What is the most fundamental element of knowledge?

By
Ashraf Azmi
Datumtron
azmi@datumtron.com
March 2014

The way data is represented in memory is the foundation of any data processing system. Currently, Relational database representation is the most widely used. It is based on representing data into Tables, Columns, Rows, and Relations among tables. Other database schemes include Network and Hierarchical based structures.

Knowledge Representation (KR) schemes add an inference process to the data representation. Inference allows discovering new knowledge from the existing knowledge. For example; if we have the fact “All birds can fly” and also the fact that “Eagles are birds”, then we can infer that “Eagles can fly” even though this fact is not explicitly given.

Examples of KR schemes are Production Systems, Semantic Networks, Frames, and Predicate Logic. Existing KR schemes use variety of elements; none of these schemes is based on a single abstract element.

This leads to the question; is it possible to find a single fundamental element of knowledge? Furthermore, for these fundamental elements to form any larger structure, they must be able to affect (or relate) to one another. Then the second question becomes: Is it also possible to find a single fundamental relationship among these elements that allows us to build a full Knowledge Representation scheme.

The Datum Universe.

The Datum Universe is a new representation that answers the above questions. Let’s consider an example from the relational representation and discuss how it is represented in the Datum Universe.

In the following illustration, information about Toys is represented as a table with three columns; Toy, Shape, and Color, and three rows for T1, T2, and T3. Each cell contains a value of each column corresponding to each row.

Toy	Shape	Color
T1	Box	Red
T2	Box	Red
T3	Sphere	Red

Table 1: Toys data table

In the new data representation, the Toy data is shown in the following figure.

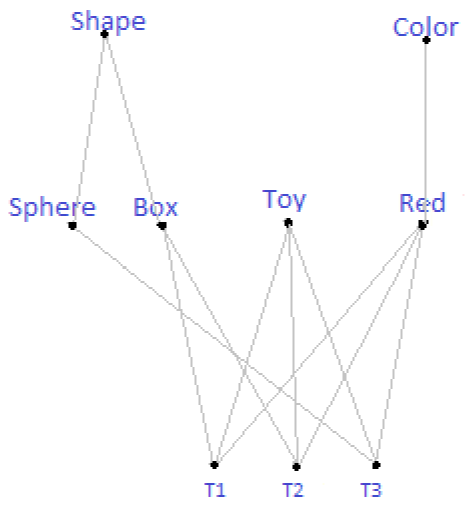

Figure 1: Toys Table converted to Datum Universe graph

In this graph, we have one type of elements (represented by a dot) and one type of relations (represented by a line between two dots). Shape, Toy, Color, Sphere, Box, Red, T1, T2, and T3 are all equal elements – no classification into Column, Row, or Value. The closest Knowledge Representation Scheme to this is Semantic Networks. However, Semantic Networks rely on many types of relations like ISA, Contains, Brother Of, Color Of, etc. This is a more fundamental form of Semantic Networks because it allows only one type of links.

The name of these abstract elements of data is the singular form of Data; Datum. "Datum" is not widely used in computer science because a singular element of data was not well defined. As for the single abstract relation between Datums, the name is also well known in the language; "is". It just indicates that there is a directed link form one datum to another. A similar relation is known in AI and Object Oriented Representations which is the "ISA" relation. The "ISA" relation facilitates class hierarchies and property inheritance. The "is" relation is a more general form of the "ISA". In the above example, we can say "T1 is Red" and "T1 is Toy". However, for the traditional "ISA" relation, we can only say "T1 ISA Toy" but not "T1 ISA Red".

In this example, we say that "T1 is Red" and "Red is Color". Similarly, we can Say that "T1 is Toy" and "Toy is (some more abstract Datum)" and so on. The "is" relation is not reversible i.e. "T1 is Red" is not the same as "Red is T1".Therefore, the "is" links are directed in the up direction but we are omitting the arrows for simplicity.

By using datum elements and "is" links, we are able to represent the Toy relational table. Any additional columns and rows in the table are just additional datums. The manner in which they are connected is what gives each datum its meaning. There is no built-in understanding of Columns, rows, tables, or primary keys. All are equal datum elements that inherit attributes and meaning just by their connections to other datums.

To show that we can represent everything that a relational database can, we must also show representing relations among tables. Consider adding the following Color table which is related to the Toy Table via the Color column.

Color	R	G	B
Red	255	0	0
Green	0	255	0
Blue	0	0	255

Table 2: Color data table

This Color table gives us more information about the values of the Color Column. The Color column in this table is considered the primary key while the Color column in the Toy table is considered a foreign key. To represent this relation in the Datum Universe representation, we simply extend the graph as in the following figure (only showing Red):

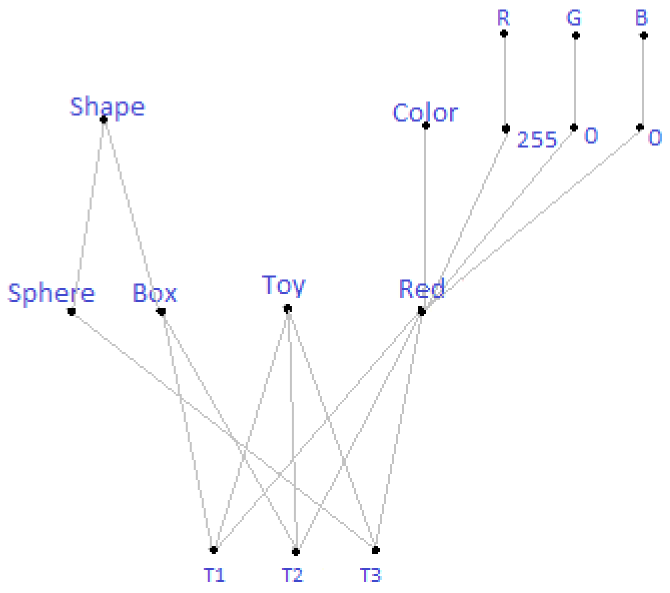

Figure 2: Color Data added to the Datum Universe graph

The graph can be extended in the same manner to represent multiple relations. This means that using the single datum element and the single "is" relation, we are able to build up graphs that have the same representational power as relational database schemes.

The graph presented so far is a Directed Acyclic Graph (DAG) in graph theoretic terms. This means that the links don't form cycles – hence no circular definitions and all links have a direction, in our case, the direction is up. Mathematically, this is a Partially Ordered Set where the elements of the set are "datums" ordered by the binary relation "is".

Inference.

You may have noticed (from the table or the graph) that all of our toys are red. This fact is not explicitly stated in the table or the graph representations. However, in Figure 1, we see links going from T1, T2, and T3 to both Toy and Red. The structure where links go from all the members of one set of nodes to all the members of another set of nodes is called a biclique graph. Whenever a biclique is created, we can simplify it by “inducing” a new node in the middle and re-wiring the links. This is shown in the following two figures

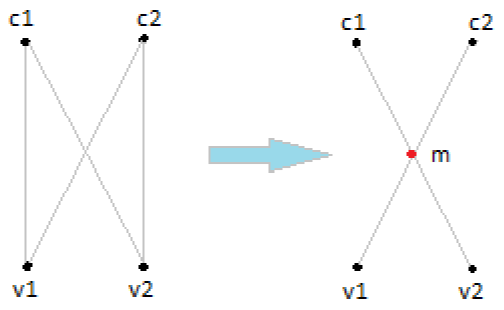

Figure 3: Reducing a biclique of 2x2 links to a star graph of 2+2 links

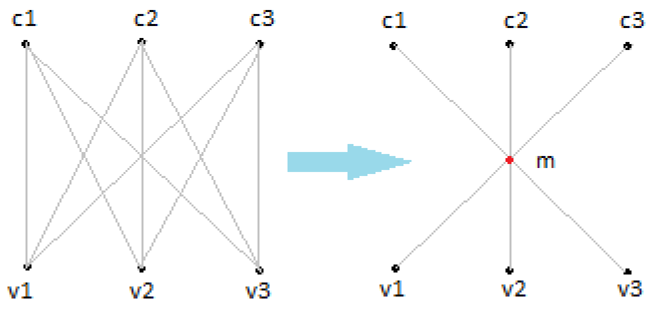

Figure 4: Reducing a 3x3 biclique to 3+3 star.

Even though, we added one more node to the graph, the graph size (in graph theory) is reduced. This is because graph size is measured by the number of links not the number of nodes. The graph size is proportional to the memory size required to store the graph. This means that the introduction of an intermediate node reduces the amount of memory needed to store the biclique graph information content. This is shown by the following formulas:

$$|G| = n*m \qquad (1)$$

$$|G_i| = n + m \qquad (2)$$

$$\Delta M = (n*m) - (n + m) \qquad (3)$$

Where n is the number of Datums in the upper set and m is the number of Datums in the lower set. $|G|$ and $|G_i|$ are the sizes of the sub graphs before and after induction, and ΔM is the memory savings.

This has important consequences but it is out of the scope of this document. We call this process an "induction" process.

To return to our example, figures 5 and 6, show the effects of the induction process on our graph. The left graph in Figure 5 is the same as the graph of Figure 1 with the biclique links marked with red.

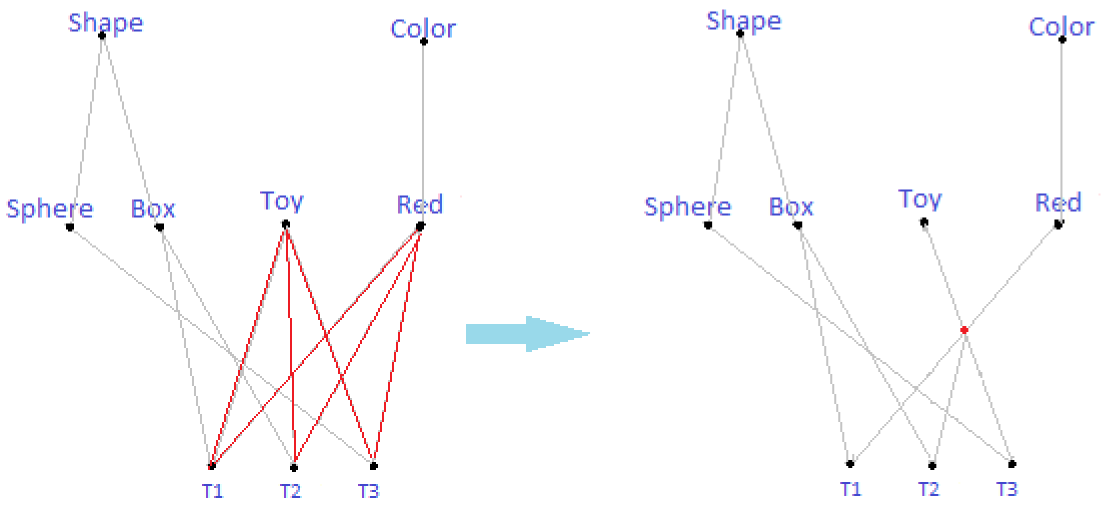

Figure 5: Induction of RedToy datum (before and after)

First, we simplified this biclique which has the upper set {Red, Toy} and the lower set {T1, T2, T3} by inducing a new datum in the middle and rewiring. We can call this datum "RedToy" where:

"RedToy is Toy", "RedToy is Red", "T1 is RedToy", "T2 is RedToy" and "T3 is RedToy"

Figure 5 still has one more biclique, which has the upper set {Box, RedToy} and the lower set {T1, T2}. This is shown in the left graph of Figure 6. Again, we induce a new datum in the middle and rewire as in the right side of Figure 6.

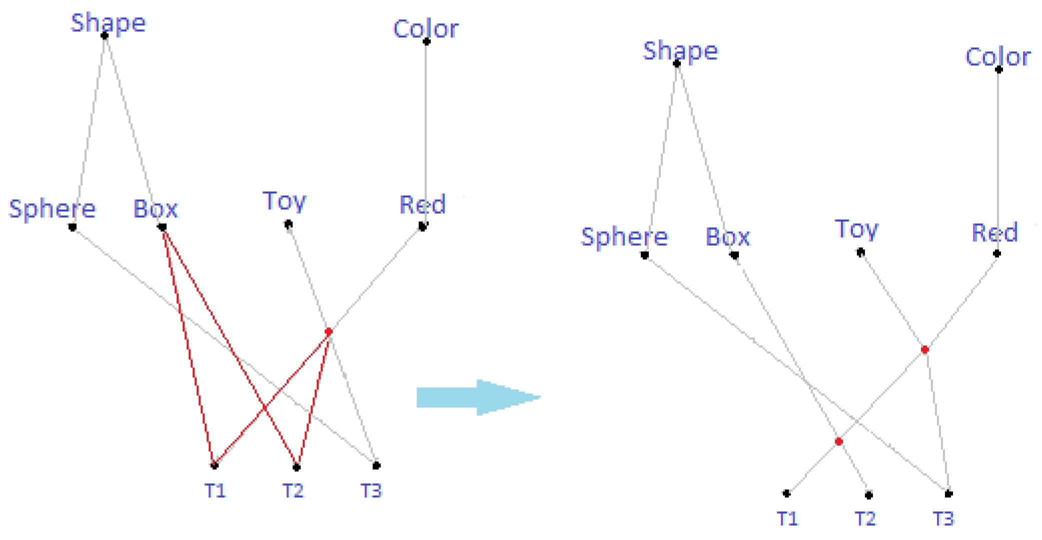

Figure 6: Induction of RedBoxToy datum (before and after).

We can call this datum “RedBoxToy” where:

“RedBoxToy is Box”, “RedBoxToy is RedToy”, “T1 is RedBoxToy” and “T2 is RedBoxToy”.

The new datums explicitly identify two new “classes” of toys one that has a red color and one that has both red color and box shape. This is a classification process. It introduces order (consumes entropy) in the Datum Universe.

Notice that the names “RedToy” and “RedBoxToy” are just names that we give for the purpose of our discussion but they do not (and need not) exist in the actual database.

Objects.

Datums have no content of their own; their existence is totally dependent on, and defined by, their relations to other Datums. Indeed all the induced Datums have nothing more than their connections.

However, our computers have built in native knowledge about zeroes and ones, strings, doubles, dates, times, images, files, and last but not least, executable code. These built in Datums are represented by their native representation and are attachable to other Datums. For example, the string “Red” , “Color”, “Toy”, “T1”, etc., are attached to the Red datum, Color datum, Toy datum, T1 datum, respectively. The attachment is such that there is an “is” relation between Red and the string “Red”. For a numerical datum like 123, the bit representation of 123 is attached, and similarly for other native representations including executable code. Each datum may have at most one attached object.

Hierarchy.

Let’s consider the relation “d1 is d2” d2 is called a Class of d1 and d1 is called a Value of d2. For example, in “Red is Color” Color is a Class of Red and Red is a Value of Color. We can also refer to the Class as a Parent and the Value as a Child when discussing the hierarchy. A datum inherits all the relations of all its parents. Parents with attached objects form a barrier for the propagation of inheritance. For example, T1 inherits from RedToy. Since RedToy doesn't have any attached objects, then the inheritance continues to Red. Since Red has an attached object, the inheritance stops here and doesn't go on to Color and up. This way we can say that T1 is Red and not T1 is Color. That doesn't mean that we can't jump over one or more inheritance barriers if we need to.

Since each datum is completely defined by its relations, a datum must have at least two parents. The only datum that has no parents is the root datum which we can call "Datum". If a datum has only one parent, then it can't be differentiated from that parent. In this case, this datum is "fused" with its parent and all of its children are linked directly to that parent.

Notice that any attached object counts as one parent. So any datum that has an attached object like Red may have at least one parent.

Transitive Reduction.

If we have the links "d1 is d2" and "d2 is d3", then we can't have the link "d1 is d3". This relation is implied by transition (inheritance) and would be redundant if it existed explicitly. If the graph has "d1 is d2" and "d1 is d3" and "d2 is d3" is being added, then "d1 is d3" must be removed. The Datum Universe is a Transitive Reduction Directed Acyclic graph in graph theoretical terms.

Patterns and Prediction.

Now let's consider the induced Datum that we called RedToy. The number of leafs under RedToy is 3 leafs; T1, T2, and T3. The number of leafs under Toy is also 3. We can conclude that 3 out of 3 toys are RedToy and since "RedToy is Red", then we can conclude that 100% of Toys are Red. The number of leafs is called the "Support" of the datum. The ratio of a datum's support to one of its parent's support is called "Generalization Confidence" or just "Confidence" with respect to the specific parent.

The fact that all Toys are red is now encapsulated in the datum RedToy. If we need to predict a color for a toy, we can predict Red with confidence 100%. Of course this is not very reliable because the Support is so small. If we have seen a thousand toys and all are red, then our prediction of Red will be more reliable. Now if we see a new Toy T4 that is Blue, it will be added under Toy and Blue. This will modify our belief that 100% of Toys are Red to 75% (3/4) of Toys are Red.

If we need to predict the color of a new Toy T5, we can predict Red with 75% confidence. Our predictions will change all the time as new data is added and new classes are induced and existing classes change support levels. This is an advantage over existing data mining technology which operates on a snap shot of the database and must be rerun when data changes.

Conclusion.

The Datum Universe is similar to database schemas in having a low level and efficient representation of data but has the advantage of incorporating a native inference process. This representation treats data and patterns in the same way – as Datums. Traditional ad-hoc data mining systems have to run on snap shots of traditional databases. The data mining process in the Datum Universe is continuous – as more data are added, new classes emerge and new patterns are identified.

Compared with traditional Knowledge Representation Schemes, like Semantic Networks, Production Systems and Frames, the Datum Universe, provides a low level data representation. This means it can be used for more business applications. The minimal predefined data structure and inference lends itself to building up more complex intelligence.

By finding the most fundamental elements of knowledge with the most abstract way to relate them, we can study the data and knowledge content in rigorous mathematical, graph theoretical manner. We can utilize recursion to develop more complex general inference and reasoning logic; datums can manipulate other datums based on their relations only, i.e., more complex inference can be built on standard structures of datum connections.

The Predict software is built to show case the Datum Universe representation and inference engine. Predict reads table oriented data, builds the Datum Universe, and induces classes that satisfy a minimum support provided by the user. The Predict software provides a graphical view of the Data Universe graph, showing the induced datums, along with their Support and relative Confidence. The user can select any of the original table columns as a "Goal". The user can instruct Predict to extract datums that represents features of this Goal column. These features can be written to an XML file that can be used to develop custom prediction logic. Predict also let the user customize the built-in prediction parameters and run on test data files or production data files. When Predict runs on a test data file, results of the prediction is compared against actual values and a number of statistics are presented to the user. This can help the user to fine tune the prediction process, based on the nature of given data.

Final Observation.

Finding the most fundamental element of knowledge and the corresponding representation, may help with understanding other fields.   For example, we can see an interesting analogy with the brain. Where neurons would be datum place holders and neural connections would be the "is" links between datums. The brain stores memory by establishing connections between its neurons. This gives each neuron its purpose. The brain's neural network is analogues to the Datum Universe. However, it remains to be seen if the induction process has a biological analogues. An Induction process may have evolved to maximize the use of the limited physical neural connections resources. This allows the brain to store more data in the same physical space.