Declarative Logic Accelerator

Wolf Kohn
Michael Eastwick
Intermetrics Inc.

William Bricken
Oz International

Introduction

Over a period of several years, the authors have conducted joint and independent research to explore formal methods in Equational Declarative and Deductive Inferencing theories [1], [2], [3], [4], [5], [6] for accelerating the real time computation of dynamic object evolution and object constraints in a variety of applications. In early February 1993, a technically feasible, sound acceleration concept was formulated.

The accelerator architecture, outlined in this introduction, consists of The following operational elements: a Declarative Wave Logic Accelerator (DWLA), two Pattern Matchers, a Graphics Driver, a Traffic Sensor System, a Renderer, a Knowledge Base of Primitive Objects, a Knowledge Base of Primitive Object Dynamics and a Requirements Driver. These elements and their interaction are show in figure 1.

- Traffic Control Application Requirements Driver

The function of the architecture is to provide acceleration of the logic and evolution computations of traffic control applications, specified by a Script of requirements stored in the application requirements driver. The script, written in equational clause syntax [6], is a set of declarative directives characterizing the desired behavior of the application.

- Declarative Wave Logic Accelerator

The Declarative Wave Logic Accelerator is a massively parallel deductive inferencer over a Boolean domain. Its function is to determine the scene state by solving a set of simultaneous Boolean equations which are determined from the data generated by the Traffic Sensor System. The solution of this set of equations, the new state of the scene, is passed to the graphics driver which transforms it for display by the Renderer.

- Pattern Matchers

Each of the pattern matchers has the function of extracting knowledge from each of the knowledge databases as a function of each term in the script . This information is in the form of a sequence of basic (minimum and equivalent) Boolean Equational expressions which are passed to the Declarative Wave logic Accelerator and the Graphics Driver respectively. The equational expressions generated by the pattern Matcher that feeds the Declarative Wave logic accelerator, produces an equational Boolean pattern that characterizes scene transformation updates, the pattern matcher that feeds the graphics driver generates a pattern that specifies the geometric characteristics of the bodies comprising the scene. Resolution and unification of these two equational patterns is fed to the renderer for display.

- Graphics Driver and Renderer

These two devices are commercially available devices.

The architecture is provided also with a system controller / scheduler, and auxiliary memory which, for simplicity, are not shown in figure 1. The architecture is applicable to accelerating any logic process. The inferencing processes, implemented in the DWLA and Pattern Matcher elements of the architecture, is the same for any application. The knowledge databases, however, would be logic process dependent.

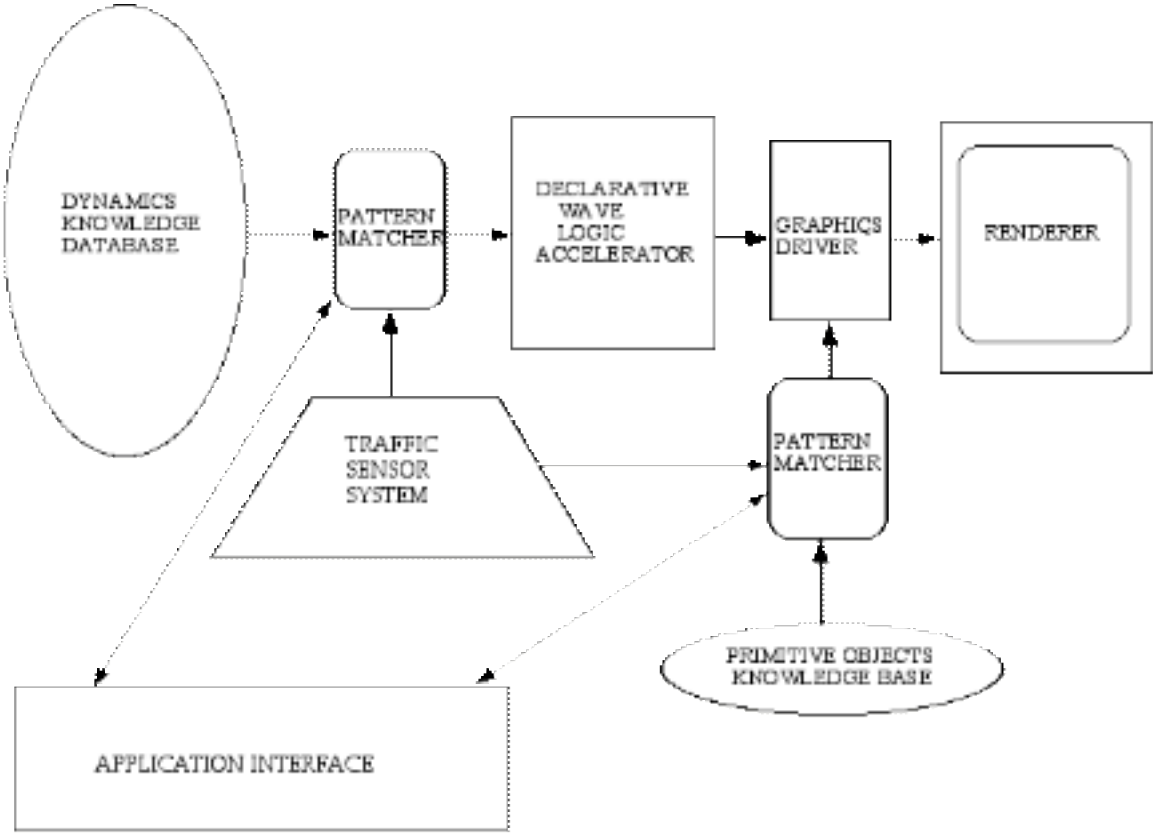

Figure 1.Functional Architecture

Technical Description

Declarative Wave Logic Accelerator (DWLA)

This section is divided in two subsections: Structure and Operation. In the first subsection we describe the functional characteristics of the proposed device. The second subsection is devoted to an overview of how it works.

DWLA Structure

The DWLA is a synchronous two dimensional array of identical processors that communicate in nearest-neighbor fashion. A diagram of the device is shown in figure 2. The device is provided with two bi-directional I/O registers labeled boundary registers. A computation process by the device, which we will describe in the Operation subsection, is triggered by data in those registers.

The I/O data and the internal data the device computes on, are given by Boolean equations over a multivalued Boolean algebra, called the Domain Algebra. This algebra characterizes the expressions from which the alluded equations are built.

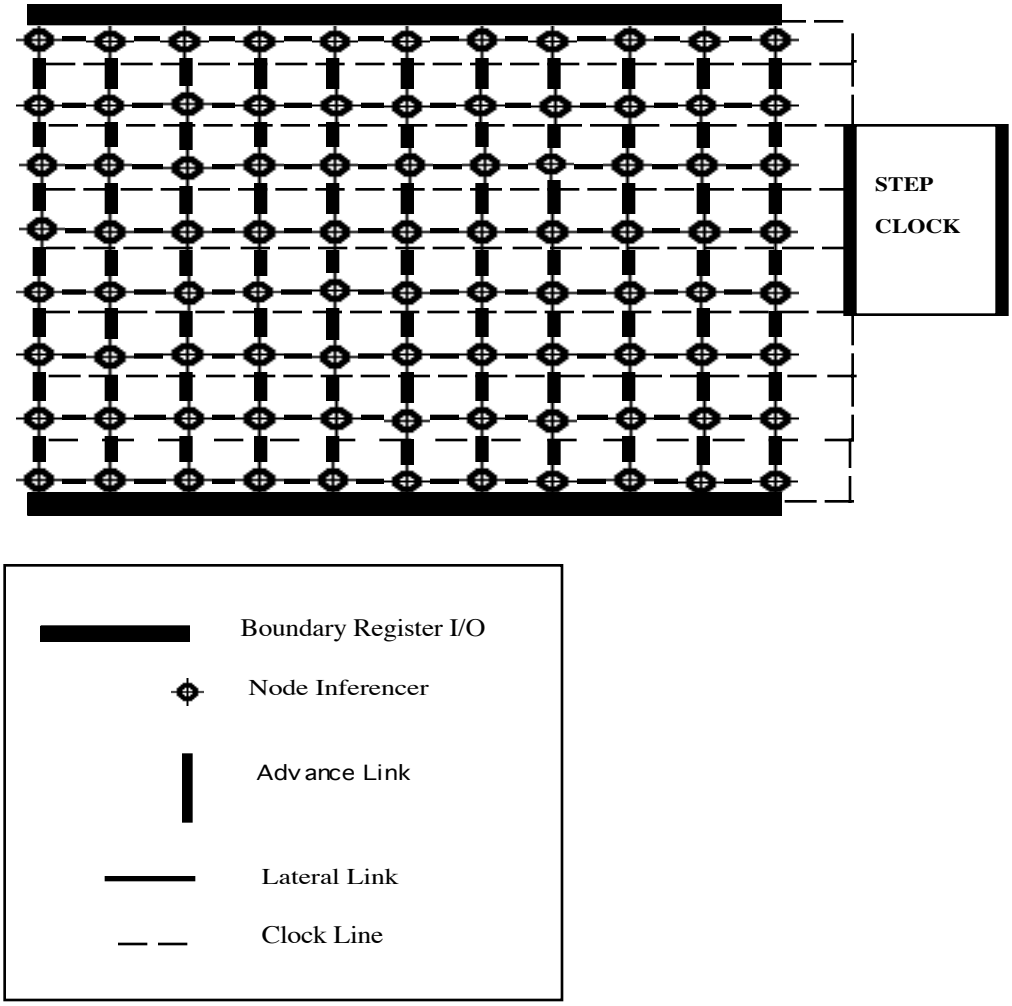

Figure 2. Declarative Wave Logic Accelerator

Each processor in the array, which is called a Node Inferencer, executes one of a finite number of basic inferences on its input data: a set of equations generated by its nearest neighbors. A basic inference step is a transformation in the space of Boolean equations. It transforms the input equations into a single output equation. The flow of equational data is accomplished through the

network connecting the processors. This network is composed of two types of links: Advance Links and Lateral Links.

The structure of both types of links is similar. Each link is a channel capable of transmitting a tuple of Boolean expressions (an equation) but, the advance links carry in addition a binary field indicating Wave Propagation Direction. We will explain this concept in the Operation sub section. In addition to the nearest-neighbor links, each processor is connected to the clock of the device via a clock channel for synchronization. Execution and data transfer is triggered by the timing signal issued by the Step Clock to each inference processor via the clock channel.

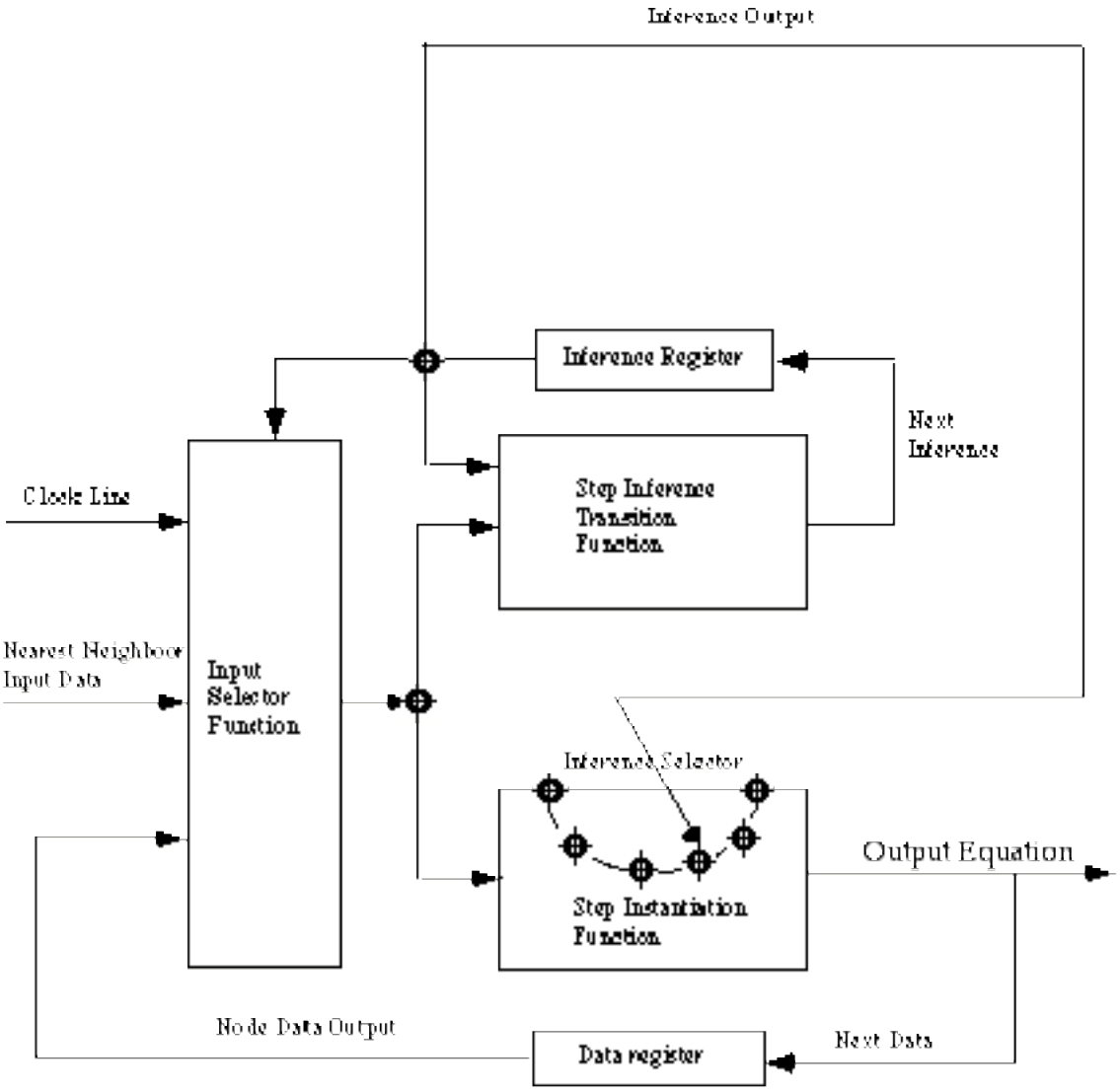

Figure 3. Node Inferencer

To a large extent, the computational capabilities of the Declarative Wave Logic Accelerator are determined by its number of inference processors. We will discuss this in the Behavior sub section.

The architecture of an Inference processor is shown in figure 3. It consists of three computational elements called Input Selector Function, Step Inference Transition Function, and Step instantiation Function and two registers, the Inference register and the Data register.

For each inference processor, the Input Selector Function extracts from its nearest neighbor's data a subset that is compatible with the current inference transformation to be executed. The Label of this inference (i.e. its name) is stored in the inference register. The subset of nearest neighbor's data associated with an inference operator is referred to as its neighborhood. Examples of some possible neighborhoods are shown in figure 4. In the figure, a small oval indicates a node in question and the big oval indicates the neighborhood corresponding to the current inference step. Notice that each inference processor in the array may be an element of its own neighborhood.

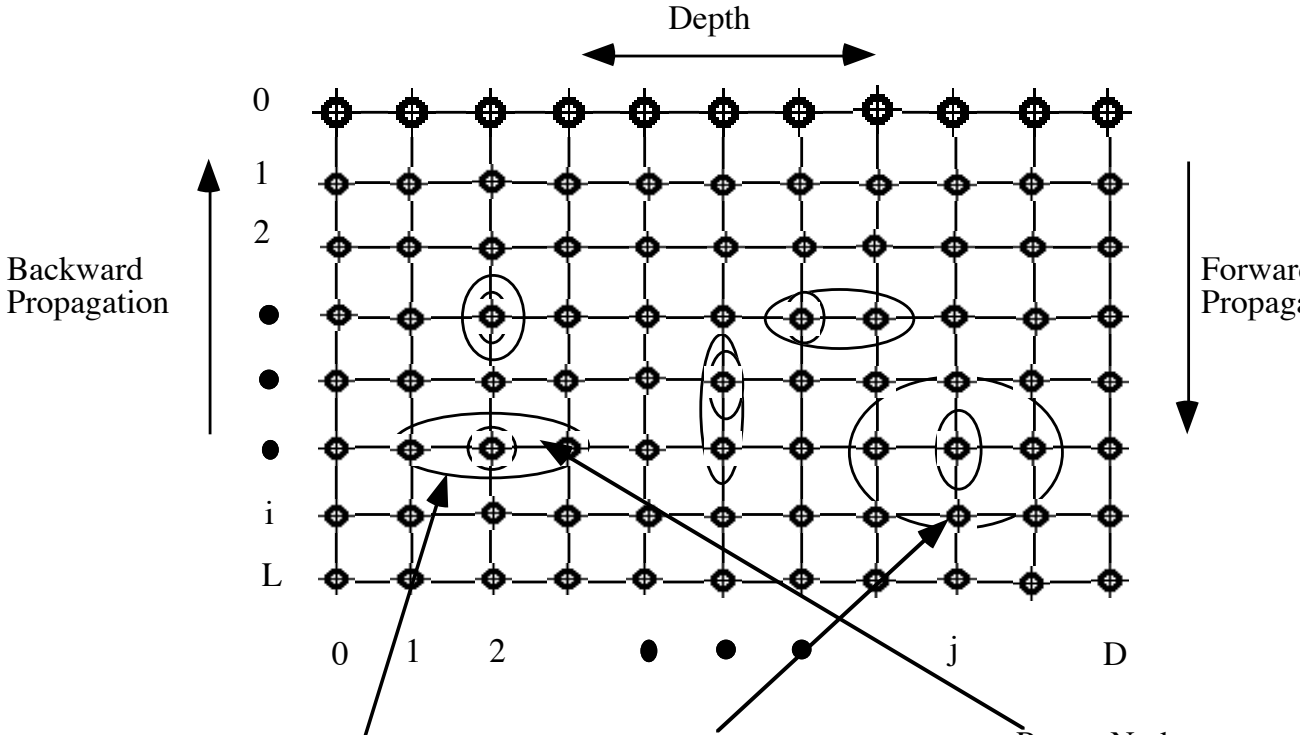

Figure 4. Node addressing and Node Neighborhoods

The input selector function data flow is as follows: upon receiving the step clock signal, it reads the inference register to determine the current inference transformation. then it computes the appropriate neighborhood and reads in the inference data registers of the processes in this neighborhood. this data is fed to the step inference transition and step instantiation functions for further processing.

The step inference transition function computes the label of the next inference transformation to be executed by the inference processor as a function of the current inference label stored in the inference register and the neighborhood data. This label is stored in the inference register

The step instantiation function is a device that encodes the inference transformations for transforming a set of Boolean equations into a single equivalent equation with lower term expression complexity. Each of the inference transformations, which are discussed in detail in [1],

are themselves given by Boolean expressions over the Cartesian product of the domain Boolean algebra of the device. Thus, each of the Boolean transformation can be implemented with combinations of the standard switching gates (and's, or's, not's). Each implemented inference transformation is assigned a unique distinct label.

In addition to the inference transformations, the step instantiation function includes an inference selector whose function is to read the inference register to get the current label of the inference transformation and to activate it.

Once an inference transformation is activated, it transforms the current neighborhood equation into an equation whose term complexity is no larger than of the neighborhood equations and stores that result in the data register.

In synthesis, the declarative wave logic accelerator is a massively parallel device for the solution of simultaneous Boolean equations. Given a set of Boolean equations, they are transformed into a single Covering equation external to the device, then this equation is fed to the accelerator via one of the boundary registers, and the corresponding variable instantiations are collected in a boundary register (possibly the same one). As we shall see in the next section, the resolution strategy is guided across the device via the implementation of a propagation strategy that makes the variable instantiation and equation transformation proceed in the form of a computational monotonic wave.

DWLA Operation

The operation of DWLA is characterized by the evolution of its State over the computation time. In principle, the state of the DWLA at each step clock update is given by the contents of the inference and data registers of each of its inference nodes. We will see below that the Operational State is a proper subset of the DWLA state.

Each inference node can be either Dormant or Active. An inference node is dormant if its inference register contains the label corresponding to the identity inference operator and its data register contains the trivial equation 1 = 1. An inference node is active if its inference register contains any label different than the identity or its data register contains anything different than the trivial equation.

A Wave is a collection C_t of tuples of the form (l, e, d, (i ,j)) where l is an inference label, e is a Boolean equation, d is a binary digit indicating propagation direction and (i,j) is the address of an inference node of DWLA. The sub index indicates the current (t) clock update.

A Computation by DWLA, is a sequence of waves propagating in one of two directions, Forward (d=1) or backwards (d=0). In figure 5, two concurrent computations are illustrated; one propagating forward and the other propagating backwards. If at time t, a non boundary inference node at location (i, j) is participating in a forward computation, the results of its inferencing can only be accessed by the input selector functions of the inference nodes at locations (i, j), (i+1, j), (i-1, j), (i, j+1). The direction of propagation, imposes similar restrictions on feasible neighborhoods for backwards propagation.

If in a computation, a node inferencer whose data register is connected to one of the two boundary registers forms part of a forward (backwards) propagation wave and has an equation with unresolved variables (i.e. an equation of the form $p(x_1, ..., x_n) = q(x_1,...,x_n)$ with $x_1,...,x_n$ variables over the Domain Boolean Algebra, and p, q, Boolean expressions) in its data register, it becomes part of a backwards (forward) propagation wave which is considered part of the same computation. This is the only mechanism provided for wave reversal.

The step inference and step instantiation functions in each node inferencer define the Wave Propagation Function W. This function prescribes the transition of waves as a function of time. If in a computation C_t is the current wave, the Next Wave C_{t+1} is given by $C_{t+1} = W(C_t)$

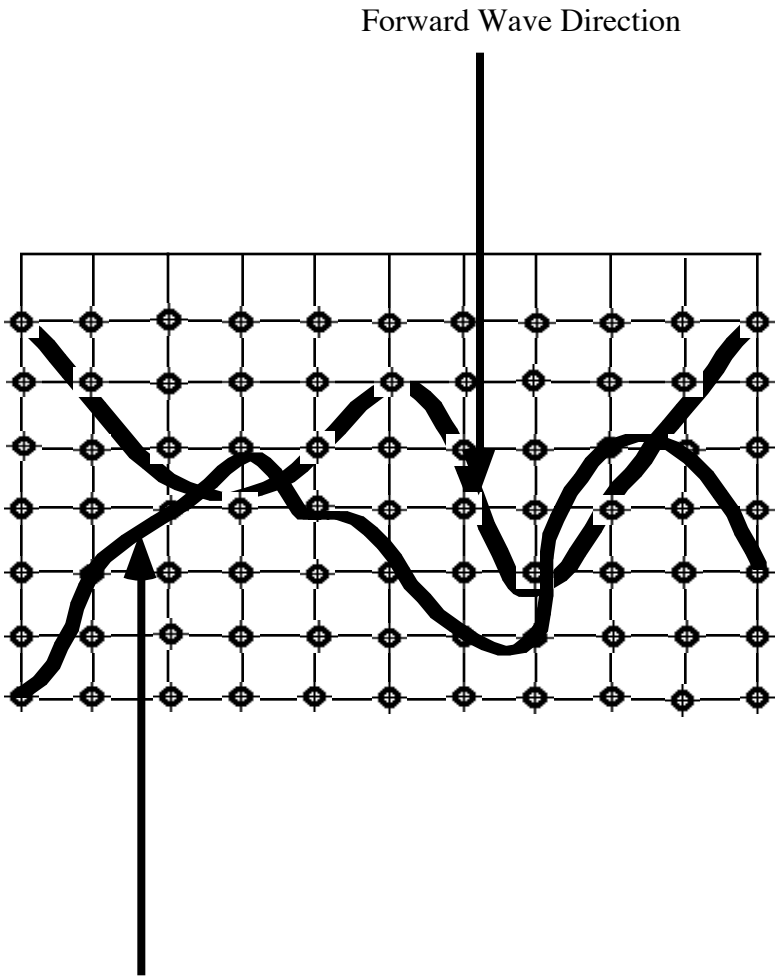

Figure 5. Wave Propagation

An equation stored in one of the boundary registers constitutes an initial wave C_0 of a computation, whose nth step is given by the wave C_n :

$$C_n = W^n (C_0)$$

where W^0 is the identity transformation and $W^n (C) = W (W^{n-1} (C))$. A Successful Computation is a computation such that Cn is a Ground Equation (a ground equation is a Boolean expression in Minterm canonical form. That is, a disjunction of conjunctions of value assignment to variables) stored in a boundary register.