

Statement of Research Interests and Goals

Rajshekhar Sunderraman
Computer Science Department
Georgia State University
May 2005

My research interests and activities over the past six years have mainly centered around three areas: *Databases*, *Middleware for Distributed and Mobile Applications*, and *Hybrid Intelligent Systems and its Applications*. In addition to these research areas, I was also actively involved in a NSF funded "Digital Library" project that created a central repository of educational materials for Computer Graphics and Visualization Education.

More recently, I have started working in the emerging area of bioinformatics, in particular data management issues related to life sciences data. As part of a P20 Planning grant from NIH, I am working on the problem of efficiently storing, querying, and mining protein structure data. I am also actively participating in a neuro-informatics project funded internally by a seed grant from the Brains and Behavior program at Georgia State University. This project involves the creation and management of a Web-based resource that catalogues and organizes identified neuronal types and their synaptic connections for a number of invertebrate species. This resource is expected to benefit neuroscientists who study model invertebrate nervous systems which are composed of individually identifiable neurons.

The following sections summarize my research efforts during the past six years in the three broad areas:

I. Databases

Data Models for Non-Standard Data: I have continued to build on my earlier research on representing and manipulating incomplete, inconsistent, and uncertain information by considering neutrosophic sets, a generalization of fuzzy, paraconsistent, and other non-classical sets, and using them to model uncertain, inconsistent, and incomplete information. I have also studied a special case of neutrosophic sets, termed paraconsistent intuitionistic fuzzy sets, that allows paraconsistent relational databases to be generalized to include "belief" and "doubt" measures. The data model is capable of modeling both incomplete as well as inconsistent information and standard relational algebraic operators are generalized to query against the incompleteness and inconsistencies present in the database. We have also proposed an SQL like construct for querying such data. Most of this work is summarized in the recently published book (publication 12 in the CV).

Global Querying and Constraint Checking in Multi-Databases: In this research, we have studied the problem of querying and updating a system of databases, each operating in an autonomous manner with an independent schema, much like a multi-database. To query a system of databases, we have proposed a Java API that hides many of the complexities involved in programming a global query across the set of databases. We have also looked at the problem of maintaining integrity constraints that are global in

nature, i.e. the constraints refer to more than one object residing in one or more of the databases. We have proposed an elegant constraint decomposition algorithm and have implemented it using mobile agents. Several optimizations have also been proposed to this algorithm as well as an extension to handle constraints that involve aggregate operations has been designed.

Web and Semi-Structured Data: In the area of Web and semi-structured data, I have looked at the problem of constraint specification and enforcement in multiple XML databases. An efficient constraint decomposition algorithm is introduced to deal with global constraints over a collection of XML data sources. The constraint checking is carried out before any updates are performed on the database thereby saving any potential rollback times that may be needed to recover from constraint violation. I have also studied the problem of effectively querying data from multiple Web sources using Wrapper methodologies.

II. Middleware for Distributed and Mobile Applications

I have also been actively involved in a middleware project, called System on Devices (SyD). SyD is a middleware test-bed which allows programmers to rapidly develop and deploy collaborative applications running on a collection of heterogeneous and possibly mobile devices on a network, each potentially hosting data stores of interest to the users of the application. Developing such applications using current technologies is tedious and time consuming and SyD relieves the programmer of many of the cumbersome data and network programming details by providing a high level API to work with. Data communication is done using XML and remote procedure calls are enabled via Web Services.

The design of SyD is modular and consists of several kernel components. Two of the main components are the SyD Listener and the SyD Engine modules. The SyD Listener module is a lightweight component that can easily be deployed on a number of devices including hand-held devices and cell phones. The listener module enables remote procedure calls to be executed on the host device and serves data from the local data store. The SyD Engine module allows for method invocations on individual as well as groups of objects. It consists of a dispatcher component that is responsible for making the necessary method invocations on remote objects and an aggregator component that accumulates the results from these method invocations and presents an aggregated view of the results back to the invoking application program.

The success of SyD middleware platform has been demonstrated in two key collaborative applications: the calendar application and the fleet application. In the calendar application, several individuals are maintaining their independent schedule information in their hand-held devices. The typical functionalities provided in such an application are: set up meeting among individuals with certain conditions to be met such as a required quorum, set up tentative meetings which could not be set up otherwise due to unavailability of certain individuals, and remove oneself from a meeting resulting in automatic triggers being executed that may possibly convert tentative meetings into

confirmed ones. The fleet application involves the operations of a shipping and delivery company such as FedEx within a city with trucks equipped with devices that communicate with other trucks as well as stationary nodes representing warehouses and central stations. Several problems such as efficient scheduling of deliveries, ad hoc rescheduling of deliveries due to accident situations, etc were studied and implemented.

III. Hybrid Intelligent Systems

In the area of Hybrid Intelligent Systems, I have used intelligent agents, neural networks, fuzzy logic, and granular computing techniques to build many different applications.

In a recent work, we have proposed a framework to evaluate Web services using soft computing methodologies. When confronted with a task of choosing between competing Web services offering the same capability, this framework allows the dynamic computation of the Quality of Service (QoS) of Web services. The main evaluation engine employs neutrosophic neural networks with genetic algorithms to calculate the QoS of competing Web services.

In an earlier work dealing with college selections, we proposed a fuzzy rule tree based approach to evaluate colleges for possibility of admission based on a number of characteristics. The system is an expert agent which uses a hierarchical fuzzy knowledge base using fuzzy logic for inferences. The system designed was more of a framework capable of implementing a number of similar applications from a number of domains. With a simple specification change, the college selection system could be easily transformed into a physician selection system which evaluates physicians for the possibility of best match to deal with a patient's problem.

In a series of papers, we have also designed and implemented several Web applications that use hybrid soft computing methodologies to incorporate intelligent behavior into the system. Several Web data mining systems were also designed and implemented that use soft computing methodologies.

Future Goals

I now present some of my future research goals. I am very much interested in solving some difficult problems that are associated with life sciences data that is growing at an astronomical pace. I have begun work with two Ph.D. students in the area of *Domain Specific Data Modeling*. In one case, we are proposing to introduce an abstract domain-specific layer (in this case genomic domain) on top of an existing object-oriented database system. This abstract layer would contain several domain specific data types and operators. A domain-specific query language is being designed that would be easily understood and used by domain scientists to pose ad-hoc queries. We are also proposing a limited domain-specific programming language using which domain scientists would be able to formulate small scripts or programs on their own and run them to transform their data. In another project in the neuro-informatics domain, we are proposing a "neuron data

model” that is capable of representing complex information about neurons and their connections to other neurons. We are also proposing a neuron query language that will allow neuro-scientists to interact with their data in an easy to use manner. This work is in conjunction with a larger database building project called NeuronBank that is creating a universal resource of neuronal information for neuro-scientists to share. This project is akin to the more widely used data banks such as Protein Data Bank, GeneBank etc.

One other near term goal of mine is in the area of XML query processing using XML to Relational or XML to object mappings. In this work, we are proposing to develop query transformation algorithms that automatically map queries in XML query languages such as XPath or XQuery into corresponding queries in relational or object databases. This mapping would be tied to a specific data mapping and would have to satisfy a correctness criterion typically used in mapping systems such as the one used here. The advantage of such query mappings would be that we could use robust relational or object systems to store XML data and provide for XML querying without the need for a native XML database.