement for the equality of the lengths of these aggregates, i.e., the conditions are fulfilled:*

$$\begin{cases} \{A_1\} \not\equiv \{A_2\} \\ \{A_1\} \cap \{A_2\} \neq \varnothing. \end{cases} \tag{4}$$

For example, the aggregates defined by (5) and illustrated by Figure 2 are quasi-compatible.

$$\begin{aligned} A_1 &= [\![M_1, M_2, M_3^1 \mid \langle a_1^{1,1}, a_2^{1,1}, a_3^{1,1} \rangle, \langle a_1^{1,2}, a_2^{1,2} \rangle, \langle a_1^{1,3}, a_2^{1,3}, a_3^{1,3}, a_4^{1,3} \rangle]\!], \\ A_2 &= [\![M_1, M_2, M_3^2 \mid \langle a_1^{2,1}, a_2^{2,1} \rangle, \langle a_1^{2,2}, a_2^{2,2}, a_3^{2,2}, a_4^{2,2}, a_5^{2,2} \rangle, \langle a_1^{2,3}, a_2^{2,3} \rangle]\!]. \end{aligned} \tag{5}$$

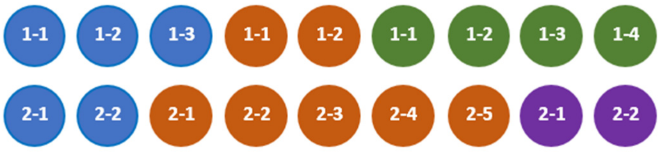


Figure 2. An example of two quasi-compatible aggregates.

Definition 4. *Aggregates A_1 and A_2 are called incompatible ($A_1 \overset{\circ}{=} A_2$), if the type and sequence of the sets in them do not match, that is, the condition is fulfilled:*

$$\{A_1\} \cap \{A_2\} = \varnothing. \tag{6}$$

For example, the aggregates defined by (7) and illustrated by Figure 3 are incompatible.

$$\begin{aligned} A_1 &= [\![M_1^1, M_2^1, M_3^1 \mid \langle a_1^{1,1}, a_2^{1,1}, a_3^{1,1} \rangle, \langle a_1^{1,2}, a_2^{1,2} \rangle, \langle a_1^{1,3}, a_2^{1,3}, a_3^{1,3}, a_4^{1,3} \rangle]\!], \\ A_2 &= [\![M_1^2, M_2^2, M_3^2 \mid \langle a_1^{2,1}, a_2^{2,1} \rangle, \langle a_1^{2,2}, a_2^{2,2}, a_3^{2,2}, a_4^{2,2}, a_5^{2,2} \rangle, \langle a_1^{2,3}, a_2^{2,3} \rangle]\!]. \end{aligned} \tag{7}$$

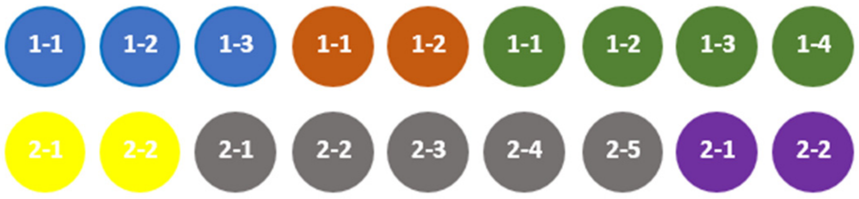


Figure 3. An example of two incompatible aggregates.

A special case of incompatibility is hidden compatibility.

Definition 5. *Aggregates A_1 and A_2 are called hiddenly compatible, $A_1\ (\div)\ A_2$, if both aggregates have the same set of sets, but their ordering is different, i.e., the conditions are fulfilled:*

$$\begin{cases} \{A_1\} \not\equiv \{A_2\} \\ \mid A_1 \mid = \mid A_2 \mid = N \\ \forall M_j \subset \{A_k\}, \end{cases} \quad (8)$$

where $j = [1, \ldots, N]$, $k = [1,\ 2]$.

For example, the aggregates defined by formulas (9) and illustrated by Figure 4 are hiddenly compatible.

$$\begin{aligned} A_1 &= [\![M_1, M_2,\ M_3 \mid \left\langle a_1^{1,1}, a_2^{1,1}, a_3^{1,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2} \right\rangle, \left\langle a_1^{1,3}, a_2^{1,3}, a_3^{1,3}, a_4^{1,3} \right\rangle]\!], \\ A_2 &= [\![M_2,\ M_3,\ M_1 \mid \left\langle a_1^{2,1}, a_2^{2,1} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2},\ a_3^{2,2}, a_4^{2,2}, a_5^{2,2} \right\rangle, \left\langle a_1^{2,3}, a_2^{2,3} \right\rangle]\!]. \end{aligned} \quad (9)$$

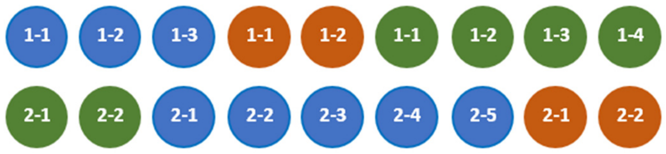


Figure 4. An example of two hiddenly compatible aggregates.

Hiddenly compatible aggregates can be made compatible by applying certain operations to them.

5. Algorithms of Operations on Aggregates

The operations on aggregates in the ASA include logical operations, ordering operations, and arithmetic operations.

5.1. Logical Operations

The logical operations [18] on aggregates are union, intersection, exclusive intersection, difference, and symmetric difference. The result of any logical operation depends on the aggregates' compatibility. For example, the rule for the union operation can be mathematically defined as follows.

The *union* of the aggregates A_1 and A_2 is the aggregate $R_\cup$, which contains elements of the tuples that belong to both aggregates and are ordered in the following way:

1. If $A_1 \div A_2$, then aggregates A_1 and A_2 are defined as

$$A_1 = [\![M_1, M_2, \ldots, M_N \mid \left\langle a_1^{1,1}, a_2^{1,1}, \ldots, a_{n_1^1}^{1,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2}, \ldots, a_{n_2^1}^{1,2} \right\rangle, \ldots, \left\langle a_1^{1,N}, a_2^{1,N}, \ldots, a_{n_N^1}^{1,N} \right\rangle]\!],$$

$$A_2 = [\![M_1, M_2, \ldots, M_N \mid \left\langle a_1^{2,1}, a_2^{2,1}, \ldots, a_{n_1^2}^{2,1} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2}, \ldots, a_{n_2^2}^{2,2} \right\rangle, \ldots, \left\langle a_1^{2,N}, a_2^{2,N}, \ldots, a_{n_N^2}^{2,N} \right\rangle]\!],$$

and elements of i-tuple of the aggregate A_2 are added to the end of i-tuple of the aggregate A_1:

$$\begin{gathered} R_\cup = A_1 \cup A_2 = [\![M_1, M_2, \ldots, M_N \mid \left\langle a_1^{1,1}, a_2^{1,1}, \ldots, a_{n_1^1}^{1,1}, a_1^{2,1}, a_2^{2,1}, \ldots, a_{n_1^2}^{2,1} \right\rangle, \\ \left\langle a_1^{1,2}, a_2^{1,2}, \ldots, a_{n_2^1}^{1,2}, a_1^{2,2}, a_2^{2,2}, \ldots, a_{n_2^2}^{2,2} \right\rangle, \ldots, \left\langle a_1^{1,N}, a_2^{1,N}, \ldots, a_{n_N^1}^{1,N}, a_1^{2,N}, a_2^{2,N}, \ldots, a_{n_N^2}^{2,N} \right\rangle]\!]. \end{gathered} \quad (10)$$

2. If $A_1 \doteq A_2$, then aggregates A_1 and A_2 are defined as

$$A_1 = \llbracket M_1, M_2^1, \ldots, M_k, \ldots, M_{N^1}^1 \mid \left\langle a_1^{1,1}, a_2^{1,1}, \ldots, a_{n_1^1}^{1,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2}, \ldots, a_{n_2^1}^{1,2} \right\rangle, \ldots, \left\langle a_1^{1,k}, a_2^{1,k}, \ldots, a_{n_k^1}^{1,k} \right\rangle, \ldots, \left\langle a_1^{1,N^1}, a_2^{1,N^1}, \ldots, a_{n_{N^1}^1}^{1,N^1} \right\rangle \rrbracket,$$

$$A_2 = \llbracket M_1, M_2^2, \ldots, M_k, \ldots, M_{N^2}^2 \mid \left\langle a_1^{2,1}, a_2^{2,1}, \ldots, a_{n_1^2}^{2,1} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2}, \ldots, a_{n_2^2}^{2,2} \right\rangle, \ldots, \left\langle a_1^{2,k}, a_2^{2,k}, \ldots, a_{n_k^2}^{2,k} \right\rangle, \ldots, \left\langle a_1^{2,N^2}, a_2^{2,N^2}, \ldots, a_{n_{N^2}^2}^{2,N^2} \right\rangle \rrbracket,$$

(a) elements of the i-tuple of the aggregate A_2 are added to the end of the i-tuple of the aggregate A_1, if the elements of these tuples belong to the same i-set;

(b) for all i-tuples whose elements belong to different sets, the tuple of tuples of aggregate A_2 is added to the end of the tuple of tuples of aggregate A_1, and the tuple of sets of aggregate A_2 is added to the end of the tuple of sets of aggregate A_1, with the exception of tuples subject to rule (a), which are excluded from the tuple of tuples:

$$R_U = A_1 \cup A_2 = \llbracket M_1, M_2^1, \ldots, M_k, \ldots, M_{N^1}^1, M_2^2, \ldots, M_{N^2}^2 \mid \left\langle a_1^{1,1}, a_2^{1,1}, \ldots, a_{n_1^1}^{1,1}, a_1^{2,1}, a_2^{2,1}, \ldots, a_{n_1^2}^{2,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2}, \ldots, a_{n_2^1}^{1,2} \right\rangle, \ldots, \left\langle a_1^{1,k}, a_2^{1,k}, \ldots, a_{n_k^1}^{1,k}, a_1^{2,k}, a_2^{2,k}, \ldots, a_{n_k^2}^{2,k} \right\rangle, \ldots, \left\langle a_1^{1,N^1}, a_2^{1,N^1}, \ldots, a^{1}_{n^{1,N^1}_{N^1}} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2}, \ldots, a_{n_2^2}^{2,2} \right\rangle, \ldots, \left\langle a_1^{2,N^2}, a_2^{2,N^2}, \ldots, a_{n_{N^2}^2}^{2,N^2} \right\rangle \rrbracket \tag{11}$$

3. If $A_1 \mathrel{\hat{=}} A_2$, then aggregates A_1 and A_2 are defined as

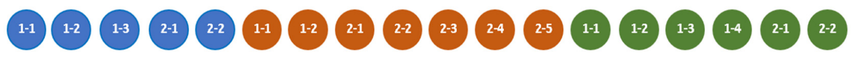
$$A_1 = \llbracket M_1^1, M_2^1, \ldots, M_{N^1}^1 \mid \left\langle a_1^{1,1}, a_2^{1,1}, \ldots, a_{n_1^1}^{1,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2}, \ldots, a_{n_2^1}^{1,2} \right\rangle, \ldots, \left\langle a_1^{1,N^1}, a_2^{1,N^1}, \ldots, a_{n_{N^1}^1}^{1,N^1} \right\rangle \rrbracket,$$

$$A_2 = \llbracket M_1^2, M_2^2, \ldots, M_{N^2}^2 \mid \left\langle a_1^{2,1}, a_2^{2,1}, \ldots, a_{n_1^2}^{2,1} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2}, \ldots, a_{n_2^2}^{2,2} \right\rangle, \ldots, \left\langle a_1^{2,N^2}, a_2^{2,N^2}, \ldots, a_{n_{N^2}^2}^{2,N^2} \right\rangle \rrbracket,$$

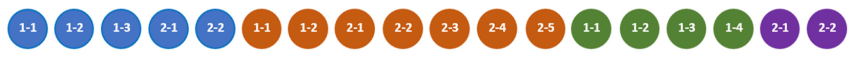
the tuple of tuples of aggregate A_2 is added at to end of the tuple of tuples of the aggregate A_1, and the tuple of sets of aggregate A_2 is added to the end of the tuple of sets of the aggregate A_1:

$$R_\cup = A_1 \cup A_2 = \llbracket M_1^1, M_2^1, \ldots, M_{N^1}^1, M_1^2, M_2^2, \ldots, M_{N^2}^2 \mid \left\langle a_1^{1,1}, a_2^{1,1}, \ldots, a_{n_1^1}^{1,1} \right\rangle, \left\langle a_1^{1,2}, a_2^{1,2}, \ldots, a_{n_2^1}^{1,2} \right\rangle, \ldots, \left\langle a_1^{1,N^1}, a_2^{1,N^1}, \ldots, a_{n_{N^1}^1}^{1,N^1} \right\rangle, \left\langle a_1^{2,1}, a_2^{2,1}, \ldots, a_{n_1^2}^{2,1} \right\rangle, \left\langle a_1^{2,2}, a_2^{2,2}, \ldots, a_{n_2^2}^{2,2} \right\rangle, \ldots, \left\langle a_1^{2,N^2}, a_2^{2,N^2}, \ldots, a_{n_{N^2}^2}^{2,N^2} \right\rangle \rrbracket. \tag{12}$$

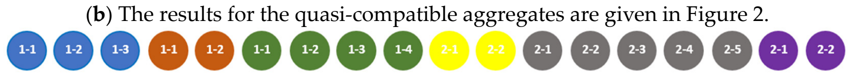
The results of applying the union operation to two aggregates with different compatibility (Figures 1–3) are shown in Figure 5.



(**a**) The results for the compatible aggregates are given in Figure 1.



(**b**) The results for the quasi-compatible aggregates are given in Figure 2.



(**c**) The results for the incompatible aggregates are given in Figure 3.

Figure 5. An example of applying the union operation to two aggregates of different compatibility.

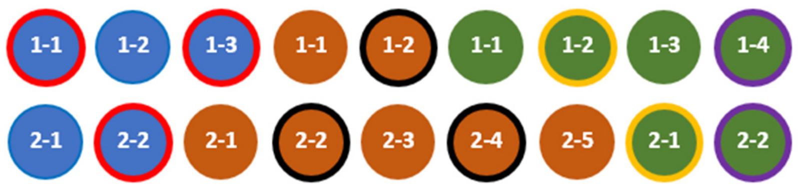
This mathematical definition can be presented as the algorithm for finding a result of the union of two aggregates as shown in Listing 2.

Listing 2: Algorithm of the union operation for two aggregates.

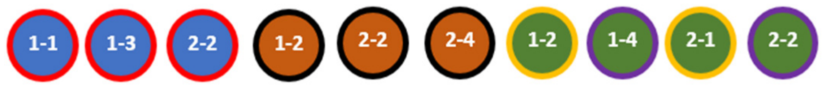
```
Input: aggregates $A_1$ and $A_2$
Output: aggregate $R$
  if $A_1 \div A_2$ then
    for $i = 1 : N$ do
      for $k_i = 1 : n_i^1$ do $r_{k_i} = a_{k_i}^{1,i}$
      for $k_i = \left(n_i^1 + 1\right) : \left(n_i^1 + n_i^2\right)$ do $r_{k_i} = a_{k_i}^{2,i}$
    end
  else if $A_1 \circeq A_2$ then
    for $i = 1 : N^1$ do $\langle r_{k_i} \rangle_{k_i=1}^{n_i} = \left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$
    for $i = \left(N^1 + 1\right) : \left(N^1 + N^2\right)$ do $\langle r_{k_i} \rangle_{k_i=1}^{n_i} = \left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2}$
  else
    for $i = 1 : \left(N^1 + N^2\right)$ do
      if $\left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$ and $\left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2} \in M_i$ then
        for $k_i = 1 : n_i^1$ do $r_{k_i} = a_{k_i}^{1,i}$
        for $k_i = \left(n_i^1 + 1\right) : \left(n_i^1 + n_i^2\right)$ do $r_{k_i} = a_{k_i}^{2,i}$
      else
        $\langle b_{k_i} \rangle_{k_i=1}^{n_i^2} = \left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2}$
        end
      end
      for $i = \left(N^1 + N^2 + 1\right) : \left(N^1 + N^2 + N^b\right)$
        $\langle r_{k_i} \rangle_{k_i=1}^{n_i^2} = \langle b_{k_i} \rangle_{k_i=1}^{n_i^2}$
      end
  end
```

The *intersection* of the aggregates A_1 and A_2 is the aggregate $R_\cap$, which contains components that are common to these aggregates and are ordered according to the rule defined in [18]. This rule can be presented as the algorithm for finding a result of the intersection of two aggregates as shown in Listing 3.

The results of the intersection operation applied to the compatible and quasi-compatible aggregates are shown in Figures 6 and 7, respectively. The elements of the same color marked with the same border color are equal, for example, $a_1^{1,1} = a_3^{1,1} = a_2^{2,1}$ (blue elements with the red border) in Figure 6.



(**a**) The given compatible aggregates.



(**b**) The results of the intersection operation.

Figure 6. An example of applying the intersection operation to two compatible aggregates.

Listing 3: Algorithm of the intersection operation for two aggregates.

```
Input: aggregates $A_1$ and $A_2$
Output: aggregate $R$
  if $A_1 \doteqdot A_2$ then
    for $i = 1 : N$ do
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \in A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
      for $k_i = (n_i^1 + 1) : (n_i^1 + n_i^2)$ do
        if $a_{k_i}^{2,i} \in A_1^i$
          $r_{k_i} = a_{k_i}^{2,i}$
        end
      end
    end
  else if $A_1 \stackrel{\circ}{=} A_2$ then
      $\langle r_{k_i} \rangle_{k_i=1}^{n_i^1} = \varnothing$
  else
  for $i = 1 : (N^1 + N^2)$ do
    if $\left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$ and $\left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2} \in M_i$ then
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \in A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
      for $k_i = (n_i^1 + 1) : (n_i^1 + n_i^2)$ do
        if $a_{k_i}^{2,i} \in A_1^i$
          $r_{k_i} = a_{k_i}^{2,i}$
        end
      end
    end
  end
```

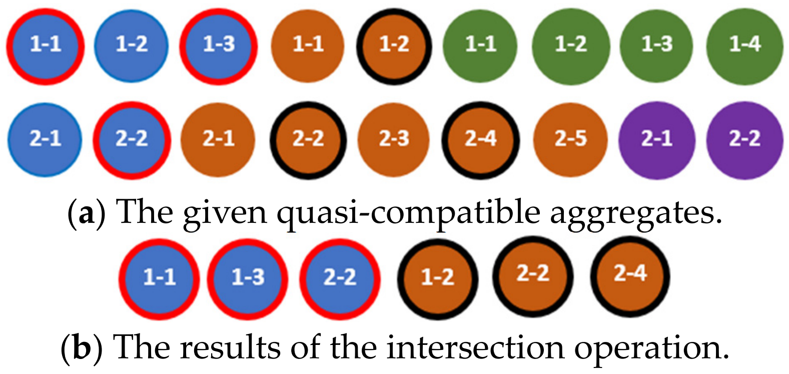

1-1
1-2
1-3
1-1
1-2
1-1
1-2
1-3
1-4
2-1
2-2
2-1
2-2
2-3
2-4
2-5
2-1
2-2
(**a**) The given quasi-compatible aggregates.
1-1
1-3
2-2
1-2
2-2
2-4
(**b**) The results of the intersection operation.

Figure 7. An example of applying the intersection operation to two quasi-compatible aggregates.

The *exclusive intersection* of the aggregates A_1 and A_2 is the aggregate $R_{\neg}$, which contains only components of the A_1 aggregate that are common in these aggregates and are ordered according to the rule defined in [18]. This rule can be presented as the algorithm for finding a result of the exclusive intersection of two aggregates as shown in Listing 4.

Listing 4: Algorithm of the exclusive intersection operation for two aggregates.

```
Input: aggregates $A_1$ and $A_2$
Output: aggregate $R$
  if $A_1 \doteqdot A_2$ then
    for $i = 1 : N$ do
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \in A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
    end
  else if $A_1 \stackrel{\circ}{=} A_2$ then
        ${r_{k_i}}_{k_i=1}^{n_i^1} = \varnothing$
  else
  for $i = 1 : N^1$ do
    if $\left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$ and $\left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2} \in M_i$ then
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \in A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
    end
  end
```

The results of the exclusive intersection operation applied to the compatible and quasi-compatible aggregates are shown in Figures 8 and 9, respectively.

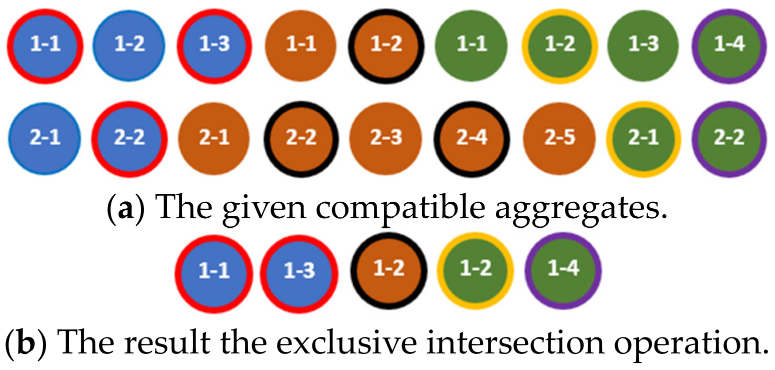


Figure 8. An example of applying the exclusive intersection operation to two compatible aggregates.

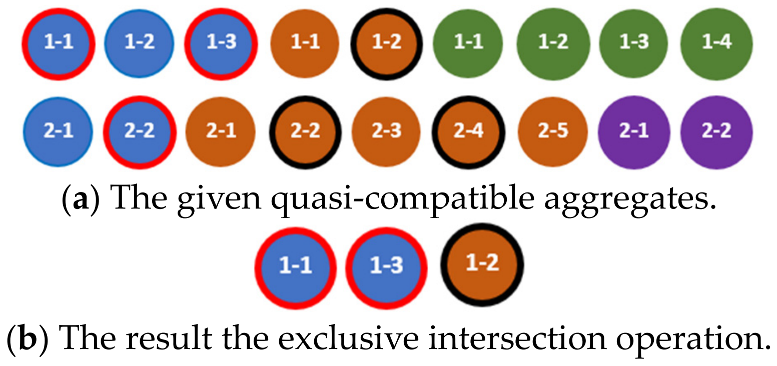

1-1 1-2 1-3 1-1 1-2 1-1 1-2 1-3 1-4
2-1 2-2 2-1 2-2 2-3 2-4 2-5 2-1 2-2
(a) The given quasi-compatible aggregates.
1-1 1-3 1-2
(b) The result the exclusive intersection operation.

Figure 9. An example of applying the exclusive intersection operation to two quasi-compatible aggregates.

The *difference* of the aggregates A_1 and A_2 is the aggregate $R_\backslash$, which contains components of the A_1 aggregate that are not present in the A_2 aggregate and are ordered according to the rule defined in [18]. This rule can be presented as the algorithm for finding a result of the difference of two aggregates as shown in Listing 5.

Listing 5: Algorithm of the difference operation for two aggregates.

```
Input: aggregates $A_1$ and $A_2$
Output: aggregate $R$
  if $A_1 \doteqdot A_2$ then
    for $i = 1 : N$ do
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \notin A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
    end
  else if $A_1 \circeq A_2$ then
    for $i = 1 : N$ do
      $\langle r_{k_i} \rangle_{k_i=1}^{n_i^1} = \langle a_{k_i}^{1,i} \rangle_{k_i=1}^{n_i^1}$
    end
  else
  for $i = 1 : N^1$ do
    if $\langle a_{k_i}^{1,i} \rangle_{k_i=1}^{n_i^1}$ and $\langle a_{k_i}^{2,i} \rangle_{k_i=1}^{n_i^2} \in M_i$ then
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \notin A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
    else
      $\langle r_{k_i} \rangle_{k_i=1}^{n_i^1} = \langle a_{k_i}^{1,i} \rangle_{k_i=1}^{n_i^1}$
    end
  end
```

The result of the difference operation applied to the compatible, quasi-compatible, and incompatible aggregates is shown in Figures 10–12, respectively.

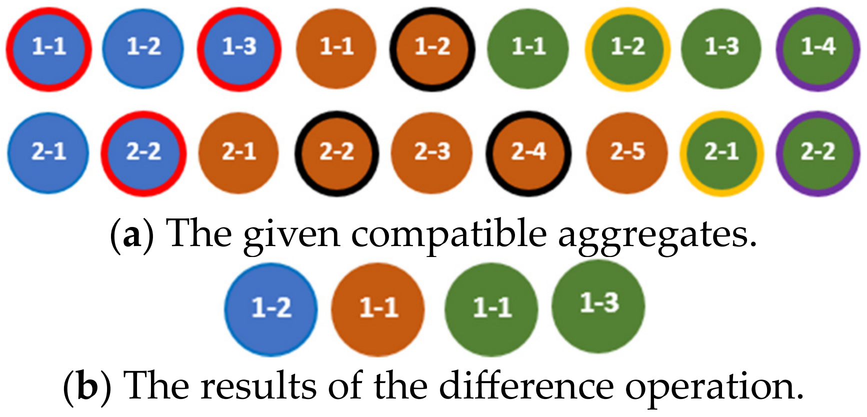
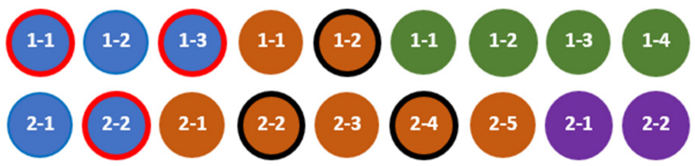
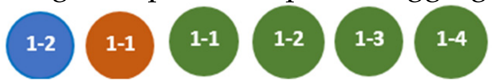

1-1
1-2
1-3
1-1
1-2
1-1
1-2
1-3
1-4
2-1
2-2
2-1
2-2
2-3
2-4
2-5
2-1
2-2
(**a**) The given compatible aggregates.
1-2
1-1
1-1
1-3
(**b**) The results of the difference operation.

Figure 10. An example of applying the difference operation to two compatible aggregates.

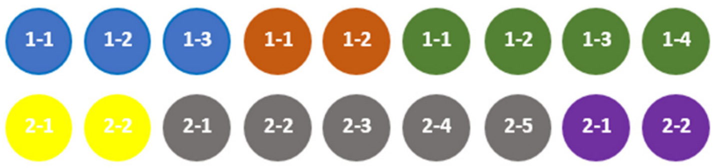


(**a**) The given quasi-compatible aggregates.

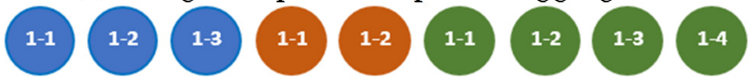


(**b**) The results of the difference operation.

Figure 11. An example of applying the difference operation to two quasi-compatible aggregates.



(**a**) The given quasi-compatible aggregates.

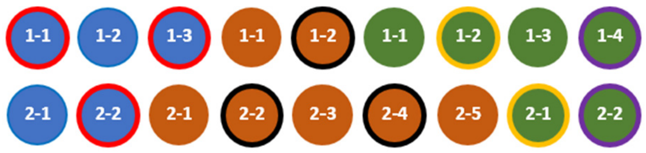


(**b**) The result the difference operation.

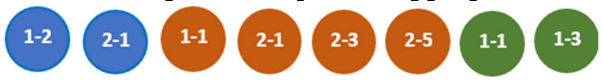
Figure 12. An example of applying the difference operation to two incompatible aggregates.

The *symmetric difference* of the aggregates A_1 and A_2 is the aggregate R_Δ, which contains components of the aggregate A_1 that are not in the aggregate A_2 and components of the aggregate A_2 that are not in the aggregate A_1, and are ordered according to the rule defined in [18]. This rule can be presented as the algorithm for finding a result of the symmetric difference of two aggregates as shown in Listing 6.

The results of the symmetric difference operation applied to the compatible, quasi-compatible, and incompatible aggregates are shown in Figures 13–15, respectively.

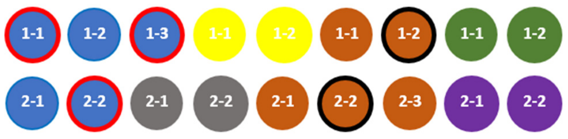


(**a**) The given compatible aggregates.

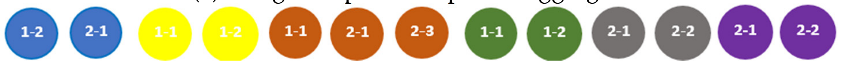


(**b**) The results of the symmetric difference operation.

Figure 13. An example of applying the symmetric difference operation to two compatible aggregates.



(**a**) The given quasi-compatible aggregates.

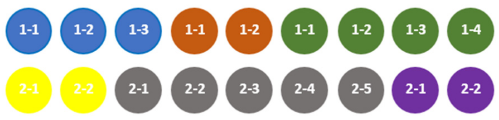


(**b**) The results of the symmetric difference operation.

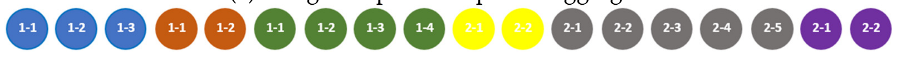
Figure 14. An example of applying the symmetric difference operation to two quasi-compatible aggregates.

Listing 6: Algorithm of the symmetric difference operation for two aggregates.

```
Input: aggregates $A_1$ and $A_2$
Output: aggregate $R$
  if $A_1 \doteqdot A_2$ then
    for $i = 1 : N$ do
      for $k_i = 1 : n_i^1$ do
        if $a_{k_i}^{1,i} \notin A_2^i$
          $r_{k_i} = a_{k_i}^{1,i}$
        end
      end
      for $k_i = (n_i^1 + 1) : (n_i^1 + n_i^2)$ do
        if $a_{k_i}^{2,i} \notin A_1^i$
          $r_{k_i} = a_{k_i}^{2,i}$
        end
      end
    end
  else if $A_1 \circeq A_2$ then
    for $i = 1 : N^1$ do
      $\langle r_{k_i} \rangle_{k_i=1}^{n_i^1} = \left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$
    end
    for $i = 1 : (N^1 + 1) : (N^1 + N^2)$ do
      $\langle r_{k_i} \rangle_{k_i=1}^{n_i^2} = \left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2}$
    end
  else
    for $i = 1 : (N^1 + N^2)$ do
      if $\left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$ and $\left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2} \in M_i$ then
        for $k_i = 1 : n_i^1$ do
          if $a_{k_i}^{1,i} \notin A_2^i$
            $r_{k_i} = a_{k_i}^{1,i}$
          end
        end
        for $k_i = 1 : n_i^2$ do
          if $a_{k_i}^{2,i} \notin A_1^i$
            $r_{k_i} = a_{k_i}^{2,i}$
          end
        end
      else
        $\langle r_{k_i} \rangle_{k_i=1}^{n_i^1} = \left\langle a_{k_i}^{1,i} \right\rangle_{k_i=1}^{n_i^1}$
        $\langle b_{k_i} \rangle_{k_i=1}^{n_i^2} = \left\langle a_{k_i}^{2,i} \right\rangle_{k_i=1}^{n_i^2}$
      end
    end
    for $i = (N^1 + 1) : (N^1 + N^2)$ do
      $\langle r_{k_i} \rangle_{k_i=1}^{n_i^2} = \langle b_{k_i} \rangle_{k_i=1}^{n_i^2}$
    end
  end
```



(**a**) The given quasi-compatible aggregates.



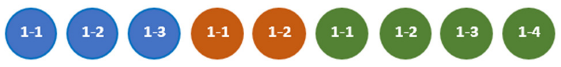
(**b**) The results of the symmetric difference operation.

Figure 15. An example of applying the symmetric difference operation to two incompatible aggregates.

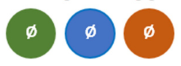
5.2. Ordering Operations

The ordering operations [19] on aggregates are sets ordering, sorting, singling, extraction, and insertion.

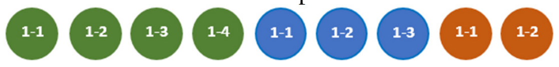
Sets ordering is an ordering operation that reorders the tuple of sets and the corresponding tuple of tuples of the aggregate A, according to the given hiddenly compatible aggregate A_T, which consists of any arbitrary elements, including dummy ones, as shown in Figure 16. The result of *sets ordering* is the aggregate $A_{\vDash} = A \vDash A_T$.



(**a**) The original aggregate.



(**b**) The given template aggregate with dummy element Ø in each tuple.



(**c**) The results of the sets ordering operation.

Figure 16. An example of applying the sets ordering operation.

The sets ordering operation can be fulfilled according to the algorithm in Listing 7.

Listing 7. Algorithm of the sets ordering operation.

```
Input: aggregates $A$ and $A_T$
Output: aggregate $R$
  while not ordered do
    $j = 1$
    for $i = 1 : N$ do
      if $\left\langle a^i_{k_i} \right\rangle^{n_i}_{k_i=1} \in M^T_j$ then
        $\langle r_{k_i} \rangle^{n_i}_{k_i=1} = \left\langle a^i_{k_i} \right\rangle^{n_i}_{k_i=1}$
        $j = j + 1$
        if $j > N$ then
          sets are ordered
        end
      end
    end
  end
```

The sets ordering operation has an important practical value that can be discovered through the following theorem and its corollary.

Theorem 1. *The theorem on compatibility.*
If $\hat{A}_1 = A_1 \vDash A_2$ and $\hat{A}_2 = A_2 \vDash A_1$, then $\hat{A}_1 \doteqdot A_2$ and $\hat{A}_2 \doteqdot A_1$ for $\forall\ A_1$, $\forall\ A_2$, such as $A_1 (\doteqdot)\ A_2$.

Proof of Theorem 1. Let us consider two arbitrary hiddenly compatible aggregates A_1 and A_2, i. e., $A_1 (\doteqdot)\ A_2$, and apply the sets ordering operation to these aggregates in two ways as follows:

1. $\hat{A}_1 = A_1 \vDash A_2$;
2. $\hat{A}_2 = A_2 \vDash A_1$.

Let us consider the aggregates $\hat{A}_1$ and A_2. As $\hat{A}_1$ is the result of applying the sets ordering operation, then by Definition 11, the result of its application to the given hiddenly compatible aggregates A_1 and A_2 is the aggregate $\hat{A}_1 = [\![\{A_2\} \mid \left\langle a^j_i \right\rangle^{n_j}_{i=1}]\!]^N_{j=1}$, where $\left\langle a^j_i \right\rangle^{n_j}_{i=1} \in \langle A_1 \rangle$, $\langle \hat{A}_1 \rangle \equiv \langle A_1 \rangle$, which means $\hat{A}_1 = [\![\{A_2\} \mid \langle A_1 \rangle]\!]$.

According to Definition 1, the aggregate A_2 can be presented as $A_2 = [\![\{A_2\} \mid \langle A_2 \rangle]\!]$.

According to Definition 5, two aggregates are hiddenly compatible if $\{A_1\} \not\equiv \{A_2\}$, $\mid A_1 \mid = \mid A_2 \mid = N$ and $\forall M_j \subset \{A_k\}, j = [1, \ldots, N], k = [1,\ 2]$.

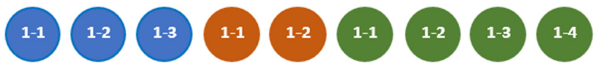
Then, $\hat{A}_1 = [\![\{A_2\} \mid \langle A_1 \rangle]\!]$, $A_2 = [\![\{A_2\} \mid \langle A_2 \rangle]\!]$, $\mid \hat{A}_1 \mid = \mid A_2 \mid = N$.

According to Definition 2, the aggregates $\hat{A}_1$ and A_2 are compatible because $\mid \hat{A}_1 \mid = \mid A_2 \mid$ and $\{\hat{A}_1\} \equiv \{A_2\}$, which we would have to prove.

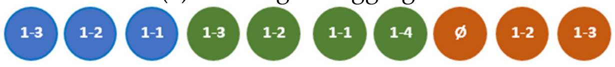
The compatibility of the aggregates of A_1 and $\hat{A}_2$ can be proved analogously. □

Remark 1. *The corollary of Theorem 1.*
The practical significance of applying the sets ordering operation to hiddenly compatible aggregates A_1 and A_2 is that the components of one aggregate (A_1) can be rearranged according to the sequence of the components of the second aggregate (A_2); therefore, as a result of applying the sets ordering operation, the hiddenly compatible aggregates become compatible.

The *sorting* operation [19] yields a new (sorted) sequence of elements of a certain tuple named the primary tuple. The result of applying the sorting operation to an aggregate A is an aggregate $A_\uparrow = A \uparrow \overline{a}^k$ (for ascending sorting) or $A_\downarrow = A \downarrow \overline{a}^k$ (for descending sorting) in all tuples, in which the elements are reordered according to the new ordering of the indices of the elements of the sorted primary tuple $\overline{a}^k$. If some tuple is shorter than the primary tuple, it is appended by a dummy element and is then sorted. If some tuple is longer than the primary one, the elements with indices that exceed the largest index in the primary tuple, are not sorted (remain on their positions). An example of the result of sorting operation is given in Figure 17.



(**a**) The original aggregate.



(**b**) The results of the sorting operation.

Figure 17. An example of applying the sorting operation.

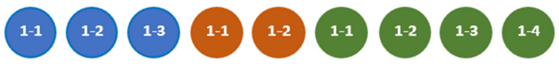
The sorting operation can be realized using any appropriate sorting algorithm [20], including the one shown in Listing 8.

Listing 8: Algorithm of the ascending sorting operation.

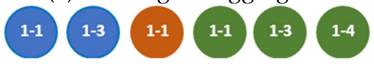
```
Input: aggregate $A$ and primary tuple $\left\langle a^p_{k_p} \right\rangle^{n_p}_{k_p=1}$
Output: aggregate $R$
  for $k_p = 1 : n_p$ do
    for $k_m = 1 : (n_p - k)$ do
      swapped is false
      if $a^p_{k_m} > a^p_{k_m+1}$ do
        for $i = 1 : N$ do
          $r^i_{k_m} = a^i_{k_m+1}$
          $r^i_{k_m+1} = a^i_{k_m}$
        end
        swapped is true
      end
    end
    if swapped is false
      break loop
    end
  end
```

The *extraction* operation removes a certain element from the primary tuple of an aggregate. The result of extraction of the element a_m^1 from the aggregate A is the aggregate $A_{\ltimes} = A \ltimes a_m^1$ in all tuples, from which the elements with index m are removed.

An example of the result of the extraction operation for $m = 2$ is given in Figure 18.



(**a**) The original aggregate.



(**b**) The result of the extraction operation.

Figure 18. An example of applying the extraction operation.

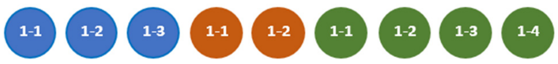
The extraction operation can be fulfilled according to the algorithm in Listing 9.

Listing 9: Algorithm of the extraction operation.

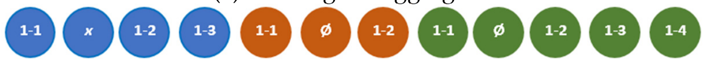
```
Input: aggregate $A$, number of set $k$ and position of element $m$
Output: aggregate $R$
  for $i = 1 : N$ do
    for $k_i = 1 : (m - 1)$ do $r_{k_i} = a_{k_i}^i$
    for $k_i = m : (n_i - 1)$ do $r_{k_i} = a_{k_i+1}^i$
  end
```

The *insertion* operation adds a given element to the known position in the primary tuple of an aggregate. The result of insertion of the element a_m^1 to the aggregate A is the aggregate $A_{\rtimes} = A \rtimes a_m^1$ in all tuples (except the primary one), to which dummy elements with index m are added.

An example of the result of insertion of a new element x to the second position of the primary tuple is shown in Figure 19, where the indices of the shifted elements remain as they numerated in the original tuple for explanatory purposes.



(**a**) The original aggregate.



(**b**) The results of the insertion operation.

Figure 19. An example of applying the insertion operation.

The insertion operation can be fulfilled according to the algorithm in Listing 10.

Listing 10: Algorithm of the insert operation.

```
Input: aggregate $A$, number of set $k$ and position of element $m$
Output: aggregate $R$
  for $i = 1 : N$ do
    for $k_i = 1 : (m - 1)$ do
      $r_{k_i} = a_{k_i}^i$
    end
    if $k = i$ then
      $r_m = a_m^k$
    else
      $r_m = \varnothing$
    end
    for $k_i = m : n_i$ do
      $r_{k_i+1} = a_{k_i}^i$
    end
  end
```

The singling operation removes the duplicate values that are located next to each other from the primary tuple of the aggregate; elements with the same indices as those of the removed duplicates in the primary tuple are simultaneously removed in all other tuples.

An example of the result of the singling operation is given in Figure 20. Elements of the same color marked with the same border color are equal.

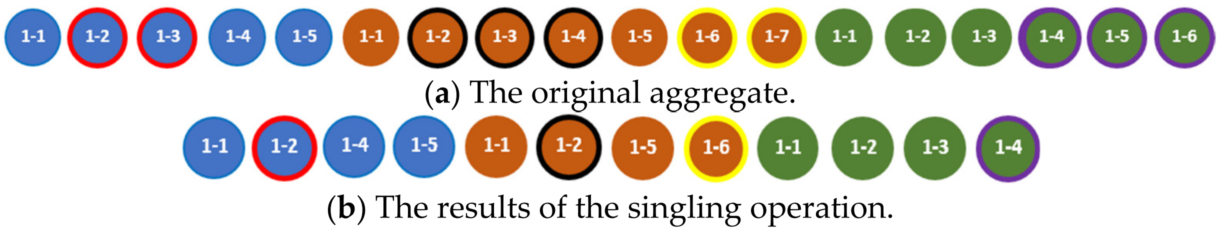


Figure 20. An example of applying the singling operation.

The singling operation can be fulfilled according to the algorithm in Listing 11.

Listing 11: Algorithm of the singling operation.

```
Input: aggregate $A$
Output: aggregate $R$
  $m = 2$
  while not all found do
    for $k_1 = m : n_1$ do
      if $a^1_{k_1} = a^1_{k_1-1}$ then
        $m = k_1$
        break loop
      end
      all found
    end
    for $i = 1 : N$ do
      extract $a^i_m$
    end
  end
```

6. Multi-Image Notion and Formation Algorithm

The multi-image concept [21] is an essential part of the approach presented in this paper as it enables the required formal description of the sequences of multimodal data about the object under study.

Definition 6. *A multi-image is a non-empty aggregate specified as:*

$$I = [\![T, M_1, \ldots, M_N \mid \langle t_1, \ldots, t_\tau \rangle, \left\langle a^1_1, \ldots, a^1_{n_1} \right\rangle, \ldots, \left\langle a^N_1, \ldots, a^N_{n_N} \right\rangle]\!] \tag{13}$$

where T is a set of time values; $\tau \geq n_j, j \in [1, \ldots, N]$.

From a practical point of view, multi-image refers to a structure of consolidated temporal multimodal data sets that describe different aspects of the same object. Let us formulate an algorithm that enables the formation of multi-image for an arbitrary object described through several temporal multimodal data sets.

The inputs to the algorithm of the object's multi-image formation are temporal data sets and the data type (modality) of each set. The algorithm consists of seven steps. The result of the algorithm is multi-image, presented in the form of an ordered collection of temporal multimodal data.

The first step of the algorithm is the formalization of the object's multi-image data structure. This step is performed according to the requirements for the object's description.

Multi-image can be defined mathematically, according to (13), or in any other way that allows a researcher to unambiguously specify the sequence of data sets and their modality.

At the second step of the algorithm, multi-image specification is decomposed into a set of specifications of partial multi-images as seen in (14).

$$I_j = [\![T, M_j \mid \left\langle t_1, \ldots, t_{\tau_j} \right\rangle, \left\langle a_1, \ldots, a_{n_j} \right\rangle]\!] \tag{14}$$

where T is the set of time values; M_j is the set of data values of j-modality; $\tau_j, n_j \in \mathbb{N}$; $\tau_j \geq n_j$; $j \in [1, \ldots, N]$; and N is the number of data sets.

At the third step, data are obtained. The procedure for receiving data involves determining the method (protocol, format) of data transmission, determining the time period for data transmission, establishing a connection with the data source and receiving data in a specified way and in a specified format.

At the fourth step, the data of each modality are prepared to form the corresponding partial multi-image. A partial multi-image is an aggregate that includes two tuples of elements, namely a tuple of time values and a tuple of elements of a certain modality. Elements are individual values or ordered sets of values (homogeneous or heterogeneous). Data preparation, which is performed at this step, consists of determining (detecting) time values in the set of data obtained at the previous stage. This procedure can be either trivial when the data format requires explicit presentation of time values for the temporal data component, or complex when the time values are presented in an implicit form or are ambiguous. A method for detecting hidden or ill-defined time values must be developed for each specific data format.

At the fifth step, partial multi-images are combined into a single multi-image. The multi-image merging procedure includes two actions: normalization and uniting.

1. Normalization of partial multi-images (a normalized multi-image (normalized tuple) is a multi-image (tuple) to the elements of which dummy elements are added; a dummy element is a value that is absent in the tuple before its normalization; an empty element $\varnothing$ can be used as a dummy element; at other steps of the multi-image processing, dummy elements are ignored):

$$\begin{gathered} \hat{I}_j = I_j \rtimes \left(\langle \varnothing \rangle_{k=1}^{E_i^j} \succ a_{n_j}^j \right) = [\![T, M_j \mid \langle t_i \rangle_{i=1}^{\tau_j}, \langle \langle a_{i_j}^j \rangle_{i_j=1}^{n_j}, \langle \varnothing \rangle_{l=1}^{E_i^j} \rangle]\!] = \\ = [\![T, M_j \mid \langle t_i \rangle_{i=1}^{\tau_j}, \left\langle \hat{a}_{i_j}^j \right\rangle_{i_j=1}^{n_j+E_i^j}]\!] \end{gathered} \tag{15}$$

   where $\rtimes$ is the insertion operation; E_i^j is the number of dummy elements; $E_i^j = \left(\sum_{j=1}^{N} \tau_j \right) - n_j$; $\hat{a}_{i_j}^j$ is an element of a normalized tuple; and $j \in [1, \ldots, N]$.

2. Uniting the normalized partial multi-images:

$$I_U = \bigcup_{j=1}^{N} \hat{I}_j = [\![T, M_1, \ldots M_N \mid \langle \langle t_i \rangle_{i=1}^{\tau_j} \rangle_{j=1}^{N}, \langle \hat{a}_{i_1} \rangle_{i_1=1}^{n_1+E_i^1}, \ \ldots \langle \hat{a}_{i_N} \rangle_{i_N=1}^{n_N+E_i^N}]\!] \tag{16}$$

At the sixth step, the multi-image obtained at the previous stage is sorted by the tuple of time values:

$$I_S = I \uparrow \bar{t} = [\![T, M_1, \ldots M_N \mid \langle \langle t_\sigma \rangle_{\sigma=1}^{\tau_j} \rangle_{j=1}^{N}, \langle \hat{a}_{\sigma_1} \rangle_{\sigma_1=1}^{n_1+E_i^1}, \ \ldots \langle \hat{a}_{\sigma_N} \rangle_{\sigma_N=1}^{n_N+E_i^N}]\!], \tag{17}$$

where σ and σ_j are indices specifying the order of the elements in the sorted tuple and $j \in [1, \ldots, N]$.

At the seventh step, the sorted multi-image obtained at the previous step is singled by a tuple of time values:

$$I = I_S \parallel \bar{t} = \llbracket T, M_1, \ldots M_N \mid \langle t_\sigma \rangle_{\sigma=1}^{(\sum_{j=1}^{N} \tau_j)-\delta}, \langle \hat{a}_{\sigma_1} \rangle_{\sigma_1=1}^{n_1+E_i^1-\delta}, \ldots, \langle \hat{a}_{\sigma_N} \rangle_{\sigma_N=1}^{n_N+E_i^N-\delta} \rrbracket, \quad (18)$$

where δ is the number of discarded time value duplicates.

The obtained I is the final multi-image that represents the consolidated data describing different aspects of the object under study.

7. Discussion

Let us consider a use case in the area of healthcare that demonstrates the practical use of the proposed approach of multimodal data processing (Figure 21).

The healthcare use case assumes that the data describing the patient's health status can be obtained from multiple sources, including medical investigation devices (MRI scanners, CT scanners, ECG machines, etc.), manual medical measurement tools and devices (pulse oximeters, thermometers, sphygmomanometers, etc.), test systems (e.g., blood testing), and medical documentation records (treatment events). Data sets collected from these sources are temporal multimodal and semantically interconnected as they describe different features of the same object that is a patient's organism. The multi-image concept and ASA operations can be employed to consolidate such a compound data collection and implement the logic of its processing.

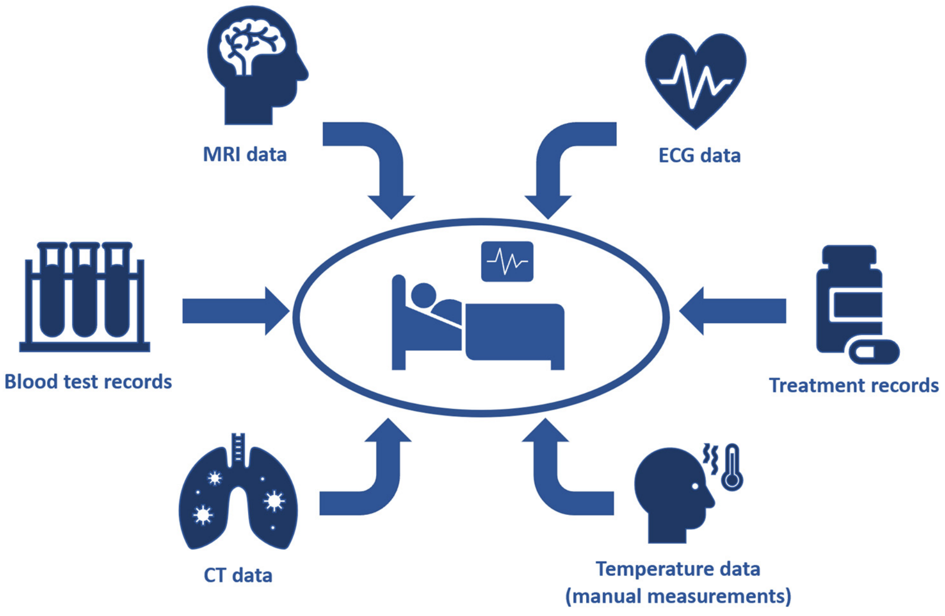


Figure 21. An example of temporal multimodal data processing context.

Let us consider an example which demonstrates the formation of multi-image from several timewise data sequences according to the algorithm proposed in Section 6 and the algorithms of the ASA operations presented in Section 5.

Suppose that the patient's blood pressure readings (systolic and diastolic), pulse rate, and oxygen saturation level are being measured several times a day. A pair of blood pressure values $\left(a_i^{press(s)}, a_i^{press(d)}\right)$ are to be received from the sphygmomanometer, and the values of pulse rate a_i^{pulse} and oxygen saturation level a_i^{satur} are to be received from a pulse oximeter. The number of measurements conducted by the sphygmomanometer is n_1; the number of measurements conducted by the pulse oximeter is n_2.

As a result of patient health parameters monitoring, the following multi-image must be obtained:

$$I = \llbracket T, M_{press(s,d)}, M_{pulse}, M_{satur} \mid \langle t_i \rangle_{i=1}^{n_1+n_2}, \left\langle \left(a_i^{press(s)}, a_i^{press(d)} \right)_{i=1}^{n_1} \right\rangle, \left\langle a_i^{pulse} \right\rangle_{i=1}^{n_2}, \langle a_i^{satur} \rangle_{i=1}^{n_2} \rrbracket.$$

According to the algorithm of multi-image formation, the received data are represented as two partial multi-images—a partial multi-image I_1 containing values received from the first device (sphygmomanometer) and a partial multi-image I_2 containing values received from the second device (pulse oximeter):

$$I_1 = [\![T, M_{press(s,d)} \mid \langle 8:30,\ 12:05, 16:10,\ 21:00\rangle, \Big\langle (160,78), (172,81), (155,76), (158,75)\rangle]\!],$$

$$I_2 = [\![T, M_{pulse}, M_{satur} \mid \langle 8:30,\ 15:08,\ 21:00\rangle, \langle 63,75,66\rangle, \langle 95,98,96\rangle]\!].$$

To obtain the multi-image, these partial multi-images must be properly merged. For this purpose, at first, we need to normalize these partial multi-images, i.e., apply the insertion operation and add dummy elements to the end of the tuple of blood pressure values and to the beginning of both the tuple of heart rate values and the tuple of oxygen saturation level values:

$$\hat{I}_1 = [\![T, M_{press(s,d)} \mid \langle 8:30,\ 12:05, 16:10,\ 21:00\rangle, \langle (160,78), (172,81), (155,76), (158,75), \varnothing, \varnothing, \varnothing\rangle]\!],$$

$$\hat{I}_2 = [\![T, M_{pulse}, M_{satur} \mid \langle 8:30,\ 15:08,\ 21:00\rangle, \langle \varnothing, \varnothing, \varnothing, \varnothing, 63,75,66\rangle, \langle \varnothing, \varnothing, \varnothing, \varnothing, 95,98,96\rangle]\!].$$

After normalization, we apply the union operation to the normalized aggregates:

$$I_U = \hat{I}_1 \cup \hat{I}_2 = [\![T, M_{press(s,d)}, M_{pulse},\ M_{satur} \mid \langle\ 8:30,\ 12:05, 16:10,\ 21:00, 8:30,\ 15:08,\ 21:00\rangle, \langle (160,78), (172,81), (155,76), (158,75), \varnothing, \varnothing, \varnothing\rangle, \Big\langle\ \varnothing, \varnothing, \varnothing, \varnothing, 63,75,66\rangle, \langle \varnothing, \varnothing, \varnothing, \varnothing, 95,98,96\rangle]\!].$$

Next, we need to sort the obtained multi-image I_U by the time tuple using the corresponding ASA operation:

$$I_S = I_U \uparrow \langle t_i\rangle_{i=1}^{n_1+n_2} = [\![T, M_{press(s,d)}, M_{pulse},\ M_{satur} \mid \langle\ 8:30, 8:30,\ 12:05, 15:08, 16:10,\ 21:00,\ 21:00\rangle, \langle (160,78), \varnothing, (172,81), \varnothing, (155,76), (158,75), \varnothing\rangle, \Big\langle\ \varnothing, 63, \varnothing, 75, \varnothing, \varnothing, 66\rangle, \langle \varnothing, 95, \varnothing, 98, \varnothing, \varnothing, 96\rangle]\!].$$

The tuples are now ordered, but they include duplicate time values. To remove them, we apply the singling operation:

$$I = I_S \parallel \langle t_i\rangle_{i=1}^{n_1+n_2} = [\![T, M_{press(s,d)}, M_{pulse},\ M_{satur} \mid \langle\ 8:30,\ 12:05, 15:08, 16:10, 21:00\rangle, \langle (160,78), (172,81), \varnothing, (155,76), (158,75)\rangle, \langle 63, \varnothing, 75, \varnothing, 66\rangle, \langle 95, \varnothing, 98, \varnothing, 96\rangle]\!].$$

As a result, we obtain a multi-image I that contains data from two independent but semantically interconnected examinations of the patient. This consolidated data structure can be used for further data processing.

This simple example demonstrates that the ASA offers operations which consider both data features defined in Section 3: multimodality (logical operations allow us to implement any logic for modality-wise data processing) and temporality (ordering operations enable timewise data processing). This principle of the proposed approach is valid for any data type because according to Definition 1, an aggregate can include both separate elements and composite elements (homogeneous or heterogeneous values).

The mathematical approach introduced in the ASA is implemented in the domain-specific programming language ASAMPL [22–24].

8. Conclusions

Algorithms for performing operations of the Algebraic System of Aggregates (ASA) are proposed in the paper. A feature of this algebraic system is the consideration of the sequence of elements (tuples, aggregates) when performing operations, including brand-new ordering operations. The properties of the ASA make it possible to use it for the formal description of objects under observation and for the development of methods for processing temporal multimodal data.

An algorithm for creating an object's multi-image is also proposed in the paper. The input data for this algorithm are separate sets of temporal data. The algorithm consists of seven steps, which include the formation of the multi-image data structure of the object under study, decomposition of the multi-image specification into a set of partial multi-image specifications, obtaining and preparing separate data sets, combining partial multi-images into a single multi-image, sorting the multi-image, and thinning the sorted multi-image by tuple time values. The result of algorithm execution is a multi-image of an object containing synchronized and aggregated sequences of temporal multimodal data characterizing this object.

The proposed mathematical approach can be used for a formal description of digital models in various tasks, including a digital twin design, semantic model construction, and temporal multimodal data aggregation and processing. The mathematical concepts offered by the ASA can also be employed for the consolidation of data for multimodal data processing in multimodal machine learning tasks. Further work should aim to increasing the efficiency of the proposed algorithms.

Author Contributions: Conceptualization, A.P. and Y.S.; methodology, Y.S. and I.D.; software, O.S.; validation, A.P., Y.S. and I.D.; writing—original draft preparation, Y.S. and O.S.; writing—review and editing, A.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Morency, L.-P.; Liang, P.P.; Zadeh, A. Tutorial on Multimodal Machine Learning. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts, Seattle, WA, USA, 10–15 July 2022; Association for Computational Linguistics: Stroudsburg, PA, USA, 2022; pp. 33–38.
2. Baltrušaitis, T.; Ahuja, C.; Morency, L.-P. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 423–443. [CrossRef] [PubMed]
3. Jesus, P.; Baquero, C.; Almeida, P.S. A Survey of Distributed Data Aggregation Algorithms. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 381–404. [CrossRef]
4. Ribeiro, R.A.; Falcão, A.; Mora, A.; Fonseca, J.M. FIF: A fuzzy information fusion algorithm based on multi-criteria decision making. *Knowl.-Based Syst.* **2014**, *58*, 23–32. [CrossRef]
5. Oliveira, D.; Martins, L.; Mora, A.; Damásio, C.; Caetano, M.; Fonseca, J.; Ribeiro, R.A. Data fusion approach for eucalyptus trees identification. *Int. J. Remote Sens.* **2021**, *42*, 4087–4109. [CrossRef]
6. Lahat, D.; Adali, T.; Jutten, C. Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects. *Proc. IEEE* **2015**, *103*, 1449–1477. [CrossRef]
7. Marinoni, A.; Chlaily, S.; Jutten, C. Addressing Reliability of Multimodal Remote Sensing to Enhance Multisensor Data Fusion and Transfer Learning. In Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS 2020), Waikoloa, HI, USA, 26 September 2020; pp. 3896–3899. [CrossRef]
8. Gaonkar, A.; Chukkapalli, Y.; Raman, P.J.; Srikanth, S.; Gurugopinath, S. A Comprehensive Survey on Multimodal Data Representation and Information Fusion Algorithms. In Proceedings of the 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 25–27 June 2021; pp. 1–8. [CrossRef]
9. Oliveira, J.P.; Lourenço, M.; Oliveira, L.; Mora, A.; Oliveira, H. A Data Fusion of IoT Sensor Networks for Decision Support in Forest Fire Suppression. In *Internet of Things. Technology and Applications. IFIPIoT 2021*; Camarinha-Matos, L.M., Heijenk, G., Katkoori, S., Strous, L., Eds.; IFIP Advances in Information and Communication Technology; Springer: Cham, Switzerland, 2022; Volume 641. [CrossRef]
10. Liang, P.P.; Zadeh, A.; Morency, L.-P. Foundations and Recent Trends in Multimodal Machine Learning: Principles, Challenges, and Open Questions. *arXiv* **2023**, arXiv:2209.03430. [CrossRef]
11. Guo, W.; Wang, J.; Wang, S. Deep Multimodal Representation Learning: A Survey. *IEEE Access* **2019**, *7*, 63373–63394. [CrossRef]
12. Kline, A.; Wang, H.; Li, Y.; Dannis, S.; Hutch, M.; Xu, Z.; Wang, F.; Cheng, F.; Luo, Y. Multimodal machine learning in precision health: A scoping review. *Npj Digit. Med.* **2022**, *5*, 171. [CrossRef] [PubMed]
13. Wei, W. *Time Series Analysis*; Pearson Addison Wesley: San Francisco, NY, USA, 2006; 614p.
14. Hannan, E.J. *Multiple Time Series*; John Wiley and Sons: Hoboken, NJ, USA, 2009; 535p.

15. Fraenkel, A.A.; Bar-Hillel, Y.; Levy, A. *Foundations of Set Theory*; Elsevier: Hoboken, NJ, USA, 1973; 415p.
16. Petrovsky, A.B. Structuring techniques in multiset spaces. In *Multiple Criteria Decision Making*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 174–184.
17. Petrovsky, A.B. Multiattribute sorting of qualitative objects in multiset spaces. In *Multiple Criteria Decision Making in New Millennium*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 124–131.
18. Dychka, I.A.; Sulema, Y.S. Logical Operations in Algebraic System of Aggregates for Multimodal Data Representation and Processing. *Sci. J. KPI Sci. News* **2018**, *6*, 44–52. [CrossRef]
19. Dychka, I.A.; Sulema, Y.S. Ordering Operations in Algebraic System of Aggregates for Multi-Image Data Processing. *Sci. J. KPI Sci. News* **2019**, *1*, 15–23. [CrossRef]
20. Knuth, D. *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd ed.; Addison-Wesley: Boston, MA, USA, 1998; ISBN 0-201-89685-0. Available online: https://dl.acm.org/doi/10.5555/280635 (accessed on 15 March 2023).
21. Pester, A.; Sulema, Y. *Multimodal Data Representation Based on Multi-Image Concept for Immersive Environments and Online Labs Development*; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2021. [CrossRef]
22. Sulema, Y. ASAMPL: Programming Language for Mulsemedia Data Processing Based on Algebraic System of Aggregates. In *Interactive Mobile Communication Technologies and Learning. IMCL 2017*; Advances in Intelligent Systems and Computing; Auer, M., Tsiatsos, T., Eds.; Springer: Cham, Switzerland, 2018; Volume 725. [CrossRef]
23. Peschanskyi, D.; Budonnyi, P.; Sulema, Y.; Andres, F.; Pester, A. Temporal Data Processing with ASAMPL Programming Language in Mulsemedia Applications. In *Artificial Intelligence and Online Engineering. REV 2022*; Lecture Notes in Networks and Systems; Auer, M.E., El-Seoud, S.A., Karam, O.H., Eds.; Springer: Cham, Switzerland, 2023; Volume 524. [CrossRef]
24. ASAMPL Compiler and Library. Available online: https://github.com/orgs/Asampl-development-team/repositories (accessed on 15 March 2023).