Formal Methods for Accelerating Formal, Semi-formal and Dynamic Property Verification through Novel Specification Styles

Ansuman Banerjee

To my family and my supervisor

Contents

CERTIFICATE

This is to certify that the dissertation entitled **"Design and Analysis of Context Aware Systems"** submitted by **Soumi Chattopadhyay** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by her under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Ansuman Banerjee
Assistant Professor,
Advanced Computing and Microelectronics Unit,
Indian Statistical Institute,

Kolkata-700108, INDIA.

Abstract

Large scale context aware distributed systems are becoming a reality with emerging paradigms like machine-to-machine communications, crowdsensing, Internet-of-things etc. Scalable data distribution is a critical requirement in such large scale systems for optimal usage of computing and communication resources. In this work, we present a novel strategy for data distribution in such large-scale context aware systems that distributes only relevant data based on its effective utility. Experimental results show the efficacy of our proposed scheme.

Keywords: *Context aware system, Co-factor algebra, Data dissemination.*

1

Background and related work

In this chapter, we first present a few background concepts needed for developing the foundation of our framework. We also present an overview of different schemes proposed in literature for data dissemination.

1.1 Background

In this section, we discuss a few background concepts.

1.1.1 Propositional Logic

Propositional logic is widely used in diverse areas such as database queries, in artificial intelligence, automated reasoning etc. A proposition is a sentence which is either true or false. If a proposition is true, then we say its truth value is true, and if a proposition is false, we say its truth value is false. The syntax of formulas in propositional logic is defined by the following grammar:

$$formula = formula \ \wedge \ formula | \neg formula | (formula) | atom$$
$$atom = BooleanIndicator | True | False$$

Other Boolean operators such as OR ($\vee$) can be constructed using AND ($\wedge$) and NOT ($\neg$).

1.1.2 Satisfiability

In computer science, Boolean, or propositional, satisfiability (often written SATISFIABILITY or abbreviated SAT) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. In other words, it establishes if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to *TRUE*. If no such assignments exist, the function expressed by the formula is identically *FALSE* for all possible variable assignments. In this latter case, it is called unsatisfiable, otherwise satisfiable. SAT was the first known example of an *NP-complete* problem [**?**].

1.1.3 Linear Temporal Logic

The use of temporal logics [**?**] in verification was proposed by Pnueli in a seminal paper [**?**]. Since then several different logics have been proposed for specifying temporal properties. All these logics use the two basic temporal operators – *next* and *until*. Some of these logics also use additional temporal operators that can be derived out of the basic two. The logics differ in terms of how we are allowed to mix these operators to express the desired formula.

In this section, we introduce the popular temporal operators and the logics that are built around them. In this part we also introduce some formalisms in an intuitive way that show us how these logics are interpreted over time.

The basic temporal operators

The formal introduction to a language has two main parts, namely the *syntax* and the *semantics*. The syntax defines the *grammar* of the language – it tells us how we may construct properties using the basic set of signals and operators. The semantics define the *meaning* of the properties.

The semantics of the traditional temporal logics were defined over *closed systems*, which are finite state machines without any inputs. This tradition has been followed in languages such as SVA and PSL

as well – there is no distinction between input and non-input variables in these languages. At this point we present the semantics of these languages in the traditional form over a non-deterministic finite state machine. Open systems (modules having input bits) can be modeled by treating the input bits also as state bits. This typically yield a non-deterministic state machine, since the choice of inputs in the next state lies with the environment, and is not a function of the present state.

Suppose J is a finite state machine having k state bits. Each of the 2^k valuations of these state bits represent a *state* of the machine. Let S denote the set of these states. Let R denote the state transition relation of J. R consists of pairs of states, (s_i, s_j), where it is possible to transit from state s_i to state s_j. Finally, J has a start state s. Formaly we say that J is a tuple $\langle S, s, R \rangle$.

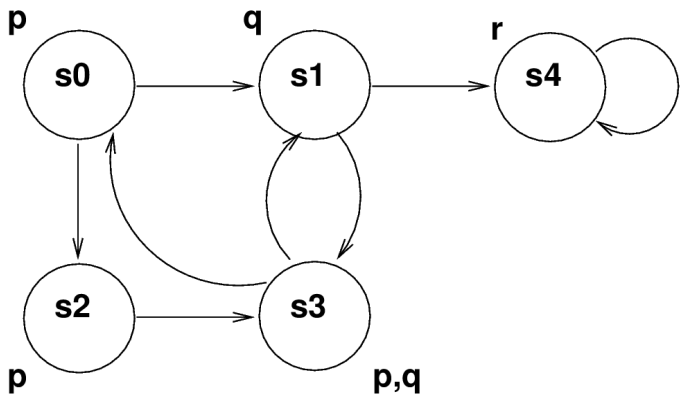

Fig. 1.1. A sample finite state machine

Example 1.1. Fig 1.1 shows a 3-bit finite state machine. Let the state bits be n_0, n_1, n_2. The state bits are shown on the nodes. The start state is s_0. Fig 1.1 shows 5 states – the remaining three states are not reachable from the start state and are not shown. The circuit has three outputs, which are functions of the state bits. These are:

$$p = n_0 \ \vee \ n_1$$
$$q = n_2$$
$$r = \neg n_0 \ \wedge \ \neg n_1 \ \wedge \ \neg n_2$$

The nodes of Fig 1.1 are labeled by the outputs that are true at that state. We shall use this toy example to demonstrate the meaning of various temporal properties. □

Intuitive explanation

To convey the semantics of the basic temporal operators, we first introduce the notion of a *run* (alternatively, a *path* or a *trace*). A run, π, of J is a sequence of states, $\nu_0, \nu_1, \ldots$, where $s = \nu_0$ is the start of the run, and for each i, ν_i represents a state in S, and R contains a transition from the state represented by ν_i to the state represented by ν_{i+1}. In other words, the run is a sequence of states repesenting a valid sequence of state transitions of J. For example the run, $\pi = s_0, s_1, s_3, s_1, s_4, \ldots$, is one run of the state machine shown in Fig 1.1. States of the machine may be revisited in the run – for example we have $\nu_1 = \nu_3 = s_1$ in π. The run, $\pi' = s_0, s_2, s_0, \ldots$, does not belong to this state machine, since it has no transition from s_2 to s_0.

Let us now consider the two fundamental temporal operators, namely *next* and *until*, and a run $\pi = s_0, s_2, s_3, \ldots$.

Next operator: A property, *next* f, is true at a state of a run iff the property f is true at the next state on the run. For example, *next* q is false at the state, s_0, of the run, $\pi = s_0, s_2, s_3, \ldots$, since q is false at the next state s_2. The property *next next* q is true at s_0, of π, because q is true at s_3.

Until operator: A property, f *until* g, is true at a state of a run iff the property g holds on some future state, z, of the run, and the property f holds on all states preceding z on the run. For example, the property, p *until* q, is true at the start state of π, since q is true at the state s_3 and p is true at the states s_0, s_2 preceding s_3 in π. The property, p *until* r, is false on all paths of Fig 1.1, because no r-labelled state can be reached along a p-labelled path starting from s_0.

We now define the two other operators namely, *always* and *eventually*. To do this, we need the definitions of the propositions, *TRUE* and *FALSE*. We say that the proposition, *TRUE*, holds in all states, and the proposition, *FALSE*, is false in all states.

Eventually operator: A property, *eventually* f, is true at a state of a run iff the property f holds on some future state in the run. Since the proposition, TRUE, holds on all states, we can express the *eventually* operator using the *until* operator as:

$$\textit{eventually } f \;=\; \textit{TRUE until } f$$

The property, *eventually* q, holds on all runs starting from s_0 in Fig 1.1. The property, *eventually* r, does not hold in the run which loops forever in the loop s_0, s_2, s_3, s_0.

Always operator: A property, *always* f, is true on a run iff the property f holds on all states of the run. This is the same as saying that $\neg f$ never holds on the run. In other words we may write:

$$\begin{aligned} \textit{always } f &= \neg \textit{eventually } \neg f \\ \textit{eventually } f &= \neg \textit{always } \neg f \end{aligned}$$

The first equation allows us to express the *always* operator using the *eventually* operator, and in turn, in terms of the *until* operator. The second, says: *sometimes is not never* – there is a seminal paper with this title by Leslie Lamport [**?**].

The property, *always* p is true in the run which loops forever in the loop, s_0, s_2, s_3, s_0, in Fig 1.1. The property is false in all other runs of the same state machine.

The duality between the *always* and *eventually* operators is not surprising. In fact, it is a variant of DeMorgan's Laws when we interpret the properties over time. This is because:

$$\begin{aligned} \textit{eventually } f &= f \;\vee\; \textit{next } f \;\vee\; \textit{next next } f \;\vee\; \textit{next next next } f \;\ldots \\ &= \neg\,(\neg f \;\wedge\; \textit{next } \neg f \;\wedge\; \textit{next next } \neg f \;\wedge\; \textit{next next next } \neg f \;\ldots) \\ &= \neg\,(\textit{always } \neg f) \end{aligned}$$

Linear Temporal Logic (LTL) is the most popular among linear time logics. We can define the syntax of linear temporal logic recursively as follows:

- All Boolean formulas over the state variables are LTL properties.
- If f and g are LTL properties, then so are: $\neg f$, Xf, and $f\ U\ g$.

We can also use the short-forms, Fg for $true\ U\ g$, and Gf for $\neg(true\ U\ \neg f)$.

Formal semantics

It is very important to know the formal semantics of a formal property specification language. If the semantics is specified ambiguously, there may be a gap between the property that the designer intends to express and the formal property tool's interpretation of the property that she writes. Bugs may hide in this gap thereby defeating the whole purpose of *formal* property verification. Language lawyer volunteers who make up the working groups of the language standards committees spend years debating over the exact formal semantics of the languages that they standardize. The goal of standardization is to ensure that languages with precise definitions are made available to improve communcation within the industry.

The problem with understanding formal semantics is that they are replete with terse notations. It is widely suspected that the intimidating nature of the notations used in existing literature on formal property verification is one of the main deterants to its wider adoption in practice.

To start with, we use a set of short-forms. We use X to denote the *next* operator, U to denote the *until* operator, G to denote the *always* operator, and F to denote the *eventually* operator. G means *globally with respect to time*, and F means *in the future*.

Let $\pi = \nu_0, \nu_1, \ldots$ denote a run, and $\pi^k = \nu_k, \nu_{k+1}, \ldots$ denote the part of π starting from ν_k. We use the notation $\pi \models f$ to denote that the property f holds on the run π. Given a run π, we also use the notation $\nu_k \models f$ to denote $\pi^k \models f$. In other words, a property is said to be true at an intermediate state of the run iff the fragment of the run starting from that state satisfies the property. The formal semantics of the basic temporal operators are as follows:

- $\pi \models Xf$ iff $\pi_1 \models f$
- $\pi \models f\ U\ g$ iff $\exists j$ such that $\pi_j \models g$ and $\forall i, 0 \leq i < j$ we have $\pi_i \models f$.

Fg is a short-form for TRUE U g, and Gf is a short-form for $\neg F \neg f$.

1.2 Logics for temporal specification

Temporal logics tell us how we can create complex temporal properties by putting together one or more temporal operators. There are broadly two classes of these logics, namely *linear time logics* and *branching time logics*. Linear time logics allow the specification of properties over linear traces or runs of a finite state machine – intuitively, we say that the property holds on the machine if it holds on all runs of the machine. Branching time logics allow the specification of properties over the computation tree created by a state traversal of the state machine.

Linear Temporal Logic

Designers and validation engineers typically express and interpret the RTL in terms of the simulation semantics of the HDL. They are accustomed to verifying the correctness of the RTL by checking certain behaviors over simulation traces. Therefore it is not suprising that linear time logics have been the natural choice for design validation, and form the backbone of most existing property specification languages, including Forspec, PSL and SVA.

The semantics of LTL is as follows. We say that the property f holds on a state machine, J, iff f holds on all paths of the state machine starting from its start state. The semantics of f on a path is as defined in the last section.

Let us see some sample LTL properties obtained by using one or more temporal operators. We refer to Fig 1.1.

- The property $p\ U\ q$ is true in the state machine, since all paths from s_0 satisfy this property.
- The property Fq is true in the state machine, but the property GFq is not true. This is because we have the path, $\pi = s_0, s_1, s_4, \ldots$, which does not satisfy Fq from s_4 onwards.
- The property $p\ U\ (q\ U\ r)$ is not true in the state machine, because it may get trapped in the loop, s_0, s_2, s_3, s_0.

Fig 1.2 shows some sample LTL properties and some sample runs that satisfy these properties.

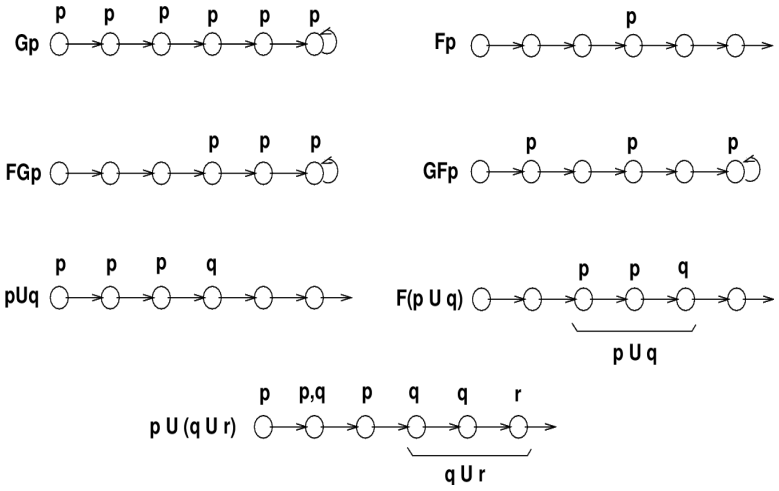

Fig. 1.2. Some sample LTL properties

Computation Tree Logic

Computation Tree Logic (CTL) is a branching time temporal logic. Properties described in this logic are interpreted over the *computation tree* obtained by unfolding the state machine as a tree. We elaborate on this shortly, but let us first study the basic features of this logic.

CTL has two *path quatifiers* in addition to the usual temporal operators. These are the *existential* path quantifier E, and the *universal* path quantifier A.

- The property $A\varphi$ is true at a state, ν, iff φ is true on *all* runs starting from ν.
- The property $E\varphi$ is true at a state, ν, iff φ is true on *some* run starting from ν.

In CTL we have the restriction that every subformula of the form Xf, Gf, Fg, and $f\ U\ g$ must be prefixed by an E or A. Therefore, we may define the language as:

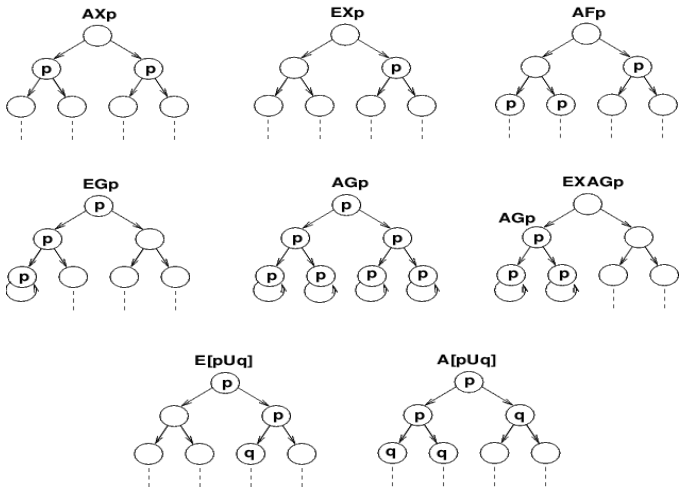

Fig. 1.3. Some sample CTL properties

- All Boolean formulas over the state variables are CTL properties.
- If f and g are CTL properties, then so are: $\neg f$, AXf, EXf, $A[f\ U\ g]$ and $E[f\ U\ g]$.

We also have the usual short-forms Fg for $true\ U\ g$, and Gf for $\neg(true\ U\ \neg f)$. Consequently, EFg, AFg, EGf, AGf are CTL properties. Fig 1.3 shows some sample CTL properties and some sample computation trees that satisfy these properties.

What is the significance of a *computation tree*? Let us consider the state machine of Fig 1.1. We can unfold the state machine into the infinite tree shown in Fig 1.4. Each path of this tree is a run or a *computation* of the state machine. CTL allows us to define properties over the nodes of this computation tree.

For example, consider the property:

$$A[p\ U\ E[q\ U\ r]]$$

This property is true at the start state s_0 of our state machine. This follows from the fact that $E[q\ U\ r]$ is true at s_1 and s_3 (since there is a q-labeled sequence of states to the r-labelled state s_4), and every path from s_0 reaches one of these states through p-labelled states. It is not possible to express this property in LTL.

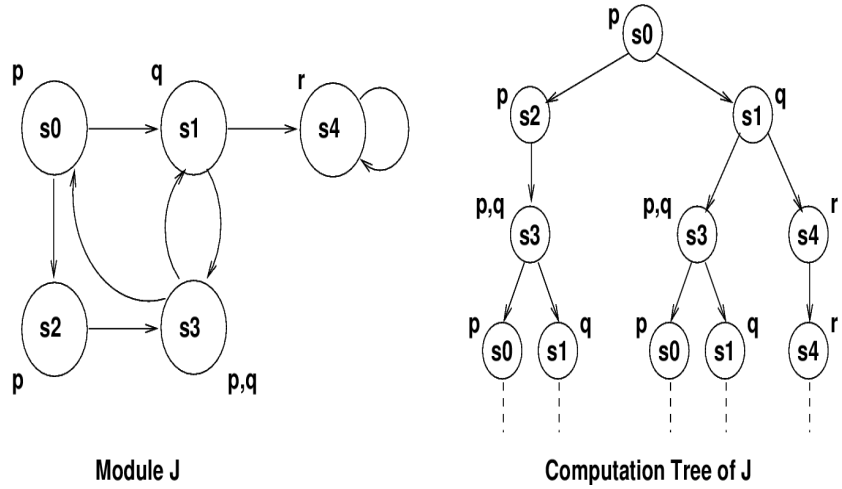

Fig. 1.4. The notion of a Computation Tree

LTL versus CTL

Can all LTL properties be expressed in CTL using the universal path quantifier, A? The answer is, no. For example, the LTL property, FGp, is not equivalent to the CTL property, $AFAGp$. Fig **??** shows an example which satisfies FGp but does not satisfy $AFAGp$.

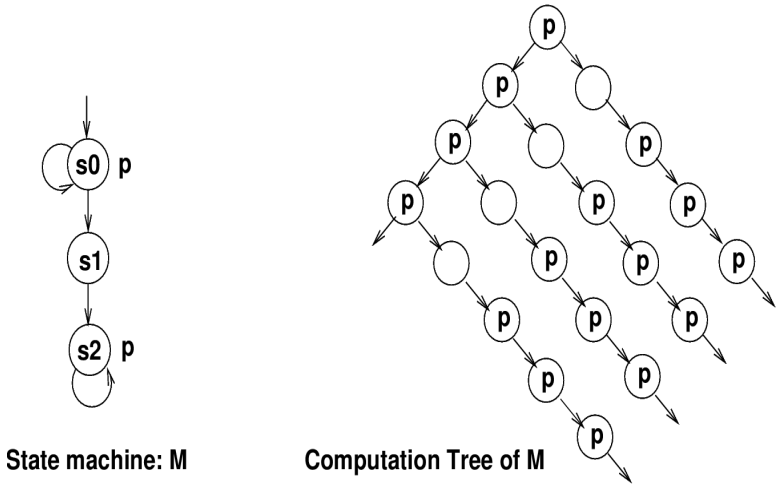

Fig. 1.5. Module M satisfies FGp, but not AFAGp

When do we use a linear time logic, and when do we use a branching time logic? This is a matter of considerable debate, and is hardly agreed upon. However, experience shows that linear time logics are the natural choice for black-box testing. For example, while specifying the behavior of a module, we can write linear time properties over its interface signals without knowing the internal state machine of the module.

On the other hand, branching time logics are useful for verifying properties over a given state machine. For example, when we develop an automotive control system we may typically model the control system as an abstract state machine, verify whether this control system satisfies certain safety properties and then expand the abstract state machine into the actual control system. When a branching time property fails, we must interpret the failure in terms of the actual states of the system, which is not possible in black-box testing.

The temporal logic CTL* combines the expressive power of linear and branching time temporal logics. CTL and LTL are fragments of CTL*. There has been several interesting extensions of these languages that demonstrate the tradeoff between the expressive power of the language and the complexity of model checking (that is, formally verifying) properties specified in these languages.

Bounded temporal logics

The temporal operators discussed so far, namely *next* (X), *until* (U), *always* (G), and *eventually* (F), are *temporal* because they can define sequences of events over time. Significantly, none of these operators with the exception of the *next* operator, attempt to *quantify* time. For example the property, *eventually* f, requires f to be true in future, but does not specify any time bound by which f needs to be true.

Real time temporal operators are intuitively simple extensions of the basic *untimed* temporal operators where we annotate the operator with a time bound. The real time extensions of CTL and LTL simply use these bounded operators (as well as the unbounded ones).

The bounded Until operator: The property $fU_{[a,b]}g$ is true on a run, $\pi = s_0, s_1, \ldots$, iff there exists a k, $a \leq k \leq b$ such that g is true at s_k on π, and f is true on all preceding states, $s_0, \ldots, s_{k-1}$. Formally,

$\pi \models f\ U_{[a,b]}\ g$ iff $\exists k, a \leq k \leq b, \nu_k \models g$ and $\forall i, 0 \leq i < k$ we have $\nu_i \models f$

The bounded LTL property $p\ U_{[1,3]}\ q$ is true at the state s_0 of Fig 1.1. The bounded CTL property:

$$A[p\ U_{[1,3]}\ E[q\ U_{[1,2]}\ r]]$$

is also true at s_0. This is because s_3 and s_1 satisfy $E[q\ U_{[1,2]}\ r]$ (since they can reach s_4 within the time bound $[1, 2]$), and we reach s_1 or s_3 along all paths from s_0 within the time bound $[1, 3]$.

The bounded Eventually operator: The property $F_{[a,b]}g$ is true on a run, $\pi = s_0, s_1, \ldots$, iff there exists a k, $a \leq k \leq b$ such that g is true at s_k on π. For example, the bounded LTL property $F_{[1,3]}q$ is true at the state s_0 of Fig 1.1. The bounded CTL property $EF_{[2,4]}r$ is true at s_0. The property $F_{[a,b]}g$ is equivalent to the bounded-until property, $true\ U_{[a,b]}\ g$.

The bounded Always operator: The property $G_{[a,b]}f$ is true on a run, $\pi = s_0, s_1, \ldots$, iff f is true in every state in the sequence, $s_a, \ldots, s_b$. The bounded CTL property $EG_{[3,9]}q$ is true in the state s_0 of Fig 1.1 – consider the run from s_0 through s_2 which alternates between s_3 and s_1 at least 8 times. The bounded LTL property $G_{[0,1]}\neg r$ is true at s_0 since no run can reach s_4 is less than 2 cycles.

Real time operators are extremely useful in practice. Most design properties have a well defined time bound, and must be satisfied within that time.

Since the real time operators deal with finite bounds, a and b, they can be expressed in terms of the X operator. For example, the property $F_{[2,4]}q$ can be rewritten as:

$$F_{[2,4]}\ q\ =\ XX(q\ \vee\ Xq\ \vee\ XXq)$$

and $p\ U_{[3,4]}\ q$ can be rewritten as:

$$p\ U_{[3,4]}\ q\ =\ (p\ \wedge\ Xp\ \wedge\ XXp)\ \wedge\ XXX(q\ \vee\ (p\ \wedge\ Xq))$$

The first part of the property specifies that p be must be true in the present cycle and the next two cycles. The second part of the property specifies that q must be true in the third cycle, failing which, p must be true in the third cycle and q must be true in the fourth cycle.

Therefore, real time operators actually help us to succinctly express properties that would require too many X operators otherwise.

1.3 Related Work

In this section, we discuss some related work that has been done so far. In the first part of this section, we discuss various data dissemination techniques existing in literature. This is followed by a brief analysis of context aware systems and energy aware sensor networking.

The directed diffusion [**?**] paradigm for data dissemination is one of the most popular techniques that has been done in the area of wireless sensor networks. This approach aims at energy efficient, scalable, robust communication, where a sink node floods interest messages to each of its neighbors periodically and sensor nodes which have matching data, called events, send to the originator of interest along multiple paths. The sensor networks reinforce one or a small number of these paths. Another data-centric data dissemination approach was proposed by Ditzel and Langendoen [**?**], which combines the advantages of directed diffusion, data-centric routing like SPIN and energy-efficient MAC protocols such as S-MAC and T-MAC. Jian Chen et al. [**?**] proposed an energy efficient data dissemination scheme in wireless sensor networks by reducing redundant data dissemination and avoiding query flooding, extra data transmission. Hung Le Xuan proposed [**?**] a grid based data dissemination approach for reducing energy consumption and increasing the network lifetime. The entire sensor network is divided into a number of grids where each grid has a coordinator which acts as an intermediate node to cache and relay data. They considered three major phases in their proposal. In the first phase, i.e., data announcement phase, a source generates the data announcement message and floods the message to all coordinators. When a sink needs data, it sends query to the coordinator in the grid, where the sink belongs to, in the query transform phase. A path is established based on the target's location and grid ID while the query traverses to the source. Finally, in the data dissemination phase, the source sends data to the sink along the established path. They also considered mobile sink in their work. The mobile sink chooses its nearest coordinator to act as its agent. The sink checks its location periodically. If it moves from one grid to another, it first sends cache removal message to clear the previous path and resends a query to set up a new path. A similar kind of idea was proposed in [**?**], where the authors present a cluster based multi-path approach for data dissemination. In their approach, they define five types of sensor nodes in intra cluster transitions: normal node, dis-

semination node, dissemination backup node, cluster head node and cluster head backup node. Dissemination nodes take care of routing decisions for query and data dissemination. One of the dissemination nodes is considered as the cluster head. The cluster head is responsible for collecting and aggregating sensed data in the cluster. When the original dissemination node dies, the backup node takes over the responsibility of that node. Sensors use greedy forwarding approach to deliver packet to the sink, i.e., as the packets are forwarded they come closer to the destination. Habib M. Ammari and Sajal K. Das [**?**] proposed an information theoretic approach for resource efficient, energy aware data dissemination in wireless sensor networks. In their proposal, they assume that the sink node is moving in a wireless sensor field and they try to quantify the uncertainty of position of a mobile sink. For this purpose they propose a weighted entropy-based data dissemination protocol. Sajid Hussain et al. [**?**] proposed collaborative agent system architecture for data dissemination in wireless sensor networking. Sink mobility repeatedly sends packets in-order to notify its current location to the sources. Frequent location updates from multiple sinks result in increased transmission overhead as well as rapid power expenditure. Wang and Chiang [**?**] proposed a grid-based Power Aware Data Dissemination scheme that reduces the power consumption and also minimize the transmission traffic of frequently transmitted packets. A similar approach is discussed in [**?**]. Min-Gu Lee and Sunggu Lee [**?**] proposed a low-control-overhead data dissemination method, in terms of pseudo-distance data dissemination. Guoliang Xing et al. [**?**] proposed a solution for dynamic multi-resolution data dissemination for minimizing total energy consumption using Minimum Incremental Dissemination Tree (MIDT), an online tree construction algorithm. Miao Xu et al. [**?**] proposed a content aware data dissemination approach which aims to deal with data privacy and availability issues by formulating a multi-target optimization problem. They proposed their solution for data dissemination using distributed graph coloring. Daojing He et al. [**?**] proposed the data dissemination technique based on hash tree. Virtual Line-based, energy efficient data dissemination protocol is proposed in [**?**]. Instead of flooding, this approach exploits a virtual line structure for data storage within the group region. The main idea is that sources store their data on virtual line structures and the sink retrieve their necessary data from this. A comprehensive survey of different data dissemination techniques can be found in [**?**].

We now present an overview of context aware systems and energy aware sensor networks. There is some work on context aware application programming [**?**], web application programming [**?**]. It is worth mentioning that public alert services are becoming popular in smart cities. Context awareness enhances the efficiency of such a system. [**?**] proposes a framework where a selective notification will improve the pervasive experience. Stream reasoning is the backbone of this system which uses rule-based reasoning and queries. The paper [**?**] has proposed a health monitoring system. The paper presents a context-aware access control system fitted to ubiquitous medical sensor networks. [**?**] proposes a middleware (ACE: Acquisitional Context Engine) for auto-learning the relationships among various context attributes and using them to optimize inference caching and speculative sensing.

Context-awareness in mobile devices based on integration of multiple diverse sensors is proposed in [**?**]. The author of this paper presents an approach of substituting a single powerful sensor by a group of various simple sensors, and context is derived from multi-sensor data. In [**?**], a scheme of integration of context aware computing by sensornet (CASN) is presented. A multidimensional framework for context-aware systems is proposed in [**?**]. Olaru et al. [**?**] proposes a software agent based model for a middleware on Ambient Intelligence (AmI). Here context-awareness is implemented both in "agents representation of context information", and in the "logical topology of the agent system". The limitation of this model is the orientation towards decentralization of the system and reliability on local behavior.

A lot of work has been done on energy aware sensor networking domain. A scalable and energy-efficient context monitoring twofold framework has been proposed in [**?**] for sensor-rich mobile environments where limited resource is concerned. The idea consists of continuous detection of context changes followed by bi-directional approaches to the context monitoring problem. The author claims to obtain higher efficiency in energy consumption. Other energy-efficient data transmission reduction methodologies [**?**, **?**] are proposed for periodical data gathering in WSN. These methods exploit overhearing, where the redundancy of the reading of each node in a WSN is determined by the overheard packets transmitted by its neighbors and restricts the redundant reading transmission. In [**?**] a method for reducing the energy consumption on a visual sensor node in Wireless Sensor Network (WSN) is proposed.

Main objective of the paper is to balance the processing and transmission tasks where subjective image based measurements are concerned. The method described here tries to find the best compression ratio that achieves a significant reduction of transmission time, compression time and returning adequate and objective image quality. In the paper [**?**], energy aware routing using sub-optimal paths is considered to get substantial gains. Different routing protocols have been proposed [**?**] considering the energy awareness in WSNs. S-MAC, a medium-access control (MAC) protocol designed for WSN is proposed in [**?**]. A better approach to conserve energy as compared to IEEE 802.11 is demonstrated in the paper. Another approach to reduce data communication in WSNs is proposed in [**?**]. This approach is based on fuzzy numbers and weighted average operators. It also states the process of estimating the lifetime of the network using the datasheet of the sensor node and the number of messages received or transmitted. A redundant data transmission protocol in wireless sensor-actuator network (WSAN) is proposed in [**?**]. They proposed an idea to forward the received message as redundant by another message sent by the sensor node. A different approach for reducing the data transmission in WSNs using The Principal Component Analysis has been proposed in [**?**]. Ciullo et al. [**?**] proposed an energy efficient scheme for data collection in WSN using mobile elements (with mobility control). Here, for minimizing total transmission energy within a certain travel time, they considered the problem of optimal vehicle trajectories as a function of data at each sensor. For storing or sending bulky data, Chang et al. [**?**] used Bloom filters to reduce the memory and network transmission necessities. Using this approach more information from WSN can be collected and also network lifetime can be extended.

For efficient query response in a context based system, the *RETE* algorithm [**?**] is used, which basically deals with a set of *If-then (-else)* rules. In this approach, facts are matched against rules. We also use this kind of a rule to reduce the number of transmissions. The main difference of our work with existing literature is the fact that we leverage on the Boolean nature of the context rule for efficient message dissemination.

To the best of our knowledge, our proposed model is the first work of its kind. We have not found any work on data dissemination using context rule analysis.