The fuzzification of an information architecture for information integration

Rico A. R. Picone[1,2], Jotham Lentz[1,2], and Bryan Powell[2]

[1] Department of Mechanical Engineering, Saint Martin's University
[2] Dialectica, LLC

Abstract. We present a new information architecture based on one recently introduced to structure categorized but otherwise unstructured information. The new architecture is based on fuzzy set theory subset operations that define graph theory nodes. Two types of graph edges are defined such that a user interface based on this architecture can logically minimize the number of visible navigable edges and atoms of information. This minimization is understood to be one of the primary advantages of the architecture for human-computer interaction due to its mitigation of information overload. The advantages of hierarchical, organic, and sequential information architectures are fused by the new architecture and the dialectical method is also integrated—all of which are intended to enhance human-computer interaction. The new architecture can easily incorporate quantitative information, which can be converted into a fuzzy set theory representation with fuzzy clustering and other techniques. Moreover, traditionally qualitative information such as narrative, audio, and video, although naturally represented with crisp sets, can be represented with fuzzy sets. Therefore, the new architecture can fuse traditionally disparate types of information.

1 Introduction

Memory is recategorization. This identity is supported by recent neuroscience, psychology, and artificial intelligence research [4,16,14], and even if the provocative identity is not strictly true, it provides insight into the memory; as Rosenfield puts it:

> We can recognize paintings of Picasso as well as adept imitations of Picasso. When we recognize a painting we have never seen as a Picasso or as an imitation, we are doing more than recalling earlier impressions. We are categorizing: Picasso and fakes. Our recognition of paintings or of people is the recognition of a category, not of a specific item. People are never exactly what they were moments before, and objects are never seen in exactly the same way.[3] [16]

[3] Quoted from Pfeifer [14, pp. 311-12].

Memory plays a central role in intelligence; furthermore, the computer's relative stability of memory is one of its most promising features for the enhancement of intelligence, be it artificial or human. For these reasons, computers have long used forms of categorization to store and present information: the two most striking architectures are the venerable hierarchy and the organic tag-based systems. The authors have previously presented an information architecture—the *dialectical architecture*—with a structure explicitly designed to incorporate the advantages of each of these [15]. In the present work, we fuzzify this architecture in order to include information best categorized in each category to a certain degree—that is, fuzzily categorized information. This type of information is especially important in applications such as robotics in which sensor information is quantitative. Well-established techniques such as fuzzy clustering [13,6] and direct assignment of membership functions can assign each atom of information fuzzy categories (sets). The architecture is developed as a method for human-computer interaction to enhance human intelligence through integrating disparate types of information—narrative, audio, video, and now quantitative data—into a single representation fundamentally based on categorization. We believe beginning with human intelligence enhancement (a worthy application) may provide insight into artificial intelligence development as well.

The fuzzy dialectical architecture, like the "crisp" dialectical architecture, unites three information "planes." The first is the *structure* plane, which is built from fuzzy set operations that define relations among nodes defined by category (set) intersections. The second is the *flow* plane, which allows information to be distributed through the structure in a sequential fashion. Flows can represent many types of information: narrative, audio, video, and data streams. The third and final plane is the *dialectic* plane, which provides a mechanism for flows to evolve within the framework of thesis–antithesis–synthesis. Together, these three planes comprise the fuzzy dialectical architecture, as will be described in detail in Sec. 2, which especially focuses on the fuzzification of the dialectical architecture. In Sec. 3, algorithmic considerations are explored. The human interface and a specific instantiation are described in Sec. 4.

2 Fuzzifying the dialectical architecture

The dialectical information architecture was introduced as a way of enhancing human intelligence by a synthesis of structure, flow, and dialectic. These teleological foundations remain intact and the methodology has been developed to include an additional type of information, that which is quantitative. The primary tool for this is fuzzy set theory, with which quantitative information can be interpreted qualitatively in the form of categories. The original or "crisp" architecture defined its structure from crisp set theoretic relations from unstructured categorized information; in this section, the structure of the fuzzy architecture will be defined from fuzzy set theoretic relations from unstructured (fuzzily) categorized information.

Consider a collection of data, each member of which we call an *atom*. Each atom is associated to a certain degree with a collection of *categories* which are represented as fuzzy sets. Crisp set operations union $\cup$ and intersection $\cap$ are analogous the fuzzy set operations union and intersection [17,25]. We exploit this analog to define the fuzzy dialectical architecture in a way similar to the definition of the crisp dialectical architecture, which made much use of the crisp set operations.

The fuzzy structure is, as its crisp analog, a *directed graph* of nodes and edges [21,2]. Other than the "universal" *union node*, which contains all atoms, every node in the graph represents the fuzzy intersection of a collection of categories (fuzzy sets). Just as an atom can belong to a given category with membership value in the interval $[0, 1]$, with zero meaning "no" membership and unity meaning "full" membership, so an atom can belong to a given node to a certain degree (membership value). This degree is computed from the fuzzy intersection operation, which returns the minimum membership value for a given atom shared between two nodes; i.e. let the element x in the universe X have membership $\mu_A(x)$ in fuzzy set A, where μ_A is the membership function for set A, let x have membership $\mu_B(x)$ in fuzzy set B with membership function μ_B, and let $\wedge$ be the operator that takes the minimum of its two arguments—then the membership of x in the fuzzy intersection $A \cap B$ is [17]

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x). \tag{1}$$

Directed edges connect the nodes to generate a natural hierarchy. All edges are defined by *has a priori subcategory* relations or s-relations; for instance, the node $A \cap B$ is an *a priori* subcategory (fuzzy subset) of fuzzy sets A and B. This generates a natural hierarchy with graph *levels* defined by the number of categories that intersect to define the node; e.g. node $A \cap B$ has level two.

Two types of s-relation are defined: the suggestively named (1) *has visible a priori subcategory* or vs-relation and (2) *has hidden a priori subcategory* or hs-relation. The definition of the hs-relation first requires the concept of a *metacategory*. A "minimal" metacategory for a given node is a collection of subcategories that contain as a subset all atoms associated with the node. A node's vs-relations are those that have tails connected to the node and heads connected to subcategory nodes contained in a minimal metacategory. By minimal, we mean containing the minimum number of subcategories to fully contain all atoms. An hs-relation is defined as any s-relation that is not a vs-relation.[4]

Finally, atoms themselves can be either "visible" or "hidden," names suggestive of how the user interface in later sections will be defined. An atom is visible at a given node if and only if it has nonzero membership in all categories intersected to define the node and zero membership in all others. This definition requires that an atom be visible in one and only one node in the structure.

[4] These definitions have strong parallels in [15], where more mathematically oriented definitions are presented. We favor a narrative approach here. The interested reader may find the explicit mathematical definitions of the previous work elucidating.

2.1 Visibility and hiddenness

The names given to the two types of atoms and s-relations—"visible" and "hidden"—are a crucial aspect of the structure's advantage for intelligence amplification in human-computer interaction. In a user interface (one instantiation to be discussed in Sec. 4), these signifiers will be taken literally: at a given node, hidden atoms and hidden s-relations (edges) will not be presented to the user. The definition of each guarantees that a hidden atom will be visible if one navigates via visible s-relations to a lower level. The primary advantage of this from a usability standpoint is that the user is not inundated with as much information, one of the key aspects of a hierarchy, while remaining in a logically categorized structure—the other key aspect of a hierarchy.

2.2 Structure as estimation

Let us consider what type of structure this graph has. It is constructed from a collection of fuzzily categorized atoms (in the case of quantitative information, these atoms are data points with membership values in each category). For a given variable, say temperature, the subset relationships are pre-defined by the membership function of the data; e.g. "luke-warm" will be a subset of "warm." However, the inter-variable relationships are typically not so; for instance, "cold" might be a subset of "high-pressure." The structure defined here can be understood as an estimation process for these relationships, one of several applications to be discussed in later sections.

2.3 Organic hierarchy

The term *organic hierarchy* was introduced when defining the crisp dialectical architecture [15], and it still applies to the fuzzy architecture. It is "organic" in the sense that it evolves with each new atom's introduction to the structure. Unlike a traditional static hierarchy that requires insertion into the structure at a specific node, an organic hierarchy evolves with the information, and a user need not explicitly define the hierarchy, which is implicit in the user's categorization of each atom.

2.4 Invariance of path

Another aspect of the crisp architecture that ports almost directly to the fuzzy architecture is that of the invariance of path—that is, the fact that navigation of the structure is invariant to the order in which one navigates. Let us represent each navigation along a vs-relation as the "selection" of the additional category for the intersection that defines the edge's head node. Let each selection add that category to the path, similar to a traditional file system path (e.g. `/A/B/C`). For the dialectical architecture, the order of the selection is inconsequential; for instance, `/A/B/C`, `/B/A/C`, and `/C/A/B` all point to the same node, due to the invariance of the fuzzy intersection operation.

2.5 Fuzzy flows

The concept of a *flow* was introduced in the context of the crisp dialectical architecture [15]. It's definition—a flow is a series of atoms—applies directly to the fuzzy dialectical architecture, but unique implications emerge. Previously, flows have been used to represent the sequential aspect of several types of information, such as narrative, audio, and video. In a fuzzy dialectical architecture representing quantitative information, each data point is an atom and a data stream is a flow. Thus each atom should not be presented to a user as an isolated data point at each node, but should be displayed in a plot (more on plotting in Sec. 4) with a trace representative of a flow. This yields an additional method of navigation, as well. A flow may intersect a node and continue on another node; the user should be able to "follow the flow" to the other node in addition to navigating the categorical structure directly, via edges.

2.6 Fuzzy dialectic

The Fichtean *dialectic* is the evolution of understanding. It is often represented as a position taken, a thesis; an alternative position taken, an antithesis (not necessarily in conflict with the thesis); and a sublation of the two to form a synthesis [9].[5] Fichte goes so far as to claim that every act of thinking is a synthesis [10], and so it is natural for an information architecture designed to enhance human thinking to express this model.

The crisp dialectical architecture includes a special type of flow to express the dialectic called the *thesis flow*, which also applies to the fuzzy dialectical architecture (and it is this aspect that is its namesake). A thesis flow is defined for each node and can be considered to be a user's description of the intersection of the categories defining the node. When another flow intersects a thesis flow, it is considered an antithesis flow to the thesis. A user would then be prompted to resolve these to form a newly informed thesis. But flow intersections are in fact not limited to thesis flows, so each intersecting flow is an antithesis to a given flow. This dialectical manner can have many instantiations; for instance, consider a thesis flow for the node $A \cap B$ (the relationship between A and B). Perhaps a user has written a document comprising this thesis flow, and then brings in a new quantitative data set such that the flow it defines intersects $A \cap B$. The thesis flow would then require the sublation of the thesis and the antithesis (data). In this way, when newly connected information is introduced to the information system, those flows that are affected can be immediately identified.

3 Algorithmic instantiation of the structure

A naïve approach to writing an algorithm to instantiate the fuzzy dialectical architecture would yield exponential computation time. In this section, we dis-

[5] See Ref. [9] for a discussion of the similarities and differences between the Fichtean and Hegelian dialectics.

cuss some salient ideas to consider when instantiating the architecture. A highly efficient algorithm for the structure remains an open problem, but progress has been made.

A key insight is that the entire structure need *not* be recomputed when a new atom is inserted or removed. This allows us to incrementally build a structure, which should, of course, be invariant to the order in which atoms are inserted. This is especially important for real-time applications such as robotics.

What requires recomputation when an atom is inserted? Only the relations originating at those nodes that are constructed by categories in associated with the new node need be recomputed. That is, (typically) most of the structure is untouched by the insertion of a new atom. Furthermore, the visibility or hiddenness of an atom never needs to be computed because an atom is visible in only one node, that which is defined by the intersection of all categories associated with it.

Moreover, any node that is *new* to the structure requires no structural computation, since all its relations must be vs-relations because no relation can possibly contain more than the others, since only one atom (the new one) is at the "bottom" of those paths. This allows extremely quick insertions for new categories and combinations of categories.

The unavoidably most computationally intensive aspect of the computation is the re-computation of metacategories for those nodes affected by the insertion of a new node. It is important to note that once a minimal metacategory has been found at a given level, no more levels are required.

It is also of note that memory resources can become an issue if the structure is maintained in memory (especially if metacategories are stored). It is advisable to use a graph database to persist and access the structure.

4 Human interfacing for the fuzzy architecture

As with any information architecture, the fuzzy dialectical architecture may have innumerable instantiations. In this section, we describe general guidelines for these instantiations and present a specific example in Sec. 4.1.

The user should be able to browse nodes like a traditional hierarchy.
The nodes represent the intersection of categories, as they typically do in a hierarchy or in tag-based browsing. The hierarchy has a long and illustrious history of value to human thinking [5]. Although the structure is, in fact, a graph, it will be natural to most users to experience it as a hierarchy. The "hierarchy" the user interacts with will be *organic* in the sense that it may change when new information is added to the system. All the spatial metaphors so valuable to hierarchies will be applicable, like "up" and "down," "in" and "out." At each node, the visible edges should be represented as single categories—the category that would be intersected with the current node to yield the lower node.

The user should be presented only visible edges.
"Information overload" has been identified as a significant challenge to our

information age [19,24]. One of the primary advantages of the dialectical architecture is that it minimizes the amount of information a user is presented at each node, much like a traditional hierarchy, which "tucks" the information that is further-categorized into lower levels. This means "hidden" atoms and edges should not be presented, explicitly (although exceptions can be made, of course). In some instances, hidden atoms, as defined above, might also be hidden from the user's view; however, caution is advisable here, since in some instances, the interface might call for their visibility.

The user should be able to browse "up" to any parent node.
The property of the architecture that the path order is invariant can be exploited to allow browsing the structure in a manner analogous to the hierarchical "up-one-level," but with multiple possibilities. The user can traverse "up" to any parent node, of which there may be several, unlike in the hierarchy, which allows each node to have only a single parent. This can be visualized by allowing the user to de-select any selected category along the path, and not merely the last-selected.

The user should be able to browse by following edges or flows.
Following edges is the *structural* method of navigating and is isomorphic to browsing traditional hierarchies. The dialectical architecture adds the ability to browse along *flows* as well. A flow can intersect a node for one or more consecutive atoms, then move to another node. For instance, an article may be discussing the intersection of several topics, then drill deeper into it with an additional categorization, which would lead it to a child node. This could be navigated by "going with the flow," such that the user continues to see the series of atoms that comprise the flow.

The user should be able to synthesize newly intersecting flows.
The dialectical aspect of the architecture requires the thesis–antithesis–synthesis structure of information development. An information attempting to enhance human thinking should certainly capture the development of that thinking, which this feature accomplishes. A flow can be "intersected" when another flow is coincident with a node the flow traverses, and this intersection may provide a new perspective to the original flow (antithesis). A user should be able to synthesize the two perspectives such that their information system remains well-curated.

The user should be able to view quantitative data in graphs.
With the inclusion of quantitative information, the fuzzy dialectical architecture should have a user interface that presents quantitative information in a concomitant manner, typically a graph. A data point (atom) that is visible at a given node may belong to a multivariate data set and belongs to the node with some membership value in the range $[0, 1]$. A two-dimensional graph of given data set intersecting a node is often the best option; the user's ability to change which variables are plotted on the abscissa and ordinate

axes is important. Data series should be connected and multiple series on the same graph should appear with different line properties or colors.[6]

The user should be presented the membership of an atom in a node.

The fuzziness of the architecture yields an interesting aspect of the information: the degree to which each atom belongs to a given node. For quantitative information, the membership value of each point in the node should be presented; we suggest opacity of the data point. For other types of information, several techniques are possible, including sorting, iconic differentiation, color, and opacity.

We now turn to an example instantiation for demonstration purposes.

4.1 A demonstration

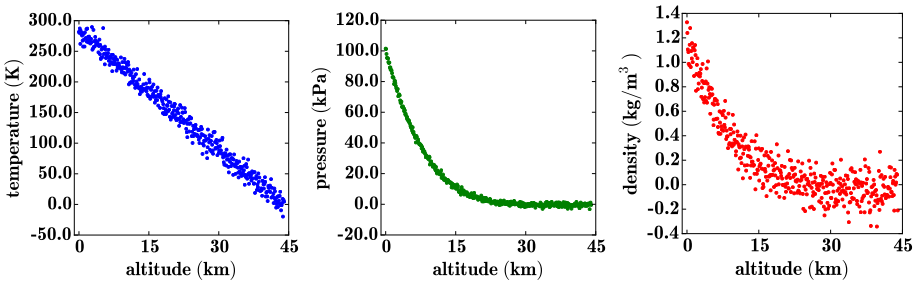
An exemplar set of data was generated for demonstrative purposes as if from sensors on a balloon deployed to measure atmospheric data at various altitudes. Each data point consists of four quantities: altitude, air temperature, air pressure, and air density. The Committee on Extension to the Standard Atmosphere (COESA) [1] has defined a mathematical model used here to synthesize sensor data. The data points were categorized using fuzzy set theory, parsed with an algorithm that computes the fuzzy dialectical structure, and presented to the user. This data set was chosen for demonstration purposes because changes in air properties with altitude are well-understood.

Simulated sensor data was generated in Python [18] using Scikit Aero [8]. An objective of this project is to analyze continuous streams of sensor data coming from scientific robots, so Python was chosen for compatibility with the robotics simulation environment MORSE [3]. Scikit Aero has the COESA standard atmosphere model. In order to simulate sensor variability, the generated data was randomized with a standard distribution appropriate to the type of data, the results of which are displayed in Figure 1.

The fuzzy architecture presents data based on relationships between categories. Quantitative information such as sensor data require preprocessing in order to be actionable by the algorithm. Fuzzy set theory was utilized to categorize the data. Three categories were defined for each variable by assigning membership functions. For simplicity all modifier categories are titled high, medium, and low for each variable. Assigning the original data membership in each category resulted in each measurement having 16 separate values (four variables plus membership values in 12 categories). All category membership functions are triangularly shaped and evenly divided across the data. The Python package Scikit Fuzzy [7] was employed to generate the membership functions and to assign a fuzzy membership value to each category for each data point.[7] Fuzzy membership functions are visualized in Figure 2.

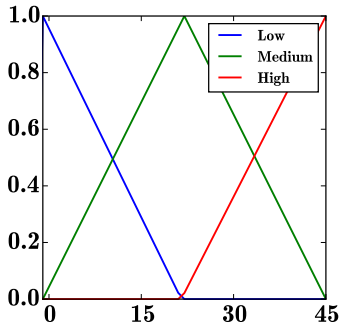
[6] We suggest a designer to make liberal use of the advice given by Tufte [22] for the visual display of quantitative information.

[7] While these categories are adequate for this demonstration, the Scikit Fuzzy package offers the flexibility to tune membership functions to more accurately align with user

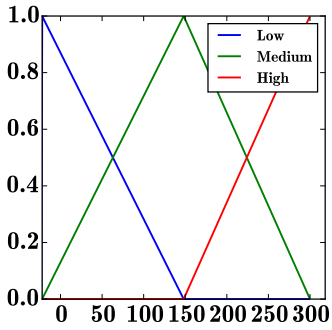


(a) (b) (c)

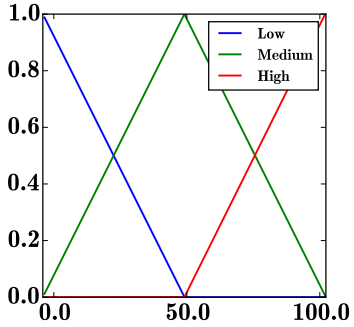
Fig. 1: simulated sensor data. Atmospheric data was generated from the COESA standard atmosphere model from 0 up to 44 km. Normal distributions simulating sensor variability applied to altitude, temperature, pressure and density variables with standard deviations 25 m, 10 K, 1 kPa, 0.1 kg/m^3.



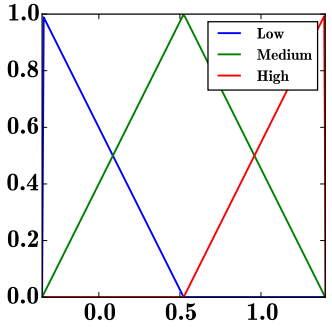
(a) altitude (km) membership functions



(b) temperature (K) membership functions



(c) pressure (kPa) membership functions



(d) density (kg/m^3) membership functions

Fig. 2: fuzzy membership functions. Three fuzzy categories for each variable are evenly distributed across the range of values for that variable.

User interface Although not demonstrative of every aspect of the fuzzy dialectical architecture, the user interface we new describe follows the guidelines described at the beginning of this section (4) to present the atmospheric data described above. We first consider the data presented at the node defined by the intersection of two categories: high-temperature and mid-pressure. Atoms having a strong association with both of these categories should be most strongly visible. A screen-capture of the information presented at this node are displayed in Figure 3a. The opacity of each data point is representative of its membership in the node. In this instantiation, all graphs present the altitude on the abscissa and the other variables on the ordinate.[8] This node from which we begin our description is displayed in the upper-right corner in green: two categories have been selected (`t_hi` intersect `p_md`) and can be deselected by clicking the "x." This node has available to it the edge traversals described by the categories in purple. Figure 3b shows what the user is displayed when browsing the node `t_hi` intersect `p_md` intersect `rho_hi`. Note how the display has changed such that only the data most strongly associated with these three nodes is displayed. Finally, Figure 3c shows the result of an "up" traversal performed by deselecting the `p_md` category.

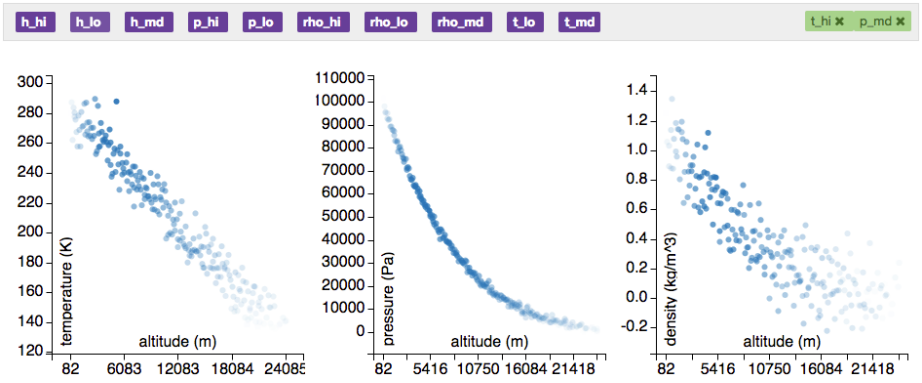
Implementation considerations To review, sensor data can be processed by a fuzzy categorization engine, written to a database, read by the fuzzy dialectical algorithm, and displayed by a user interface, as shown in Figure 4. In this instantiation, simulated measurements and their corresponding membership values were written to a PostgreSQL database [20] for later retrieval by the fuzzy dialectical algorithm. The use of the database separates categorization from analysis and presentation. This modular hierarchy of simulation, categorization, storage, and retrieval was chosen because it enhances the resiliency and flexibility of the system. Each module can be run on a different physical system at a different geographical location depending on the requirements of the individual implementation. Once continuous streams of data are categorized and stored, they are no longer time-sensitive, and can be batch-processed by the fuzzy architecture algorithm.

The fuzzy dialectical algorithm was written in Ruby [23, v 2.4.0] and powers the user interface, which was built using the Pakyow [12, v 0.11] web application framework, which features graphs generated by c3.js [11, v 0.4.11]. Future implementations will include flow visualization and traversal disparate data types (this instantiation shows only quantitative information, but this is incidental and not an inherent limitation).

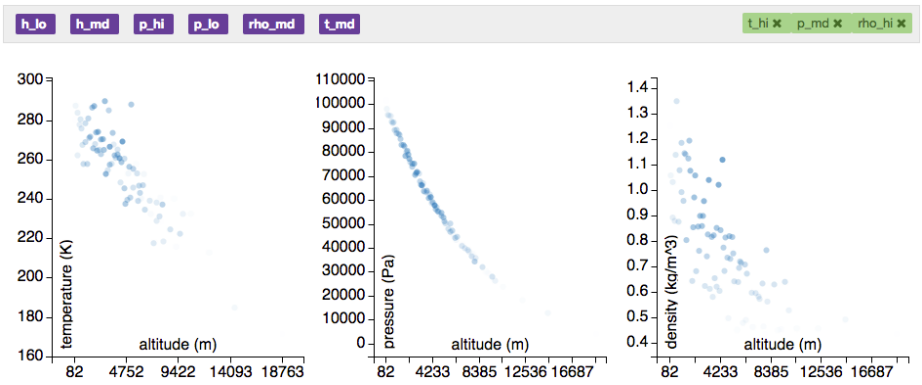
---

selected categories. A user would have the ability to quickly and intuitively define categories based on data type and origin.

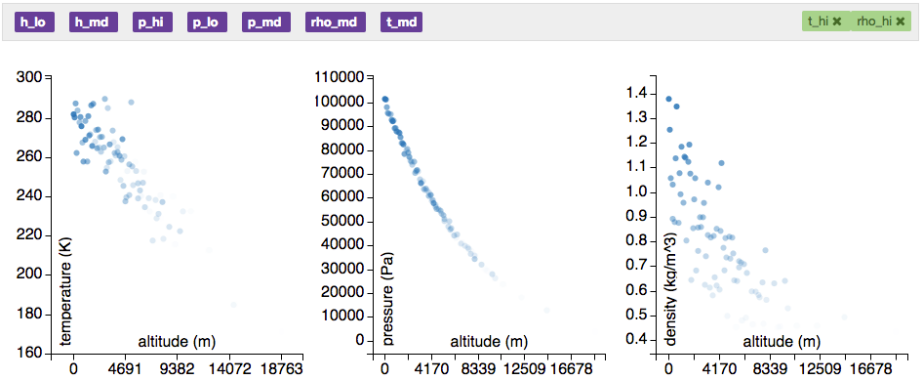
[8] In future instantiations, the authors envision allowing the user to select which variable is plotted on each axis.



(a) user view at node `t_ui` intersect `p_md`



(b) user view at `t_ui` intersect `p_md` intersect `rho_hi`



(c) user view at `t_ui` intersect `rho_hi`

Fig. 3: The user interface for the demonstration of the fuzzy dialectical architecture. In Figure 3a, the user begins at node `t_ui` intersect `p_md`. In Figure 3b, the user has traversed "down" to node `t_ui` intersect `p_md` intersect `rho_hi`. In Figure 3c, the user has traversed "up" to node `t_ui` intersect `rho_hi`. Opacity is a function of the data point's membership in the node.

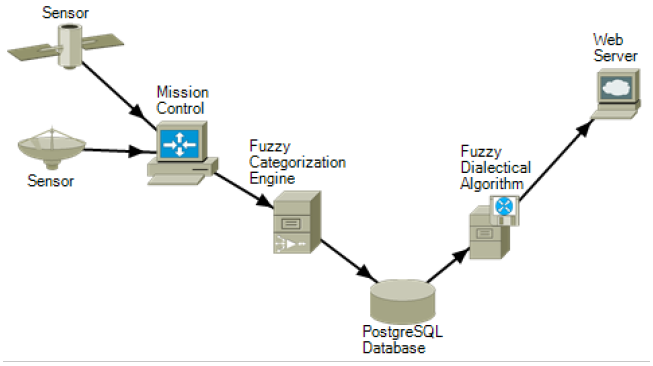


Fig. 4: a data flow diagram showing how sensors provide data that can be processed by a fuzzy set theory categorization algorithm and stored in a database to be retrieved by the fuzzy dialectical algorithm and displayed to the user.

5 Conclusion and prospects

What emerges from the information architecture presented herein is a framework for fusing information structure, category, sequence, and dialectic; fusing the representation of complex information with simple interface; and fusing information that is quantitative with that which is qualitative. It is an architecture designed for information integration and intelligence amplification. Applications include many situations for which traditional architectures have been effective, yet restrictive. The integration of quantitative information has many application, but the authors are developing a human-robot interface based especially on this aspect of the architecture.

We have presented the methods of fuzzification of the dialectical information architecture, algorithmic considerations, human interfacing, and a specific demonstration of the architecture. The methods employed were motivated by concepts from neuroscience, psychology, artificial intelligence, and philosophy. This presentation has been primarily synthetic and lays the groundwork for analytical investigation.

References

1. U.S. standard atmosphere, 1976. Tech. rep., National Oceanic and Atmospheric Administration and National Aeronautics and Space Administration and United States Air Force (February 1976)
2. Bondy, A., Murty, U.: Graph Theory. Graduate Texts in Mathematics, Springer London (2011)
3. Echeverria, G., Lemaignan, S., Degroote, A., Lacroix, S., Karg, M., Koch, P., Lesire, C., Stinckwich, S.: Simulating complex robotic scenarios with morse. In: SIMPAR. pp. 197–208 (2012), http://morse.openrobots.org

4. Edelman, G.M.: Neural Darwinism: The theory of neuronal group selection. Basic Books (1987)
5. Garrett, J.: Elements of User Experience,The: User-Centered Design for the Web and Beyond. Voices That Matter, Pearson Education (2010)
6. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. 31(3), 264–323 (Sep 1999), http://doi.acm.org/10.1145/331499.331504
7. Joshua Warner: Scikit-fuzzy: A fuzzy logic toolbox for scipy, http://pythonhosted.org/scikit-fuzzy/
8. Juan Luis Cano: scikit-aero: Aeronautical engineering calculations in python, https://github.com/AeroPython/scikit-aero
9. Kaufmann, W.: Hegel: A Reinterpretation. A Doubleday Anchor book, Doubleday (1966)
10. Kroeger, A.E.: The difference between the dialectic method of hegel and the synthetic method of kant and fichte. The Journal of Speculative Philosophy 6(2), 184–187 (1872), http://www.jstor.org/stable/25665792
11. Masayuki Tanaka: C3.js: D3-based reusable chart library, http://c3js.org
12. Metabahn: Pakyow: A realtime web framework for Ruby, https://www.pakyow.org
13. de Oliveira, J., Pedrycz, W.: Advances in Fuzzy Clustering and its Applications. Wiley (2007), https://books.google.com/books?id=Pn0e1xm4YBgC
14. Pfeifer, R., Bongard, J.: How the Body Shapes the Way We Think: A New View of Intelligence. MIT Press (2006), https://books.google.com/books?id=EHPMv9MfgWwC
15. Picone, R.A., Powell, B.: A new information architecture: A synthesis of structure, flow, and dialectic. In: Yamamoto, S. (ed.) Human Interface and the Management of Information. Information and Knowledge Design, Lecture Notes in Computer Science, vol. 9172, pp. 320–331. Springer International Publishing (2015), http://dx.doi.org/10.1007/978-3-319-20612-7_31
16. Rosenfield, I.: The Invention of Memory: A New View of the Brain. Basic Books (1988), https://books.google.com/books?id=5e_aAAAAMAAJ
17. Ross, T.: Fuzzy Logic with Engineering Applications. Wiley, third edn. (2011)
18. Rossum, G.: Python reference manual. Tech. rep., Amsterdam, The Netherlands, The Netherlands (1995)
19. Strother, J.B., Ulijn, J.M., Fazal, Z.: Information Overload: An International Challenge for Professional Engineers and Technical Communicators. No. ISBN 9781118360491, Wiley-IEEE Press (2012), http://ieeexplore.ieee.org/servlet/opac?bknumber=6354045
20. The PostgreSQL Global Development Group: Postgresql, https://www.postgresql.org/docs/9.6/static/index.html
21. Trudeau, R.: Introduction to Graph Theory. Dover Books on Mathematics, Dover Publications (2013)
22. Tufte, E.: The Visual Display of Quantitative Information. Graphics Press (2001), https://books.google.com/books?id=GTd5oQEACAAJ
23. Yukihiro Matsumoto: Ruby, https://ruby-lang.org
24. Zeldes, N., Sward, D., Louchheim, S.: Infomania: Why we can't afford to ignore it any longer. First Monday 12(8) (August 2007), http://firstmonday.org/ojs/index.php/fm/article/view/1973/1848
25. Zimmermann, H.: Fuzzy Set Theory—and Its Applications. Springer Netherlands (2001)