# Fuzzy Relational Visualization for Decision Support

**Brian Zier and Atsushi Inoue** [*]
Eastern Washington University
Cheney, WA 99004 USA

**Abstract**

A study on fuzzy relational visualization in system development aspects is presented. The front-end enables dynamic and scalable changes in visualization according to user's expertise and inspiration. Integrative management of various data and knowledge is handled by the back-end at any scale in cloud computing environment. Extended Logic Programming is used as the core of fuzzy relational management in the back-end, and is capable of consistent uncertainty management among probabilistic reasoning and fuzzy logic while maintaining asymptotically equivalent run-time with the ordinary Logic Programming. A multi-view relational visualization is being implemented and important graphical features are highlighted in this paper.

**Keywords:** Visualization, Probabilistic Reasoning, Fuzzy Logic, Logic Programming.

## Introduction

Given the rapid advancement and penetration of information technologies, visualization becomes more significant in many domains. In sciences, this is utilized in many aspects such as populations, evolutions, radiations, transformations and structures (Wattenberg and Kriss 2006). In business, this is often found instrumental in various decision making, e.g. sales charts, change of markets and customer relational charts. In mathematics, modern education demands visualization as an essential element, e.g. demonstration of three dimensional functions as free surfaces. In engineering, this is a very critical, mandatory tool for system design, e.g. fluid analysis for nuclear power plants, aerodynamics for aircraft and heat radiation for CPU units to list a few.

Unfortunately, the majority of conventional visualization tends to be application-specific and its analytical model is rather static in terms of their relational representation and visualization configuration. For example, MS Excel limits its visualization within limited dimensions (1, 2 or 3), its analytical model is limited to statistical, and it only accepts tabular data. While this indeed serves in many applications, there is a fatal limit in critical decision making support, e.g. infrastructure assurance where integrated leverage of knowledge concerning policies and factual (sensory) data is essential (Inoue 2010).

On the other hand, many studies indicate effectiveness of sharing various visualization among a group in order to study extensive exploitation and in-depth understanding of data sets (Heer 2006; Heer, Viegas, and Wattenberg 2007; Viegas et al. 2007; Wattenberg and Kriss 2006). In this framework, group consensus is made as a result of sharing different interpretations through various visualizations. This suggests necessity of a general visualization platform that is capable of visualizing subjects with a dynamic and scalable change of configuration (Shneiderman 1996) via interaction with users (Shneiderman 1998; Zhang 2008).

Our general application framework for intelligent systems deploys Extended Logic Programming (ELP) and a multi-view visualization scheme, and its efficiency and effectiveness have been demonstrated throughout a showcase of various applications (Springer, Henry, and Inoue 2009). In this paper, we discuss the system development of fuzzy relational visualization for decision support within this application framework. First, the specification and progress are reported. Then the management of fuzzy relations (back-end) and their visualization scheme (front-end) are described respectively.

## Specifications and Progress

The ultimate goal of this development is a fuzzy relational visualization with the following general specification:

**S1.** Various relations are dynamically visualized in various aspects.

**S2.** Various types of data are visualized, e.g. tabular, texts, images, multimedia streams, diagrams, etc.

**S3.** Relations can be uncertain, i.e. probabilistic, possibilistic, and perceptual (subjective).

**S4.** Knowledge and data are integratively managed throughout a canonical representation and process.

**S5.** This is scalable in cloud computing environment.

In this specification, the first two (S1 and S2) are considered as matters of the front-end that provide graphical interfaces and interactions with users, and the rest (S3, S4 and S5) as matters of the back-end that manage relations among all data and knowledge.

Two major works on the front-end are visualization scheme and human-computer interaction. There are two

[*]E-mail: `inoueatsushij@gmail.com`

major progresses on the visualization scheme: first, the Logic Programming (LP) visualizer for educational purpose (Henry and Inoue 2007), then the visualization scheme for reasoning under uncertainty (Springer and Inoue 2009). For this fuzzy relational visualization, we further specify this scheme in order to realize dynamic and scalable visualization as a result of developing an interactive graphic interface. Studies on more sophisticated human-computer interaction, including integration of this relational visualization scheme with various conventional visualization, e.g. geographical, spacial, statistical and functional, are planned and upcoming.

The back-end consists of ELP and extraction functionalities, and they are placed in a cloud computing environment. ELP is developed with Support Logic Programming (SLP) (Baldwin, Martin, and Pilsworth 1995) and a simple extension of fuzzy probability (Inoue 2008). Extraction functionalities consists of translators from various types of data into ELP and utilities to manage massiveness and high dimensions (Codd 1970; Nugues 2006; Moore and Inoue 2008; Yager 1982). Parallelization of ELP in a cloud computing environment is currently underway (Joxan and Maher 1994).

## Management of Fuzzy Relations: Back-End

This section describes management of fuzzy relations from computational aspects: representation and query processing of ELP, as well as how various types of data are extracted and translated into this representation, i.e. extended Horn clauses.

### Representation

Table 1 shows how fuzzy relations can be represented in ELP. Like ordinary LP, Horn clauses are used to represent relations in general. Two extensions are made in those Horn clauses. One is the various probability annotation such as a point (e.g. 0.62), an interval (e.g. $[0.6, 0.68]$) and a fuzzy (i.e. linguistic) (e.g. 'very_low' and 'high'). The interpretation of those annotated Horn clauses, i.e. probabilistic events, is $P(h) \in [0, 1]$, where $P$ is the annotated probability and $h$ is the Horn clause. We interpret $P(h) = 1$ when no probability is annotated. The other is use of fuzzy predicates such as *'Tall'*, *'BigFeet'*, *'ProceedAtPace'* and *'Level'* in this table. This is simply a matter of fuzzy predicates observed in the Horn clause $h$, and such predicates are specially processed in their unification.

Fuzzy probability is formally defined as a normal, convex fuzzy set defined over interval $[0, 1]$ (i.e. a fuzzy number), s.t. $\mu_p(x \in [0, 1])$. In addition, a linguistic label is associated with such a fuzzy set for our advantage, i.e. the linguistic extension of annotated probability in ELP.

Fuzzy predicate is formally expressed s.t. $p_f(x_1, \ \ldots, \ x_n)$ and its semantics is determined by a corresponding fuzzy set s.t. $\mu_{p_f}(x_1, \ \ldots, \ x_n)$, where $(x_1, \ \ldots, \ x_n) \in U$, the universe of discourse for this fuzzy set. Truth values of such a predicate, by its nature, are partial, i.e. $\tau \in [0, 1]$ where $\tau = 0$ corresponds to false and $\tau = 1$ corresponds to true, as well as other values

Table 1: Fuzzy Relations in ELP

| **Probability** | **Event** | |
|---|---|---|
| | ***Crisp*** | ***Fuzzy*** |
| ***None*** | Human( Kyle ) ← Student( Kyle ) | *Tall*( Kyle ) |
| ***Point*** | Awake( Kyle ) ← Healthy( Kyle ): 0.62 | BuysTallPants( Kyle ) ← *Tall*( Kyle ): 0.4 |
| ***Interval*** | Awake( Kyle ) ← Healthy( Kyle ), Weekend(): [0.6, 0.68] | *Tall*( Kyle ) ← *BigFeet*( Kyle ), TallerThanObserver( Kyle ): [0.8, 0.92] |
| ***Fuzzy*** | LightningStrike( Kyle ): *very_low* | *ProceedAtPace*( vehicle ) ← *Level*( terrain ), *HighTraction*() : *high* |

in-between correspond to partial truth. Currently we do not consider cases that fuzzy terms (i.e. fuzzy sets) appear in its arguments. They are rather unorthodox in Fuzzy Logic framework and, if necessary, can be translated into fuzzy predicates $p_{f_i}(x'_i)$, where $f_i$ represents the $i$-th fuzzy term appearing in the arguments, to be properly inserted into the original Horn clause.

### Query Processing

The most critical advantage of ELP is its computational efficiency, that is asymptotically equivalent with that of LP while the extensions of uncertainty management are indeed in a part of its computation. Consider the following simple extended Horn clauses, with query $a$ and unification of some fuzzy predicates such as $a$ and $a'$ as well as $c$ and $c'$, in order to demonstrate this efficiency:

h1: $a \leftarrow b \wedge c \wedge d : p1$
h2: $a \leftarrow e : p2$
h3: $c \leftarrow e \wedge f : p3$
h4: $c \leftarrow d : p4$
h5: $a' \leftarrow c' \wedge d : p5$
h6: $b : p6$
h7: $d : p7$
h8: $e$
h9: $f$

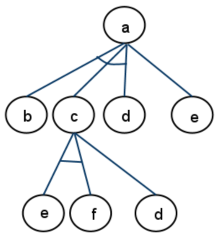

Figure 1: Snapshot of processing query $a$ in LP

First, we consider ordinary LP in order to process query $a$ with the assumption of all annotated probabilities $p1$, ..., $p7$ to be 1 (i.e. the equivalence of no annotations). Figure 1 indicates the snapshot of this query processing in an AND-OR tree. In general, the query processing in LP is Depth-First Search starting from node $a$. In LP, we only consider symbolic unification so that there is no partial unification such as $a$ and $a'$, as well as $c$ and $c'$. Further, we only need one Horn clause to be proven true (so-called an *existential query*) – either h1 or h2 (together with either h3 or h4 in order to prove sub-query $c$) in this query. Sometimes, we need to prove all possible cases (so-called a *universal query*), i.e. both h1 and h2 (together with both h3 and h4) in this query. The selection between existential and universal queries depends on applications. LP assumes *close world assumption* (i.e. negation as failure). Since recent knowledge representation technologies often deploy *open world assumption* such as Web Ontology Language (OWL), this is often considered as a shortcoming.

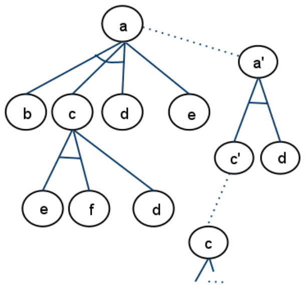

Figure 2: Snapshot of processing query $a$ in ELP

Query processing in ELP remains in Depth-First Search starting from node $a$ as indicated in Figure 2. In order to process query $a$ in ELP, we need to disjunctively combine all partial truth of h1 and h2, as well as h5 (i.e. the case of fuzzy predicates) s.t. $P_{h1}(a) \cup P_{h2}(a) \cup P_{h5}(a)$. Therefore, all Horn clauses need to be proven, i.e. the equivalence of the universal query in LP.

The partial truth in ELP is represented as a probability: either one of point, interval and fuzzy, and this is computed according to Jeffreys' rule (Jeffrey 1965) s.t. $P(c) = P(c|h) \cdot P'(h) + P(c|\neg h) \cdot P'(\neg h)$. Therefore, proof by satisfying sub-queries in ELP is a matter of computing such probabilities in a chain reaction. Let a conditional Horn clause be $c \leftarrow h : p = P(c|h)$ and the result of a sub-query (or simply a fact) be $h : p' = P'(h)$. Then the partial truth of query $c$, i.e. $P(c)$, is computed depending on the type of annotated probability according to Jeffreys' rule as shown in Table 2. Note that $P(c|\neg h) = 0$ (i.e. false) is expected for close world assumption (i.e. negation as failure) and $P(c|\neg h) = [0, 1]$ (i.e. unknown) for open world assumption.

Table 2: Partial truth of query $c$, i.e. $P(c)$

| Probability | negation as failure | open world assumption |
|---|---|---|
| Point | $p \cdot p'$ | $[p \cdot p',\ p \cdot p' + 1 - p']$ |
| Interval: $p = [l, u]$ | $[l \cdot l',\ u \cdot u']$ | $[l \cdot l',\ u \cdot u' + 1 - u']$ |
| Fuzzy (trapezoidal) core of $\mu_p$: $[lc,\ uc]$ support of $\mu_p$: $[ls,\ us]$ | core: $[lc \cdot lc',\ uc \cdot uc']$ support: $[ls \cdot ls',\ us \cdot us']$ | core: $[lc \cdot lc',\ uc \cdot uc' + 1 - uc']$ support: $[ls \cdot ls',\ us \cdot us' + 1 - us']$ |

When $h$ in the conditional Horn clause $c \leftarrow h : p$ consists of multiple predicates, s.t. $h = h_1 \wedge \ldots \wedge h_n$, we compute $p' = P'(h) = P(h_1) \cdot \ldots \cdot P(h_n)$.

Partial truth between fuzzy predicates $f$ and $f'$ (e.g. dotted lines between $a$ and $a'$, and between $c$ and $c'$ in Figure 2) is determined by applying Mass Assignment Theory, the conditional mass assignment $m_{f|f'}$ that yields an interval of probability[1] (Baldwin, Martin, and Pilsworth 1995). This is so-called *semantic unification* as opposed to symbolic unification. Importantly this is not symmetric unlike symbolic unification, i.e. $m_{f|f'} \neq m_{f'|f}$. In the query processing aspect of ELP, this is considered as insertion of Horn clauses $f \leftarrow f' : p$ and $f \leftarrow \neg f' : \bar{p}$, where $p = P(f|f') = m_{f|f'}$ and $\bar{p} = P(f|\neg f') = m_{f|\neg f'}$. Note that neither close world assumption nor open world assumption holds in any query process with semantic unification. This is indeed consistent with Fuzzy Logic.

Computing partial truth adds a few simple arithmetic to unification as shown in Table 2. While this may increase a coefficient of run-time, its asymptotic complexity still remains the same. Similarly to semantic unification in comparison with symbolic unification, its computation depends on the shape of fuzzy sets (i.e. #pivotal points) but not on the number of Horn clauses or that of predicates within those. Furthermore, this computation even becomes less as those fuzzy sets are more simply represented (e.g., trapezoidal–only 4 points).

## Extraction

Extraction functionalities translate various types of data into a collection of extended Horn clauses. Following the concept of deductive databases, any data in tabular forms are translated into a collection of unconditional Horn clauses, i.e. facts, and any relational queries, e.g. SQL, are translated into a collection of Horn clauses (Codd 1970; Ceri, Gottlob, and Tanca 1990). Unstructured texts are to be translated into a collection of Horn clauses as a result of applying Natural Language Processing (NLP) such as tagging, syntax parsing and semantic processing in LP (Nugues 2006). Semi-structured data such as XML, E-mail and Electric Data Interchange (EDI) have a high compatibility with Horn clauses (Almendros-jimnez, Becerra-tern, and j. Enciso-baos 2008). As a consequence of this, anything that can be represented in XML is translated, e.g. diagrams.

Multimedia data such as images, audio and video are handled through their summarization, e.g. color histograms, edge and shape extractions and any other image processing. Tagged information and attributes are straightforwardly translated into Horn clauses. Texts such as captions are translated by applying NLP. Their contents can be efficiently summarized and, in a sense, compressed by applying Granular Computing and linguistic summary (Moore and Inoue 2008; Yager 1982). This can also be applied to any other data that are massively large and highly dimensional.

Overall, a rich set of extraction functionalities serves as a strong interface because Horn clauses are considered rather

[1]The conditional mass assignment, i.e. semantic unification, may also yield a point probability (Baldwin, Martin, and Pilsworth 1995). However, we do not consider this in ELP.

as a pivotal language (thus, users do not have to be extensively exposed to ELP). In knowledge management for infrastructure assurance, various factual (sensory) data can be entered in tabular forms and XML. Knowledge such as policies and scheduling rules can be entered in texts. Then, minor modification and refinement are to be made as deemed necessary through some human-computer interaction for visualization in decision making.

## Visualization Scheme: Front-End

The concepts of creating a visualization with various views represent the data at different levels of detail. We chose to implement a global view and a local view. The global view displays the relations in a wide range. This view allows a user to gain a broad understanding of the various components and relationships as a whole. Additionally, it is important for the user to be able to more closely understand particular subsets of the whole, particularly when the visualization is large. This necessitates the local view, which allows the user to drill down to a particular subsection of the global visualization and view the details of the relations.

We designed and implemented the prototype front-end application with several things in mind. This included the capacity to utilize the program on various platforms, leaving the door open for future expansion. For example, we wanted to ensure that this application was independent of any specific operating system. We also kept in mind that the future (or even the current) trend of technology is moving to service-based applications in cloud computing. Because the potential for this visualization system could grow to very large applications, having a powerful back-end system performing the operations and calculations could be beneficial, requiring only minimal processing power of the front-end system. Additionally, the system would be universally available and accessible regardless of where the user is. Due to these future possibilities, we designed the input specification around XML and implemented the visualization components in the Java programming language.

### Input file format design

We had to develop a format which would include all necessary information about the reasoning processes to be visualized. Because the reasoning process can easily be represented in a tree structure, we chose an XML format for the input file.

### Fuzzy probabilities in the local view

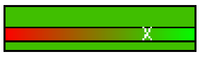

Figure 3: Point probability

In the aforementioned research, two methods for presenting probabilities in the local view are offered (Springer and Inoue 2009). One of these methods represents a crisp, single-point probability; for example: $0.8$. The paper suggests that a rectangular box is displayed. Inside this box, a smaller rectangular bar is displayed representing the possible range of probabilities (from $0.0$ to $1.0$) using a color spectrum or gradient. Therefore, the color at the left end of the bar represents a probability of $0$, and the color at the other end represents $1$. Any color in between is then easily seen as representing a probability somewhere between these possible values. The outer box is then filled with the color representing the point probability for that particular event. (See figure 3).

The second method involves representing an interval probability. This method is very similar to the last, in that we still have a rectangular box with a smaller bar with the spectrum representing the range of probabilities. The difference is that the outer box is filled also with a gradient over the probability interval for that event. So if the probability was $[0.1, 0.4]$, then the outer box would be filled with a gradient ranging over the colors represented within the spectrum between those values. (See figure 4).

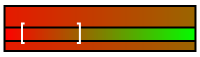

Figure 4: Interval probability

Both of these methods are very good visual representations of the probability. These visualizations make it quick to easily identify the probability of a particular event. They are also easy to compare, even between the single point probability and the interval probability. The challenge which we faced was determining an equally good method of visualizing a fuzzy probability. In this case, the probability of a particular event is represented by a fuzzy set. This means that each probability will have a membership value based on the membership function which defines the fuzzy set. After some discussion, we came up with three feasible representations of fuzzy probability for this particular project. There obviously could be many more ways to represent fuzzy probabilities; however, we needed ways that would be easy to directly compare with the other two representations.

The method which came to mind first was to represent the shape of the fuzzy set. This was quickly modified to include the gradient of color to enhance this representation. Inside the rectangular box used for the other two methods, the shape of the fuzzy set would be drawn and filled with the portion of the gradient which fit within that shape. (See figure 5).

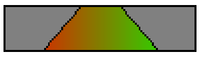

Figure 5: Fuzzy probability using the shape of the fuzzy set

The second method which we consider is to represent the probability with color gradients layered based on particular $\alpha$-cuts of the fuzzy set. After some experimentation, we discovered that the most effective method for representing in this way was to use the maximum number of $\alpha$-cuts based on the height of the containing rectangular box (in pixels).

So, if the containing rectangular box was 24 pixels tall, we take 24 $\alpha$-cuts of the fuzzy set and paint that row of pixels with the gradient representing the probability interval at that $\alpha$-cut. This results in a color pattern which we will describe as a two dimensional gradient. (See figure 6).

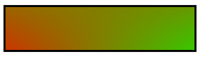
Figure 6: Fuzzy probability using interval gradients for each $\alpha$-cut

The third and final method we considered was to represent the fuzzy set by changing the color value of the gradient based on the fuzzy set. For example, decreasing the saturation or the brightness of the color based on the membership value of the particular point. For this representation, we developed three variations, one which decreased the saturation, another the brightness, and the other a combination of brightness and saturation. (See figure 7). After comparing these three options, we found that the method which decreased the brightness was the most clear and intuitive (see figure 8).

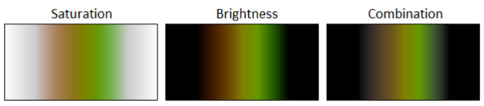

Figure 7: Comparing variations of color value

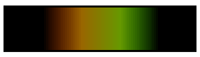
Figure 8: Fuzzy probability with decreasing brightness

After examining all three of these methods (shape of the fuzzy set, two dimensional gradients, and color value), we determined that certain people will view each of these with different degrees of usefulness. One person may find the first option the most intuitive. However, others may find the second or third options most intuitive. We decided that it would be most beneficial to include all three representations of fuzzy probability (see figure 9) in the local view program and allow the user to toggle between them depending on their personal preference or intuition. This will allow the user to choose an option which suits their eye and allows them to easily compare between point, interval, and fuzzy probabilities.

## Local view development

For this development, we chose to use the Java programming language for several reasons. First and most importantly, Java is platform independent. Java GUI programs can also easily be converted to web applets, which could make the application even more portable by making it available through a web interface online.

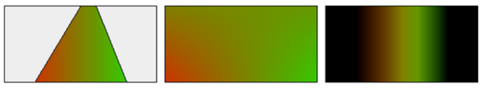
Figure 9: Comparison among shape of the fuzzy set, $\alpha$-cuts and decreasing brightness

**Layout** The local view was developed to be a box-in-box style layout. There is a top panel, a left panel, and a bottom right panel. The top panel is used for displaying information about the node, currently just the node's name. The left panel is used for displaying the calculated probability panel underneath the name of the node. The bottom right panel is then used as a container for any children of the node. All nodes with dependencies and children are given this same three-part layout. This layout is then added to the parent's bottom right panel, creating an embedded box-in-box style as specified. For the leaf nodes with no children, we simply display a single panel which contains the name of the node and a given probability panel to the right. The given probability panels for the deepest leaf nodes are drawn to touch the right border of their enclosing box, while other leaf nodes that are not as deep are indented to the left to allow quick vertical comparison between different levels. According to the input file specification, we can have several different 'branches' (i.e. separate Horn clauses) or dependencies grouped together by 'and's (i.e. conjunctively connected predicates within a Horn clause). This is represented in the local view by a slightly thicker border between the different children. Figure 10 shows a complete local view.

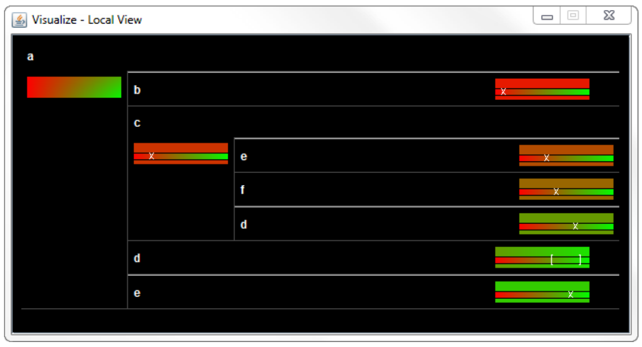

Figure 10: Complete local view

## Global view development

The global view is conceptually straight-forward, and simpler than the local view. However, implementation turns out to be more challenging. The global view is a representation of the entirety of the reasoning process. Ideally with this visualization application, a user will be able to view the whole reasoning process in the global view with little to no detail and, in order to see more detail, look at a particular subsection of the reasoning process in the local view. This means that the global view should accommodate a large number of nodes in a small amount of space, while still providing significant information regarding the reasoning process. Our previous work determined that the global view should be, what we call, a circular tree. It "*combines* the relationship visibility of *a standard tree structure* with the radial organization and space efficiency of *a tree ring structure*." (Springer and Inoue 2009).

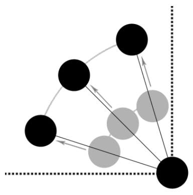

Figure 11: With a small radius, the nodes are too close together, but by increasing the radius, we create more space between the nodes while maintaining the same angles of placement

**Node Positioning** In order to develop this view, we needed to confront challenging issues. The first and the most critical issue is the node placement. We had to determine a method of calculating the position of each node. We considered a couple of different options, but decided that it was simplest to divide the space among the child nodes evenly. For example, the root node will begin with 360 degrees of space, which it will divide evenly among its children. Each of those nodes will then be given a placement angle as well as a certain number of degrees to allocate to their children. One issue with dividing the *'arc space'* evenly among the children is that we could have one branch with many children and descendants and another with very few. However, both branches would be given an equal amount of space. This requires us to ensure that all nodes are given enough space in their angle on the circumference. If we have nodes of a particular size and which have been given a certain angle with which to work, the only thing left to manipulate in order to give enough space is the radius (see figure 11). We decided to use a consistent distance between levels of the tree. This was the simplest to implement, as well as, what we believe to be, the most clear visually. So in order to calculate node positions, we must determine the required circumference to give enough space for the most nodes in the smallest angle as follows (equation 1, where $C$ is the set of children, $d$ is the node diameter, $a$ is the node's given arc space, and $l$ is the node level.).

$$\frac{1.5 \cdot |C| \cdot d}{2\pi \cdot a \cdot (l-1)} - d \tag{1}$$

Once we know the circumference, we can determine the radius, and since we know the depth level of the nodes, we can also determine the distance between levels, or what we call the link length. With this information, we can now easily calculate the positions of each node because we know its specified angle, its depth level, and the link length (i.e. the distance between each level). After calculating the placement of each node, we must generate the links between them. Each node had a reference to its parent node, and both nodes know their own positions, so we simply have each node draw a link from its position to its parent.

**'And Arcs'** With both the nodes and links in place, we must also draw connecting arcs for the branches which are grouped by 'and's. These 'and arcs' must connect the links from the first child node to the last which are part of the conjoined dependencies. To implement this, we must know the point of origin (i.e. the parent node's position), the angle of the first child in relation to the parent, and the angle of the last child in relation to the parent. This was a challenge as we have the angle for each node from the root node, not the angle with respect to the parent node. However, because we can calculate the coordinates of the child node as well as the parent node, we can calculate the angle from the parent as follows (equation 2, where $A_s$ is the start angle, $A_e$ is the end angle, $(x_s, y_s)$ is the start location, $(x_e, y_e)$ is the end location, and $(x_p, y_p)$ is the parent location. For coding, we assume $A_s,\ A_e \geq 0$ and make necessary conversion, e.g. adding 360 degrees.).

$$\begin{aligned} A_s &= \tan^{-1} \tfrac{y_s - y_p}{x_s - x_p} \\ A_e &= \tan^{-1} \tfrac{y_e - y_p}{x_e - x_p} \end{aligned} \tag{2}$$

After calculating the angle from the parent to both the first and last child in the 'and' group, we can then simply draw an arc from the first to the last.

**Zooming and Fuzzy Nodes** The capability to zoom in or out on the global view is very critical in visualization. As far as coding was concerned, this is fortunately very simple because all of the position calculations are based on the node diameter and angles. The zoom feature simply scales the node diameter, which effectively scales everything else. It has been designed as a slider control in the bottom of the window, but could easily be changed to be any other type of interface control as well.

Lastly for the global view, we added a simple indication of fuzzy nodes. For those nodes (represented in the local view with italicized names), we drew a dashed white circle just outside the node. This allows the user to easily differentiate between fuzzy and crisp events, but does not detract from the ability to see the coloring of the node. A completed global view is shown in figure 12.

## Coloring

The color calculations for the probabilities are made by a simple scale of the two primary color (red and green in our case) components by the probability. Because the probability is between 0 and 1, this scales each value between 0 and 255 in the RGB color space. To calculate the red component, we invert the calculation such that the higher the probability is, the lower the red value becomes. The combination of the scaled red and green values gives us our color.

For the interval probabilities, the calculations are the same, except that we just have to do the one for the low end of the interval and the other for the high end. To create the gradient, we generate the start color for the low probability and the end color for the high probability. This creates a gradient from the low to the high.

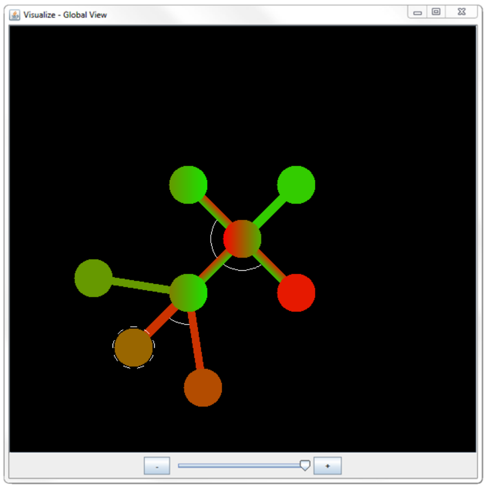

Figure 12: Completed global view

The concept of coloring for the local view had already been well-defined in Springer's previous work; however, the global view had not been specified aside from remaining consistent with the coloring in the local view. We determined that the leaf nodes should display their given probabilities and the non-leaf nodes should display their calculated probabilities. Then the links between nodes display the given probability for their appropriate 'and' branch. This is consistent with the local view in which the leaf nodes display their given probabilities and non-leaf nodes their calculated probabilities.

In order to represent point and interval probabilities on a node, we simply color the node the appropriate solid or gradient color in the scheme previously discussed. For fuzzy probabilities, we take the core, which is an $\alpha$-cut where $\alpha = 1$–representing the best-case interval. This produces an interval, which we represent with a gradient over the color spectrum discussed above. The links are colored in the same way, with the gradient going across the width of the link rather than down its length (see figure 13). Otherwise it could appear that the probability changes from one node to the next, when we are trying to represent a probability of the rule or event as a whole, not some transitional probability.

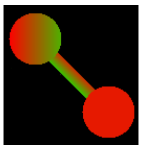

Figure 13: Node and link coloring

## Transition between local and global views

A future goal of this project is to allow a user to easily make transition between the local view and global view, and also to potentially include more detail in the global view as the user zooms and manipulates the view. First ideas include: the ability to look at a particular subsection or branch of the tree in the global view (from the global view), the ability to look at a particular subsection or branch of the tree in the local view (from the global view), the ability to go back to the complete global view (from the local view), and popup windows which include more specific information about the nodes and 'and' branches in both views (particularly the probabilities). We will briefly describe each of these ideas further.

The ability to look at a particular subsection or branch in the global view could be implemented in such a way that you click on a particular node in the global view and it makes this node into the root node in the center of the screen and repositions each of its children around it in the same circular fashion. This allows the user to more closely examine a particular branch without yet having to see all detail associated with the nodes (as in the local view). For example, assume we have a tree with a root node that has five children, and each of these branches has hundreds of descendants. The user could click on one of the five children, which would then become the root and take the center position, and its descendants would then be repositioned all around it, giving each more space and hopefully making it clearer to see the dependency links.

The ability to look at a particular subsection or branch in the local view would be very similar to the previous idea. However, once a user has found a particular (small) branch which a user wishes to examine more closely, the user can choose to view a particular node (and all dependencies/children) in the local view. This would transition them seamlessly and allow them to see all of the information the local view presents which the global view does not. Following this concept is the idea that a user should be able to easily return to the global view after examining a subsection or branch of the whole tree. Ideally the user should be returned to the same subsection from which they came in the global view to maintain a consistent frame of reference between views. However, this could also be accomplished by highlighting or outlining the nodes (in the global view) of the particular subsection the user was examining in the local view so the user can easily recognize and find the nodes in the scope of the rest of the tree.

Finally, we have had some study about various popups in each view. In the global view, a user could click a node and initiate a popup indicating the node's name together with calculated or given probability. Likewise, if the user were to click a link, a popup could indicate the given probability for that particular branch. Currently the given probability for an 'and' group is not shown in the local view, but a popup could display that probability for its branch. There are many ways in which a popup-style window could enhance both of the views. We plan to study those throughout a challenging knowledge management case.

We have a few other potential enhancements. As an alternative to the popup for displaying given probabilities of 'and' groups in the local view, we have studied effectiveness of placing additional probability panels to the left of the children which are a part of the 'and' branch. These panels would span the height of the children in the group and probably be narrower than the regular probability panels. However, this would distinguish these probabilities from those directly associated with a node, and would allow easy comparison between several branches at the same level of the tree. This could be fairly easily implemented by utilizing the existing probability panel classes which we have already implemented. A screenshot of such panels is shown in figure 14.

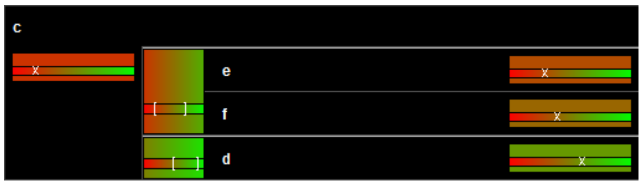

Figure 14: Given probability panels for 'and' groups (conceptual screenshot)

As mentioned earlier in the paper, we have a plan to develop this application in a Java applet on a web server for cloud computing.

## Concluding Summary

Our study on fuzzy relational visualization in system development aspects is presented. Our recent work has resulted in a prototype proof-of-concept that is capable of dynamic and scalable visualization. The visualization scheme and the first set of fundamental human-computer interactions have been developed. With the ever increasing complexity of various decision making, visualization is becoming more and more essential. There are many areas in which work still needs to be done; however, this prototype pushes the boundaries of current visualization techniques and limitations in the directions as introduced at the beginning of this paper.

While completing and further enhancing elements discussed in this paper, we are anticipating the following future works throughout challenging knowledge management domains, e.g. infrastructure security and health informatics:

- More sophisticated human-computer interaction and study on its effectiveness from aspects of cognitive sciences.
- Parallelization and scalability in aspects of concurrent logic programming for the back-end and that of distributed computing features in Java for the front-end.

## Acknowledgment

This work is conducted for and partially supported by NSF-IUSTF International Collaboration on Infrastructure Security and Health Informatics Technology and Management Initiative at Eastern Washington University. Computational resources are provided by Computer Science Department at Eastern Washington University.

## References

Almendros-jimnez, J.; Becerra-tern, A.; and j. Enciso-baos, F. 2008. Querying XML Documents in Logic Programming. *Theory and Practice of Logic Programming* 8(3).

Baldwin, J. F.; Martin, T.; and Pilsworth, B. 1995. *FRIL: Fuzzy and Evidential Reasoning in AI.* Research Studied Press.

Ceri, S.; Gottlob, G.; and Tanca, L. 1990. *Logic Programming and Databases.* Springer-Verlag.

Codd, E. F. 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* 13(6):377–387.

Heer, J.; Viegas, F.; and Wattenberg, M. 2007. Voyagers and Voyurs: Supporting Asynchronous Colalbollative Information Visualization. In *CHI2007.*

Heer, J. 2006. Socializing Visualization. In *CHI2006 Workshop on Social Visualiztion.*

Henry, M. D., and Inoue, A. 2007. Visual Tracer for Logic Programming. In *ISIS2007.*

Inoue, A. 2008. Uncertainty Management by Extension of Logic Programming. In *FSS2008.*

Inoue, A. 2010. Toward a Comprehensive Knowledge Management for Infrastructure Assurance. In *iWiA2010/IFIP TM2010*, 90–96.

Jeffrey, R. 1965. *The Logic of Decision.* McGraw-Hill.

Joxan, J., and Maher, M. J. 1994. Constraint Logic Programming: a Survey. *Journal of Logic Programming* 19/20:503–581.

Moore, Z. I., and Inoue, A. 2008. Effectiveness of Value Granulation in Machine Learning for Massively Large and Complex Domain. In *IPMU2008.*

Nugues, P. 2006. *An Introduction to Language Processing with Perl and Prolog.* Springer-Verlag.

Shneiderman, B. 1996. The Eyes have It: A Task by Data Type Taxonomy for Information Visualizations. In *VL96*, 336–343.

Shneiderman, B. 1998. *Designing the User Interface (3rd eds).* Addision-Wesley.

Springer, K., and Inoue, A. 2009. Novel Visualization Scheme for Reasoning With Uncertainty. In *NAFIPS2009.*

Springer, K.; Henry, M.; and Inoue, A. 2009. A General Application Framework for Intelligent Systems. In *MAICS2009*, 188–195.

Viegas, F.; Wattenberg, M.; Ham, F.; Kriss, J.; and McKeon, M. 2007. ManyEyes: a Site for Visualization at Internet Scale. *IEEE Trans. on Visualization and Computer Graphics* 13(6):1121–1128.

Wattenberg, M., and Kriss, J. 2006. Designing for Social Data Anaysis. *IEEE Trans. Visualization and Computer Graphics* 12(4):549–557.

Yager, R. 1982. A New Approach to the Summarization of Data. *Information Sciences* 28:69–86.

Zhang, J. 2008. *Visualization for Information Retrieval.* Springer-Verlag.