Journal of Universal Computer Science, vol. 5, no. 5 (1999), 288-306
submitted: 23/2/99, accepted: 6/5/99, appeared: 28/5/99

The Average Case Performance of an Algorithm for Demand-Driven Evaluation of Boolean Formulae

Paul E. Dunne and Paul H. Leng
(University of Liverpool, U.K.
ped@csc.liv.ac.uk)

Abstract: Demand-driven simulation is an approach to the simulation of digital logic circuits that was proposed, independently, in the work of several authors. Experimental studies of the paradigm have indicated that this approach may reduce the time required for simulation, when compared with event-driven techniques. In this paper we present some analytic support for these experimental results by analysing the average number of gates evaluated with a naive demand-driven algorithm for formula evaluation.

Key Words: Simulation, Logic Design; Average-case Analysis of Algorithms;

Categories: F.2.m, B.6.3

1 Introduction

Almost all systems for the computer-aided design of digital systems and VLSI circuits employ simulation as a means of analysing and verifying the functional behaviour of the circuit under design. Logic-based simulation is most readily effected using a model of the circuit as a directed graph in which nodes representing logic gates are linked by edges representing circuit connections through which logic signals are propagated. Simulation proceeds by performing component evaluations in an appropriate sequence to determine signal values for edges of the graph.

In the case of modern VLSI systems employing 10^6 or more circuit elements (gates), speed of simulation becomes significant, and this has led to continuing interest in efficient methods for logic-level simulation. The simulation method most usually employed is *event-driven* simulation, e.g. [Szygenda and Thompson 1975], [Ulrich 1969], in which each change in value of a signal arising from the activation of a component is used to schedule the activation of components for which this signal is an input. The resulting *event-list* of scheduled component evaluations is serviced continuously, becoming empty when a cycle of simulation is completed.

Since, in general, a circuit component may have more than one input signal, event-driven simulation may lead to unnecessary repeated evaluation of a component. To overcome this problem, an alternative method of *demand-driven* simulation has been investigated independently in [Jackson 1986], [Jackson *et al.* 1987], and [Smith *et al.* 1987]. In this case, the activation of a circuit component involves a set of demands for the values of signals which are inputs to this component, and thence to demands for the activation of those components which are the sources of these signals. The recursive demand-sequence is satisfied ultimately using signal values provided as circuit inputs, or propagated from a previous simulation cycle.

As well as ensuring that, in any combinational circuit, no component need be activated more than once, demand-driven simulation offers another significant advantage arising from the characteristics of certain logic functions, including in particular $\wedge$ and $\vee$-gates. Since the output of an $\wedge$-gate, for example, is known to be 0 if any one input has the value 0, the invocation of demands for gate inputs can cease as soon as an input is found to have the value 0. In these cases, it may be possible to evaluate circuit outputs without the need to evaluate all intermediate gates and signal values, as described in [Charlton *et al.* 1991].

The potential for *lazy evaluation*, cf [Henderson and Morris 1976], can be exploited most effectively if in the simulation of appropriate components, demands for the evaluation of input signals are made in an order which minimises the total number of signal evaluations required. In [Dunne and Leng 1992], this problem was examined using a model in which there is associated with each signal, x_i, a probability, p_i of its having the value 1, and a cost, w_i which is interpreted as the *expected* time to evaluate x_i. Using this model, an optimal ordering strategy is described for the sequential evaluation of circuits comprising components which are *threshold functions* i.e. those Boolean functions that return the result 1 if and only if at least (or at most) some number k of the arguments have the value 1. A variant of this model was investigated within a more general context in [Jackson 1986].

Our aim in the present paper is to provide analytic support for the experimental evidence concerning the efficiency gains offered by demand-driven simulation methods. To this end we analyse the 'typical' behaviour of a naive demand-driven algorithm. More precisely, we obtain bounds on the average number of components that are evaluated in Boolean formulae of a given size. Thus, given a random Boolean logic formula of size, m, we are concerned with determining the number of components we expect to have to evaluate, as a function of m.

The remainder of the paper is organised as follows. In the next section we introduce the notations and definitions that will be used throughout. In particular we define a formal model for calculating the expected behaviour of demand-driven simulation methods. Also in this section, we discuss the choice of probability distributions for selecting formulae of a given size. This choice has an important bearing on the generality of the results derived since we are concerned with an average-case analysis. Section 3 concentrates on simplifying the expression for average-case evaluation complexity that results from the models introduced in the previous section. In Section 4, the main results of this paper are proved: we consider two probability distributions over formulae — both of which are discussed in Section 2 — and prove that one of these gives rise to an upper bound of $O(m^{0.5})$ for the average number of gates evaluated in a formula of size m; while the other distribution yields an $O(1)$ upper bound on the same measure. We discuss some issues concerning the development and application of the main technical results in Section 5. Conclusions are given in Section 6.

2 Preliminary Definitions

2.1 The Simulation Model

In this section we describe the simulation model, terminology, and notations that will be used subsequently. Our aim is to analyse the average number of evaluation steps taken by a specific demand-driven logic simulation method on Boolean *formulae* of a given size.

Definition 2.1: B_2 denotes the set of binary Boolean logic operations. Let $\Omega \subseteq B_2$ and $\mathbf{X_n} = \langle x_1, x_2, \ldots, x_n \rangle$ be an ordered set of n Boolean variables. An n-input *combinational circuit over the basis* Ω is modelled as a directed acyclic graph, S, in which there are two distinguished types of node: *input* nodes which have in-degree 0; and *gate* nodes which have in-degree 2. There are n input nodes, each of which is associated with a specific variable $x_i \in \mathbf{X_n}$. Each gate node is associated with some binary operation $\theta \in \Omega$. S contains a unique node with out-degree 0, termed the *output*. In a natural way, an assignment of Boolean values, α to the input variables $\mathbf{X_n}$ induces a Boolean result $S(\alpha)$ at the output of S. S *realises* a given n-input Boolean function, $f(\mathbf{X_n})$ if and only if, for all assignments α, $S(\alpha) \equiv f(\alpha)$.

An n-input Boolean *formula* over the basis Ω is a combinational circuit in which all non-output *gate* nodes have out-degree equal to 1. The *size* of a formula $F(\mathbf{X_n})$, denoted $|F|$, is defined to be $\sum_{x_i \in \mathbf{X_n}} out\text{-}degree(x_i)$, i.e. the total fanout from the input nodes of F. •

A formula, $F(\mathbf{X_n})$, may be viewed as a binary tree with $|F|$ leaf nodes, thus in such a tree there are $out\text{-}degree(x_i)$ distinct leaves associated with the input x_i for each $x_i \in \mathbf{X_n}$. We shall adopt this view of a formula of size m as an m-leaf binary tree in the analyses of the remainder of the paper. To avoid excessive repetition, we shall refer to formulae over the basis Ω as Ω*–formulae*.

Definition 2.2: Let $\Omega \subseteq B_2$. A Boolean operation $\theta \in B_2$ is called $\wedge$*–type* if

$$x \,\theta y \;\equiv\; (x^{\alpha} \wedge y^{\beta})^{\gamma} \qquad \{\alpha, \beta, \gamma\} \in \{0,1\}$$

where $z^{\varepsilon} = z \oplus \varepsilon \oplus 1$. θ is called an $\oplus$*–type* function if

$$x \,\theta y \;\equiv\; x \oplus y \oplus \alpha \qquad \alpha \in \{0,1\}$$

($\oplus$ denotes the binary Boolean function which takes the value 1 if and only if its two arguments have different values).

If θ is $\wedge$*–type* then there is a Boolean value, denoted c_{θ}, with the property that $c_{\theta}\,\theta 0 \equiv c_{\theta}\,\theta 1$. •

$\oplus$-type functions require the evaluation of *both* arguments in order for their result to be determined whereas $\wedge$-type functions may only need one input to be calculated.

For $\Omega \subseteq B_2$ we use the the following notations:

ω will denote $|\,\Omega\,|$.
$\Omega_{\wedge} \;=_{def}\; \{\ \theta \in \Omega : \theta \textit{ is } \wedge\text{–}type\ \}$; σ will denote $|\,\Omega_{\wedge}\,|$.
$\Omega_{\oplus} \;=_{def}\; \{\ \theta \in \Omega : \theta \textit{ is } \oplus\text{–}type\ \}$.

The demand-driven simulation algorithm which we examine is a rather naive one.

```
function evaluate ( F(X_n) : formula;
                    m : integer    -- m = |F|;
                    α : assignment to X_n )
         return boolean is -- returns F(α)
 y, z : boolean;
 begin
   -- F contains a single input node x_i
   if m = 1 then
     return value of x_i under α;
   else
     -- m > 1, so F(X_n) ≡ G(X_n) θH(X_n) for some G, H and θ.
     -- Since F is a formula, we have |G| + |H| = |F|
     y := evaluate (G, |G|, α);
     if y θ0 = y θ1 then
       return y θ0;
     else
       z := evaluate (H, |H|, α);
       return y θz;
     end if;
   end if;
 end evaluate;
```

Algorithm 2.1: Demand-driven formula evaluation

The recursive form of Algorithm 2.1 above, provides the basis for defining the number of evaluation steps used on a given formula with a specific input instantiation.

Definition 2.3: Let $F(\mathbf{X_n})$ be a formula of size m and $\alpha \in \{0,1\}^n$ be an assignment to the input arguments of F. The *evaluation complexity* of $F(\mathbf{X_n})$ under α, is denoted by $W(F, \alpha)$ and recursively defined as:

If $m = 1$, then $W(F, \alpha) = 0$

If $m > 1$ then $F(\mathbf{X_n}) \equiv G(\mathbf{X_n})\,\theta H(\mathbf{X_n})$ where $|G| + |H| = m$. In this case

$$W(F, \alpha) = \begin{cases} 1 + W(G, \alpha) & \text{if } G(\alpha)\,\theta 0 = G(\alpha)\,\theta 1 \\ 1 + W(G, \alpha) + W(H, \alpha) & \textit{otherwise} \end{cases}$$ •

Following the convention of [Jakoby *et al.* 1994] it is assumed that the input arguments of a formula are available without any cost, hence $W(x_i, \alpha) = 0$.

The quantity with which this paper is concerned is the *average* value of $W(F, \alpha)$ where this average is calculated over all Ω-formulae $F(\mathbf{X_n})$ of size m that use particular bases $\Omega \subseteq B_2$ and over all input assignments $\alpha \in \{0,1\}^n$. We denote this value by $S(m, \Omega)$, thus

$$S(m, \Omega) =_{def} \sum_{F(\mathbf{X_n}) \in F(m, \Omega)} \sum_{\alpha \in \{0,1\}^n} \mu_n(\alpha)\,\beta_m(F(\mathbf{X_n}), \Omega)\, W(F, \alpha) \qquad (2.1)$$

where, μ_n is a probability distribution over $\{0,1\}^n$ and β_m is a probability distribution over $F(m, \Omega)$ the set of n-input Ω-formulae having $|F| = m$. We note at this point, that whenever the term 'probability distribution' is used subsequently, with such a distribution, λ over a set Z say, it is assumed that λ always satisfies $\sum_{z \in Z} \lambda(z) = 1$.

2.2 Distributions over Formulae and Input Assignments

The expression defining $S(m, \Omega)$, the average number of evaluations carried out by Algorithm 2.1 over formulae of size m, involves two probability distributions: μ_n over assignments α to the formula arguments $\mathbf{X_n}$; and β_m over Ω-formulae of size m.

An interesting property of the average case analysis is that the value of $S(m, \Omega)$ as given by expression (2.1) above, is the same irrespective of the choice of μ_n, provided that the following two conditions hold.

S1) The basis $\Omega \subseteq B_2$ satisfies $(\theta \in \Omega) \Leftrightarrow (\hat{\theta} \in \Omega)$ (where $\hat{\theta}$ is the dual of θ, i.e. the operation defined by $x\,\theta y = \neg((\neg x)\,\theta(\neg y))$.

S2) In the distribution β_m, the probability that a gate in a random formula is chosen to have the operation $\theta \in \Omega$ is $1/|\Omega|$ for all $\theta \in \Omega$.

Some discussion of these restrictions is in order. Considering the condition S1, one weakness of this is that it prevents consideration of complete bases containing only one operation, e.g. $\Omega = \{ NAND \}$. Such bases, however, may be sub-optimal in terms of the Boolean formula realisation of some simple functions, e.g. [McColl, 1978] has shown that any n-input *symmetric* Boolean function when realised in this basis requires a circuit to have depth at least $\lceil 2\log_2 n \rceil$. For many functions in this class, e.g. the function which is 1 if at least two inputs have the value 1, formulae of depth $\lceil \log_2 n \rceil$ can be constructed using a basis satisfying the restriction S1, e.g. $\{ \wedge, \vee \}$. The second condition, S2, provides a 'natural' probability distribution with which to instantiate a given tree structure to an Ω-formula. It is worth noting, however, that in view of Lemma 3.3 below, there is be some scope for relaxing this and considering other 'natural' distributions on the probability of a given gate operation being chosen. Any such relaxation would, however, require that the probability of specific formulae arising could be expressed in a 'reasonably succinct' form in order to avoid the analysis supporting the proof of Theorem 3.1 and Section 4 becoming unmanageable. One possibility that would be consistent with these analysis would be to give $\wedge$-type operations a different weighting from $\oplus$-type, e.g. with σ the number of $\wedge$-type operations and $\omega = |\Omega|$ one could consider distributions in which the probability of choosing *any* $\wedge$-type operation is c and any $\oplus$-type $1-c$, with specific operations consistent with this choice being equally likely, i.e. $P[\,\theta \mid \theta\, chosen\ as \wedge\!-type\,] = 1/\sigma$. The model implied by S2 gives $c = \sigma/\omega$. As will be apparent from our subsequent analyses, variations in c in this context, will result in different asymptotic (or constants in the uniform tree generation model) values for the expected evaluation costs. Since it is unclear whether the additional level of notational complexity that would be introduced justifies a detailed analysis of such variations, we will concentrate on the simpler S2 model.

In view of the fact — to be proved in the next section — that with the conditions above we do not need to be specific about the distribution on input assignments, we concentrate on the probability distribution for Ω-formulae of size m. This will be described in terms of methods of generating random Ω-formulae of size m, in which case the likelihoods of specific formulae being generated define a particular probability distribution. Let $F(\mathbf{X_n})$ be an Ω-formula of size m. F is characterised by 3 attributes:

F1. The underlying *graph* structure, T_F, i.e. the manner in which the nodes and edges of F define a binary tree with m leaves.

F2. The assignment of *gate* operations from Ω to the *non-leaf* nodes of T_F.

F3. The association of input variables, x_i from $\mathbf{X_n}$, with the *leaf* nodes of T_F.

Let:

$$T_m \ =_{def} \ \{ \, T \, : \, T \ is \ an \ m-leaf \ binary \ tree \, \}$$

$$\Phi_m(T) \ =_{def} \ \{ \, \phi : \{ \, Non-leaf \ nodes \ of \ T \, \} \to \Omega \quad where \ T \in T_m \, \}$$

$$\Gamma_{n,m}(T) \ =_{def} \ \{ \, \gamma : \{ \, Leaves \ of \ T \, \} \to \mathbf{X_n} \quad where \ T \in T_m \, \}$$

So that, $\Phi_m(T)$ is the set of all mappings from the $(m-1)$ non-leaf nodes of an m-leaf binary tree T onto specific gate operations; and $\Gamma_{n,m}(T)$ the set of all mappings of the m leaves of T onto specific input variables from $\mathbf{X_n}$.

For $T \in T_m$, $\phi \in \Phi_m(T)$, and $\gamma \in \Gamma_{n,m}(T)$, $F_{\phi,\gamma}(T, \mathbf{X_n})$ will denote the Ω-formula of size m that results by instantiating T with the gate operations specified by $\phi(T)$ and input arguments from $\mathbf{X_n}$ specified by $\gamma(T)$.

With these sets we can view the process of generating a random formulae of size m as consisting of 3 distinct phases: first generate a random binary tree with m leaves, i.e. choose a random element T from T_m; second, fix a random allocation of gate operations, i.e. choose a random element of $\Phi_m(T)$; and, finally, select a random association of leaves with specific input variables, i.e. choose a random element of $\Gamma_{n,m}(T)$.

In summary, we consider probability distributions on Ω-formulae, $F(\mathbf{X_n})$ of size m, with the property that:

$$\begin{aligned} \beta_m(F(\mathbf{X_n}), \Omega) &\equiv Prob \, [\, F(\mathbf{X_n}) \ is \ chosen \,] \\ &\equiv Prob \, [\, F_{\phi,\gamma}(T, \mathbf{X_n}) \ is \ chosen \,] \\ &= \frac{\delta_m(T) \ \rho_{m,n}(\gamma)}{|\Phi_m|} \end{aligned} \tag{2.2}$$

where $\delta_m(T)$ is a probability distribution over m-leaf binary trees and $\rho_{m,n}(\gamma)$ is a probability distribution over $\Gamma_{n,m}(T)$. At the risk of appearing over-pedantic we distinguish $F(\mathbf{X_n})$ - a completely described Ω-formula, from $F_{\phi,\gamma}(T, \mathbf{X_n})$ - which is *exactly* the same formula as $F(\mathbf{X_n})$ but whose description is given more completely in terms of the underlying tree structure (T), the association of tree leaves with input variables (γ), and the mapping of non-leaf nodes onto operations (ϕ).

Thus, recalling that ω denotes the number of gate operations in Ω, so that $|\Phi_m| = \omega^{m-1}$, we may rewrite the expression for $S(m, \Omega)$ in (2.1) as,

$$\frac{1}{\omega^{m-1}} \sum_{T \in T_m} \sum_{\phi \in \Phi_m(T)} \sum_{\gamma \in \Gamma_{n,m}(T)} \sum_{\alpha \in \{0,1\}^n} \delta_m(T) \, \rho_{n,m}(\gamma) \, \mu_n(\alpha) \, W(F_{\phi,\gamma}(T, \mathbf{X_n}), \alpha) \tag{2.3}$$

In Section 3 it is shown that, as with the distribution μ_n, this expression does not vary with different choices for $\rho_{n,m}(T)$ (for all $T \in T_m$), subject to the conditions S1 and S2 described earlier. Since we have assumed that each gate operation is equally likely all that remains to be described is the method of generating a random m-leaf tree, in order to characterise the distribution $\delta_m(T)$.

Consider the following general procedure for producing a random m-leaf binary tree, in which $\eta(m)$ returns a random integer value $1 \le \eta(m) \le m-1$, according to some probability distribution ζ_m, i.e. $Prob \, [\, \eta(m) = i \,] = \zeta_m(i)$.

function Δ (m : **integer**) **return** $T \in T_m$ **is**
L, R : **binary tree**;
i : **integer**;
begin
if $m = 1$ **then**
return a single node v as result
else
$i := \eta(m)$;
$L := \Delta(i)$;
$R := \Delta(m\text{-}i)$;
return the m-leaf tree $T := L \blacklozenge R$;
-- i.e. T has left sub-tree L and right sub-tree R
end if;
end Δ;

Algorithm 2.2: Random Tree Generation Algorithm

In terms of our earlier notation, it is clear that

$$\delta_m(L \blacklozenge R) =_{def} P[\Delta(m) = L \blacklozenge R] = \zeta_m(|L|)\,\delta_{|L|}(L)\,\delta_{|R|}(R).$$

The only restriction we impose on ζ_m is that we require it to be symmetric, i.e. $\zeta_m(i) = \zeta_m(m-i)$ for all m and $1 \le i \le m-1$. Differing choices for ζ_m will yield different distributions over the set T_m of m-leaf binary trees. For example, in order to obtain the uniform distribution in which each tree was equally likely, we could use $\zeta_m(i) = |T_i|\,|T_{m-i}|/|T_m|$.

The uniform distribution is a natural choice for average-case analyses, and this will be one of the two cases examined in Section 4. There are, however, arguments against this choice arising from properties of typical trees generated by this. In particular, the average-depth of binary trees under the uniform distribution is quite large.

For this reason our subsequent analyses will also involve choosing ζ_m to be such that

$$\forall\ m \ge 2, \quad \forall\ 1 \le i \le m-1 \qquad \zeta_m(i) = \frac{1}{m-1}$$

i.e., each choice of sub-tree size (measuring size by numbers of leaves) is equally likely.

This choice gives rise to the distribution known as the *binary search tree* distribution, or *bst-distribution*, the properties of which have been extensively investigated in [Baeza-Yates *et al.* 1992], [Dunne *et al.* 1995], [Knuth 1973], and [Robson 1979]. Its chief interest, for our purposes, lies in the fact that the average depth of m-leaf trees under this distribution is $\Theta(\log m)$ as opposed to the $\Theta(m^{0.5})$ depth proved to hold for the uniform distribution in [Flajolet and Odzlyko 1982]. In practice, Boolean logic circuits tend to have depth which is small in terms of the number of input signals: the depth of a circuit being an important measure of the operating speed of a digital system. If the Boolean operations available have constant fan-in then logarithmic depth is the best attainable. We note also that formulae for which the number of gates is $O(n^k)$ for some constant k, can be restructured (without changing the basis Ω) to have depth at most $c_\Omega k \log_2 n$ where c_Ω is a constant depending on Ω, [Brent *et al.*, 1973], hence in practice it should not be necessary to consider formulae with ‘large’ depth and a ‘small’ number of gates. Such formulae are, however, are likely to be generated using the uniform distribution. Thus, we argue that the bst-distribution provides a more realistic basis for analysing average-case behaviour of the demand-driven simulation method on formulae, by virtue of the fact that this distribution biases towards trees with 'small' depth.

3 Simplification of Expression (2.3) for S(m,Ω)

Our concern in this section is to simplify expression (2.3) by eliminating the dependencies on the distributions μ_n and $\rho_{n,m}$. The following three Lemmata allow this simplification to be achieved.

Lemma 3.1: Let $\Omega \subseteq B_2$ such that $(\theta \in \Omega) \Leftrightarrow (\hat{\theta} \in \Omega)$. For $T \in T_m$, $\gamma \in \Gamma_{n,m}(T)$ and $\alpha \in \{0,1\}^n$, let

$$\Phi_\varepsilon(T, \gamma, \alpha) =_{def} \{ \phi \in \Phi_m(T) : F_{\phi,\gamma}(T, \mathbf{X_n})(\alpha) = \varepsilon \}$$

(where $\varepsilon \in \{0,1\}$) $\forall m \geq 2$, $\forall T \in T_m$, $\forall \gamma \in \Gamma_{n,m}(T)$, $\forall \alpha \in \{0,1\}^n$ it holds that:

$$| \Phi_0(T, \gamma, \alpha) | = | \Phi_1(T, \gamma, \alpha) |$$

Proof: Let $T \in T_m$, where $m \geq 2$, $\gamma \in \Gamma_{n,m}(T)$ and $\alpha \in \{0,1\}^n$. Since $m \geq 2$ it follows that $T \equiv L_i \blacklozenge R_{m-i}$ for some trees $L_i \in T_i$, $R_{m-i} \in T_{m-i}$ and $1 \leq i \leq m-1$. Consider any $\phi_L \in \Phi_i(L_i)$ and $\phi_R \in \Phi_{m-i}(R_{m-i})$. Let γ_L (respectively γ_R) denote the restriction of γ to the leaves of L_i (respectively R_{m-i}), so that $\gamma_L \in \Gamma_{n,i}(L_i)$ and $\gamma_R \in \Gamma_{n,m-i}(R_{m-i})$. Finally, let G and H denote the formulae

$$G_{\phi_L,\gamma_L}(L_i, \mathbf{X_n}) \quad ; \quad H_{\phi_R,\gamma_R}(R_{m-i}, \mathbf{X_n})$$

Under the assignment α we have

$$G(\alpha) = \alpha_G \quad ; \quad H(\alpha) = \alpha_H$$

for some $\alpha_G, \alpha_H \in \{0,1\}$.

With ϕ_L and ϕ_R fixed, only one gate operation remains to be chosen to instantiate a formula $F_{\phi,\gamma}(T, \mathbf{X_n})$. Since the assignment of gate operations to L_i and R_{m-i} has been chosen arbitrarily, in order to prove the Lemma it suffices to establish that

$$| \{ \theta \in \Omega : (\alpha_G \, \theta \, \alpha_H) = 0 \} | = | \{ \theta \in \Omega : (\alpha_G \, \theta \, \alpha_H) = 1 \} | \tag{3.1}$$

The condition on Ω required in the Lemma statement ensures that 3.1 is the case, since $(\alpha_G \, \theta \, \alpha_H) = 0$ if and only if $(\alpha_G \, \hat{\theta} \, \alpha_H) = 1$. □

In order to avoid repetition it will be assumed subsequently that $\Omega \subseteq B_2$ satisfies the condition specified in the statement of Lemma 3.1.

Lemma 3.2: Let $T \equiv (L \blacklozenge R)$ where $T \in T_m$, $L \in T_i$, and $R \in T_{m-i}$ $(1 \leq i \leq m-1)$. For $\gamma \in \Gamma_{n,m}(T)$ and $\alpha \in \{0,1\}^n$, let

$$Q(T, \gamma, \alpha) =_{def} \sum_{\phi \in \Phi_m(T)} W(F_{\phi,\gamma}(T, \mathbf{X_n}), \alpha)$$

(with $Q(T, \gamma, \alpha) = 0$ if $T \in T_1$).

Let γ_L and γ_R denote the restrictions of γ to the leaves of L and R respectively.

$\forall m \geq 2$, $\forall T \in T_m$, $\forall \gamma \in \Gamma_{n,m}(T)$, $\forall \alpha \in \{0,1\}^n$ it holds that:

$$Q(L \blacklozenge R, \gamma, \alpha) = |\Phi_m(T)| + \omega \, |\Phi_{m-i}(R)| \, Q(L, \gamma_L, \alpha) + (\omega - \sigma/2) \, |\Phi_i(L)| \, Q(R, \gamma_R, \alpha)$$

where σ is the number of $\wedge$-type functions in Ω.

Proof: Let $T \in T_m$. Since $m \geq 2$ it follows that $T \equiv L \blacklozenge R$ for trees $L \in T_i$, $R \in T_{m-i}$, and some value i with $1 \leq i \leq m$.

By definition, $Q(T, \gamma, \alpha)$

$$
\begin{aligned}
&= \sum_{\phi \in \Phi_m(T)} W(F_{\phi,\gamma}(T, \mathbf{X_n}), \alpha) \\
&= \sum_{\phi_L \in \Phi_i(L)} \sum_{\phi_R \in \Phi_{m-i}(R)} \sum_{\theta \in \Omega} W(G_{\phi_L,\gamma_L}(L, \mathbf{X_n})\, \theta H_{\phi_R,\gamma_R}(R, \mathbf{X_n}), \alpha) \\
&= \sum_{\phi_L \in \Phi_i(L)} \sum_{\phi_R \in \Phi_{m-i}(R)} \sum_{\theta \in \Omega_\oplus} (1 + W(G_{\phi_L,\gamma_L}(L, \mathbf{X_n}), \alpha) + W(H_{\phi_R,\gamma_R}(R, \mathbf{X_n}), \alpha)) \\
&\quad + \sum_{\phi_R \in \Phi_{m-i}(R)} \sum_{\theta \in \Omega_\wedge} \sum_{\phi_L \in \Phi_i(L) : G(\alpha) = c_\theta} (1 + W(G_{\phi_L,\gamma_L}(L, \mathbf{X_n}), \alpha)) \\
&\quad + \sum_{\phi_R \in \Phi_{m-i}(R)} \sum_{\theta \in \Omega_\wedge} \sum_{\phi_L \in \Phi_i(L) : G(\alpha) \neq c_\theta} (1 + W(G_{\phi_L,\gamma_L}(L, \mathbf{X_n}), \alpha) \\
&\qquad + W(H_{\phi_R,\gamma_R}(R, \mathbf{X_n}), \alpha)) \\
&= |\Phi_m(T)| + \omega |\Phi_{m-i}(R)| \sum_{\phi_L \in \Phi_i(L)} W(G_{\phi_L,\gamma_L}(L, \mathbf{X_n}), \alpha) \\
&\quad + |\Phi_i(L)| \sum_{\theta \in \Omega_\oplus} \sum_{\phi_R \in \Phi_{m-i}(R)} W(H_{\phi_R,\gamma_R}(R, \mathbf{X_n}), \alpha) \\
&\quad + \sum_{\theta \in \Omega_\wedge} \sum_{\phi_L \in \Phi_i(L) : G(\alpha) \neq c_\theta} \sum_{\phi_R \in \Phi_{m-i}(R)} W(H_{\phi_R,\gamma_R}(R, \mathbf{X_n}), \alpha) \\
&= |\Phi_m(T)| + \omega |\Phi_{m-i}(R)| Q(L, \gamma_L, \alpha) + (\omega - \sigma) |\Phi_i(L)| Q(R, \gamma_R, \alpha) \\
&\quad + (\sigma |\Phi_i(L)|/2) Q(R, \gamma_R, \alpha)
\end{aligned} \tag{3.2}
$$

The expression (3.2) following from Lemma 3.1, since for fixed γ_L and α and a given $\wedge$-type operation θ, there are exactly $|\Phi_i(L)|/2$ instantiations of $G_{\phi_L,\gamma_L}(L, \mathbf{X_n})$ such that $G(\alpha) \neq c_\theta$.

A simple re-arrangement of (3.2) gives,

$$
\begin{aligned}
Q(L \blacklozenge R, \gamma, \alpha) = |\Phi_m(T)| &+ \omega |\Phi_{m-i}(R)| Q(L, \gamma_L, \alpha) \\
&+ (\omega - \sigma/2) |\Phi_i(L)| Q(R, \gamma_R, \alpha)
\end{aligned}
$$

as required. □

The quantity $Q(T, \gamma, \alpha)$ introduced in Lemma 3.2, can be interpreted in the following way: fix a particular m leaf tree structure (T), a mapping, γ, from the leaves of T to variables in $\mathbf{X_n}$, and an assignment, α, of Boolean values to these variables. Then, $Q(T, \gamma, \alpha)$ is the total number of steps taken by Algorithm 2.1, to evaluate $F_{\phi,\gamma}(T, \alpha)$ summed over *all* mappings, ϕ of gate operations onto non-leaf nodes of T. The key point of the next Lemma is in showing that the quantity $Q(T, \gamma, \alpha)$ depends neither on the specific mapping of leaf nodes to variables, nor on the specific instantiation of the variables defined by α, i.e. that the value of $Q(T, \gamma, \alpha)$ is the same for all choices of γ and all choices of α. This shows that the significant influences on expected evaluation costs in our model are the likelihoods of a particular tree structure and assignment of gate operations occurring.

Lemma 3.3: $\forall\, m \geq 1,\ \forall\, T \in T_m,\ \forall\, \gamma_1, \gamma_2 \in \Gamma_{n,m}(T),\ \forall\, \alpha_1, \alpha_2 \in \{0,1\}^n$, it holds that

$$Q(T, \gamma_1, \alpha_1) = Q(T, \gamma_2, \alpha_2)$$

Proof: By induction on $m \geq 1$. The base case is immediate since in this instance $Q(T, \gamma, \alpha) = 0$ for all γ and α.

Assuming that the Lemma holds for all values $k < m$, we shall prove it holds for m also. Let $T \in T_m$ and γ_1, γ_2, α_1, and α_2 be as in the Lemma statement. Since $m \geq 2$, $T = L \blacklozenge R$ for some trees $L \in T_i$ and $R \in T_{m-i}$. For $j=1$ and $j=2$, let $\gamma_{j,L}$ and $\gamma_{j,R}$ denote the restrictions of γ_j to the leaves of L and R respectively. From Lemma 3.2,

$$Q(L \blacklozenge R, \gamma_1, \alpha_1) = |\Phi_m(T)| + \omega |\Phi_{m-i}(R)| \, Q(L, \gamma_{1,L}, \alpha_1)$$
$$+ (\omega - \sigma/2) |\Phi_i(L)| \, Q(R, \gamma_{1,R}, \alpha_1)$$

Applying the Inductive Hypothesis, shows that this is

$$= |\Phi_m(T)| + \omega |\Phi_{m-i}(R)| \, Q(L, \gamma_{2,L}, \alpha_2) + (\omega - \sigma/2) |\Phi_i(L)| \, Q(R, \gamma_{2,R}, \alpha_2)$$

$$= Q(L \blacklozenge R, \gamma_2, \alpha_2)$$

from Lemma 3.2. □

As a consequence of the result of Lemma 3.3, we shall write $Q(T)$ rather than $Q(T, \gamma, \alpha)$. The main result of this section is now given in,

Theorem 3.1:

$$\forall\ m \geq 2 \qquad S(m, \Omega) = \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T) \, Q(T)$$

(Note: that $S(1, \Omega) = 0$ is immediate from the definition of $W(F, \alpha)$ when $|F| = 1$).

Proof: In expression (2.3) we have that, $S(m, \Omega)$

$$= \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \sum_{\phi \in \Phi_m(T)} \sum_{\gamma \in \Gamma_{n,m}(T)} \sum_{\alpha \in \{0,1\}^n} \delta_m(T) \, \rho_{n,m}(\gamma) \, \mu_n(\alpha) \, W(F_{\phi,\gamma}(T, \mathbf{X_n}), \alpha)$$

$$= \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T) \sum_{\gamma \in \Gamma_{n,m}(T)} \sum_{\alpha \in \{0,1\}^n} \rho_{n,m}(\gamma) \, \mu_n(\alpha) \sum_{\phi \in \Phi_m(T)} W(F_{\phi,\gamma}(T, \mathbf{X_n}), \alpha)$$

$$= \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T) \sum_{\gamma \in \Gamma_{n,m}(T)} \rho_{n,m}(\gamma) \sum_{\alpha \in \{0,1\}^n} \mu_n(\alpha) \, Q(T, \gamma, \alpha) \qquad (3.3)$$

From Lemma 3.3 and the notational convention introduced following the proof of this, (3.3) is

$$= \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T) \, Q(T) \sum_{\gamma \in \Gamma_{n,m}(T)} \rho_{n,m}(\gamma) \sum_{\alpha \in \{0,1\}^n} \mu_n(\alpha)$$

$$= \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T) \, Q(T)$$

from the fact that μ_n and $\rho_{n,m}$ are probability distributions. □

4 Analysis of Average-Case Complexity

In this section the main results of the paper are presented. We analyse the behaviour of the demand-driven simulation approach given in Algorithm 2.1 and determine its expected running time for Ω-formulae of size m. In Section 3, we have shown that this quantity $S(m,\Omega)$ is given by the expressions:

$$S(1,\Omega) \; = \; 0 \tag{4.1}$$

$$S(m,\Omega) \; = \; \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T)\, Q(T) \tag{4.2}$$

If $m \geq 2$, then $T = L \blacklozenge R$ for $L \in T_i$ and $R \in T_{m-i}$ $(1 \leq i \leq m-1)$. By virtue of Lemma 3.2, Lemma 3.3, and the fact that $|\,\Phi_m(T)\,| = \omega^{m-1}$, the function $Q(L \blacklozenge R)$ satisfies the relationship,

$$Q(L \blacklozenge R) \; = \; \omega^{m-1} + \omega^{m-i}\, Q(L) + (\omega - \sigma/2)\, \omega^{j-1}\, Q(R) \tag{4.3}$$

Finally we recall that δ_m is a probability distribution over the set of m-leaf binary trees, which satisfies

$$\delta_m(T) \; = \; \begin{cases} 1 & \text{if} \quad m = 1 \\ \zeta_m(i)\, \delta_i(L)\, \delta_{m-i}(R) & \text{if} \quad m \geq 2, \quad T = L \blacklozenge R \end{cases} \tag{4.4}$$

where ζ_m is a (symmetric) probability distribution over the integers $\{1, 2, \ldots, m-1\}$.

We use (4.1-4.4) to derive a preliminary recurrence relationship governing the behaviour of $S(m, \Omega)$.

Lemma 4.1: For all probability distributions δ_m over T_m that satisfy the relationship given in (4.4), the relationship of (4.2) for $S(m, \Omega)$ may be written,

$$S(m,\Omega) \; = \; 1 + \frac{4\omega - \sigma}{2\omega} \sum_{i=1}^{m-1} \zeta_m(i)\, S(i,\Omega)$$

(recalling that σ is the number of $\wedge$-type operations in the basis Ω).

Proof: From (4.2)

$$S(m,\Omega) \; = \; \frac{1}{\omega^{m-1}} \sum_{T \in T_m} \delta_m(T)\, Q(T)$$

$$= \; \frac{1}{\omega^{m-1}} \sum_{i=1}^{m-1} \sum_{L \in T_i} \sum_{R \in T_{m-i}} \delta_m(L \blacklozenge R)\, Q(L \blacklozenge R)$$

$$= \; 1 + \frac{1}{\omega^{m-1}} \sum_{i=1}^{m-1} \zeta_m(i) \sum_{R \in T_{m-i}} \delta_{m-i}(R) \sum_{L \in T_i} \delta_i(L)\, \omega^{m-i}\, Q(L)$$

$$+ \; \frac{1}{\omega^{m-1}} \sum_{i=1}^{m-1} \zeta_m(i) \sum_{L \in T_i} \delta_i(L) \sum_{R \in T_{m-i}} \delta_{m-i}(R)\, (\omega - \sigma/2)\, \omega^{j-1}\, Q(R) \qquad \text{(via 4.3)}$$

$$= \; 1 + \sum_{i=1}^{m-1} \frac{\zeta_m(i)}{\omega^{j-1}} \sum_{L \in T_i} \delta_i(L)\, Q(L) + \frac{2\omega - \sigma}{2\omega} \sum_{i=1}^{m-1} \frac{\zeta_m(i)}{\omega^{m-i-1}} \sum_{R \in T_{m-i}} \delta_{m-i}(R)\, Q(R)$$

$$= \; 1 + \sum_{i=1}^{m-1} \zeta_m(i) S(i,\Omega) + \frac{2\omega - \sigma}{2\omega} \sum_{i=1}^{m-1} \zeta_m(i) S(m-i,\Omega) \; = \; 1 + \frac{4\omega - \sigma}{2\omega} \sum_{i=1}^{m-1} \zeta_m(i) S(i,\Omega) \qquad \square$$

4.2 S(m,Ω) under the Binary Search Tree Distribution

For the distribution over m-leaf binary trees defined by the bst-distribution, we have $\zeta_m(i)=1/(m-1)$, for all $1 \le i \le m-1$. Applying the relevant substitution to the recurrence proved in Lemma 4.1, we obtain for this case, for $m \ge 2$

$$S(m,\Omega) = 1 + \frac{4\omega-\sigma}{2\omega(m-1)} \sum_{i=1}^{m-1} S(i,\Omega) \tag{4.5}$$

To put (4.5) into a more manageable form, we introduce a function $H(m)$ defined by

$$H(m) =_{def} \sum_{i=1}^{m} S(i,\Omega)$$

With this function, obviously, $S(m,\Omega)=H(m)-H(m-1)$, and so in order to determine the asymptotic behaviour of $S(m,\Omega)$ it suffices to determine similar behaviour for $H(m)$. Combining (4.5) with the definition of $H(m)$ results in a considerably simpler recurrence relation.

Lemma 4.2:

$$H(m) = \begin{cases} 0 & \text{if } m=1 \\ 1 + \dfrac{2\omega(m+1)-\sigma}{2\omega(m-1)} H(m-1) & \text{if } m > 1 \end{cases}$$

Proof: Immediate from substituting the definition for $H(m)$ into (4.5) and re-arranging the resulting expression. □.

Theorem 4.1: Let $\Omega \subseteq B_2$ with $\omega=|\Omega|$ and σ denoting the number of $\wedge$-type operations in Ω. Let $\tau = 2-\sigma/2\omega$.

$$H(m) \approx 1 + K_\Omega (m-1)^\tau$$

where K_Ω is a constant depending on Ω.

Proof: From Lemma 4.2, if $m \ge 2$

$$\begin{aligned} H(m) &= 1 + \left[1+\frac{\tau}{m-1} \right] H(m-1) \\ &= 1 + \sum_{k=1}^{i-1} \prod_{j=1}^{k} \left[1+\frac{\tau}{m-j} \right] + \prod_{j=1}^{i} \left[1+\frac{\tau}{m-j} \right] H(m-i) \\ &= 1 + \sum_{k=1}^{m-2} \prod_{j=1}^{k} \left[1+\frac{\tau}{m-j} \right] = 1 + \sum_{k=1}^{m-2} \prod_{j=m-k}^{m-1} \frac{j+\tau}{j} \\ &= 1 + \frac{\Gamma(m+\tau)}{\Gamma(m)} \sum_{k=1}^{m-2} \frac{\Gamma(k+1)}{\Gamma(k+\tau+1)} \end{aligned} \tag{4.6}$$

where $\Gamma(x)$ if the Γ-function, defined for $x \in \mathbf{R}^+$, by $\Gamma(x+1)=x\,\Gamma(x)$.

Applying Stirling's approximation — $\Gamma(x+1) \approx (x^x/\exp(x))(2\pi x)^{0.5}$ — and simplifying the resulting form of (4.6), gives

$$H(m) \approx 1 + \frac{(m-1)^\tau}{\exp(\tau)} \left[1+\frac{\tau}{m-1} \right]^{m+\tau-0.5} \sum_{k=1}^{m-2} \frac{\Gamma(k+1)}{\Gamma(k+\tau+1)}$$

To further simplify this, we observe that

$$1 \;<\; \left[\; 1+\frac{\tau}{m-1} \;\right]^{m+\tau-0.5} \;\lesssim\; \exp(\,\tau+0.5\,)$$

Finally, the summation

$$\sum_{k=1}^{m-2}\frac{\Gamma(\,k+1\,)}{\Gamma(\,k+\tau+1\,)} \;\approx\; \sum_{k=1}^{m-2}\frac{1}{k^{\eta}}$$

for some constant $\eta > 1$. Thus this sum is constant bounded for all m. It follows that there is a constant $K_{\Omega} \in \mathbf{R}$ such that

$$H(\,m\,) \;\approx\; 1 + K_{\Omega}\,(\,m-1\,)^{\tau} \qquad \square$$

Theorem 4.1., gives an asymptotic estimate of the function $H(\,m\,)$. This can now be applied to give the main result of this section.

Theorem 4.2: Let $\Omega \subseteq B_2$ be as in Theorem 4.1. $S(\,m,\,\Omega\,)$, the expected running time of the demand-driven simulation method in Algorithm 2.1, when averaged over formulae of size m under the binary search tree distribution, satisfies

$$S(\,m,\,\Omega\,) \;=\; O(\,(m-1)^{\tau-1}\,)$$

Proof: The function $H(\,m\,)$ introduced at the start of Section 4.1, satisfies for $m \geq 2$

$$S(\,m,\,\Omega\,) \;=\; H(\,m\,) - H(\,m-1\,)$$

From Theorem 4.1, this yields

$$S(\,m,\,\Omega\,) \;\approx\; K_{\Omega}\,(\,(m-1)^{\tau} - (m-2)^{\tau}\,)$$

which by elementary analysis is $O(\,(m-1)^{\tau-1}\,)$. $\square$

Some specific special cases are presented in the following.

Corollary 4.1:

i. $S(\,m,\,\{\,\wedge,\vee,\neg\wedge,\neg\vee\,\}\,) = O(\,(\,m-1\,)^{0.5}\,)$.

ii. $S(\,m,\,\hat{B}_2\,) = O(\,(m-1)^{0.6}\,)$, where $\hat{B}_2$ consists of those binary Boolean function that depend on both arguments, i.e. constant functions and projections are excluded.

iii. $S(\,m,\,\{\,\oplus,\neg\oplus\,\}\,) = O(\,m\,)$

Proof: For (i) we have $\omega=\sigma=4$; for (ii) $\omega=10$ and $\sigma=8$; and for (iii), $\sigma=0$. It may be noted that an exact result for (iii) may be obtained directly from the recurrences of Lemma 4.2. $\square$

4.3 S(m,Ω) under the Uniform Distribution

It has been shown, in Lemma 4.1, that the expected evaluation complexity of Ω-formula of size m, is governed by the recurrence

$$S(\,m,\,\Omega\,) \;=\; 1 + \frac{4\omega-\sigma}{2\omega}\sum_{i=1}^{m-1}\zeta_m(\,i\,)\,S(\,i,\,\Omega\,) \qquad (4.7)$$

where $\omega = |\,\Omega\,|$, $\sigma = \{\,\theta\in\Omega : \theta \textit{ is } \wedge\textit{-type}\,\}$ and ζ_m is a probability distribution over $\{\,1,2,\,.\,.\,.\,,\,m-1\,\}$. In Section 4.1, we determined an asymptotic bound for the choice of ζ_m that leads to the bst-distribution. In this section we establish bounds on $S(m, \Omega)$ for the choice of ζ_m that yields the uniform distribution on m-leaf binary trees, i.e. the distribution under which each m-leaf binary tree is equally likely to be chosen.

It is easily seen that to realise the uniform distribution, ζ_m, must satisfy

$$\zeta_m(i) = \frac{t_i\, t_{m-i}}{t_m} \quad \forall\ 1 \le i \le m-1 \tag{4.8}$$

where t_m denotes $|T_m|$, the number of m-leaf binary trees. It is well-known that for all $m \ge 1$

$$t_m = \frac{1}{4m-2} \left[\begin{array}{c} 2m \\ m \end{array} \right] \tag{4.9}$$

The behaviour of ζ_m for the uniform distribution is clearly quite different from the choice for the bst-distribution. Nevertheless, it turns out that the resulting form of (4.7) is susceptible to precise analysis. In order to reduce the notational complexities involved in this analysis we make use of the following notations. Throughout $\zeta_m(i)$ is as defined by the relationships (4.8) and (4.9) above.

$\tau =_{def} (4\omega - \sigma)/(2\omega)$, as previously.

$\zeta_\infty(i) =_{def} \lim_{m \to \infty} \zeta_m(i)$.

$\Lambda(m, r) =_{def} \sum_{i=1}^{r} \zeta_m(i)\, S(i, \Omega)$, where $r < m$.

$\Lambda(\infty, r) =_{def} \sum_{i=1}^{r} \zeta_\infty(i)\, S(i, \Omega)$.

We first establish some basic facts about these functions.

Lemma 4.3:

i) $\forall m, \forall\ 1 \le i \le m-1,\ \zeta_{m+1}(i) < \zeta_m(i)$.

ii) $\zeta_\infty(i) = t_i/4^i$, where t_m is the number of m-leaf binary trees as given in expression (4.9) above.

iii) $\Lambda(m+1, r) < \Lambda(m, r)$.

iv) $\lim_{r \to \infty} \sum_{i=1}^{r} \zeta_\infty(i) = 0.5$

Proof: For (i) we have by definition that

$$\zeta_{m+1}(i) = \frac{t_i\, t_{m+1-i}}{t_{m+1}} \quad ; \quad \zeta_m(i) = \frac{t_i\, t_{m-i}}{t_m}$$

In order to prove (i), it is sufficient to establish that $\zeta_{m+1}(i)/\zeta_m(i) < 1$.

$$\frac{\zeta_{m+1}(i)}{\zeta_m(i)} = \frac{t_m\, t_{m+1-i}}{t_{m+1}\, t_{m-i}} = \frac{2m^2 - 2im + m - 2i - 1}{2m^2 - 2im + m - i - 1}$$

and this is less than 1, so proving part (i).

For part (ii)

$$\zeta_\infty(i) = \lim_{m \to \infty} \zeta_m(i)$$

$$= \lim_{m \to \infty} \frac{t_i\, t_{m-i}}{t_m}$$

$$= t_i \lim_{m\to\infty} \frac{(4m-2)\,(2m-2i)!\,(m\,!)^2}{(4m-4i-2)\,(2m)!\,((m-i)!)^2}$$

From Stirling's approximation, $\lim_{n\to\infty} n\,! = (n/e)^n\,(2\pi n)^{0.5}$, the right-hand side of this is

$$= \frac{t_i}{4^i} \lim_{m\to\infty} \left[\frac{m}{m-i} \right]^{0.5} = \frac{t_i}{4^i}$$

as required.

Part (iii) is immediate from (i) and the definition of $\Lambda(m,r)$.

For part (iv), we first observe that $Z(r) = \sum_{i=1}^{r} \zeta_\infty(i)$ is a strictly monotone increasing function of r and thus any limit as r tends to ∞ is approached from below. To establish this part of the Lemma, it suffices to show that $\sum_{n=1}^{\infty} \zeta_\infty(n) = 0.5$.

Let $A_n =_{def} 4^{-n} \left[\begin{matrix} 2n \\ n \end{matrix} \right]$. From part (ii) of the Lemma,

$$\sum_{n=1}^{\infty} \zeta_\infty(n) = \sum_{n=1}^{\infty} \frac{t_n}{4^n} = \sum_{n=1}^{\infty} \frac{A_n}{2(2n-1)}$$

$$= \sum_{n=1}^{\infty} \frac{A_n}{4n^2-1} + \frac{1}{2}\sum_{n=1}^{\infty} \frac{A_n}{2n+1} = 1 - \frac{\pi}{4} + \frac{1}{2}\left[\frac{\pi}{2} - 1 \right]$$

$$= 0.5$$

The sums of the two infinite series may be found in [Jolly 1961] (series nos. 387 and 388) from which our notation A_n is taken. □

Theorem 4.3 Let Ω be such that $\sigma > 0$, i.e. Ω contains some $\wedge$-type functions. Let $g : \mathbf{N} \to \mathbf{N}$ be any strictly monotone increasing function that satisfies $g(n) \geq n+1$. Then for any such function $g(n)$, under the uniform distribution the following holds of $S(m, \Omega)$:

$\forall\, 0 < \varepsilon < \dfrac{\sigma}{2(4\omega-\sigma)}$, $\exists R_\varepsilon \in \mathbf{N}$, such that $\forall\, m \geq g(R_\varepsilon)$

$$S(m, \Omega) < \frac{4\omega}{\sigma - 2\varepsilon(4\omega-\sigma)} \left[1 + \frac{4\omega-\sigma}{2\omega} \Lambda(g(R_\varepsilon), R_\varepsilon) \right] + o(1)$$

$$S(m, \Omega) > \frac{4\omega}{\sigma} \left[1 + \frac{4\omega-\sigma}{2\omega} \Lambda(\infty, R_\varepsilon) \right] - o(1)$$

Proof: For the upper bound, we have

$$S(m, \Omega) = 1 + \tau \sum_{i=1}^{m-1} \zeta_m(i)\, S(i, \Omega)$$

$$= (1 + \tau\Lambda(m,r)) + \tau \sum_{i=r+1}^{m-1} \zeta_m(i)\, S(i)$$

$$\leq (1 + \tau\Lambda(m,r)) + \tau(1 - \sum_{i=1}^{r} \zeta_\infty(i))\, S(m-1)$$

From part (iv) of Lemma 4.3, $\forall\, \varepsilon > 0$, $\exists\, R_\varepsilon$, such that $\forall\, r \ge R_\varepsilon$, $\sum_{i=1}^{r} \zeta_\infty(i) \ge 0.5 - \varepsilon$. Thus if $m \ge g(R_\varepsilon) > R_\varepsilon$

$$\begin{aligned} S(m, \Omega) &\le (1 + \tau\Lambda(m, R_\varepsilon)) + \tau(0.5+\varepsilon) S(m-1) \\ &< (1 + \tau\Lambda(g(R_\varepsilon), R_\varepsilon)) + \tau(0.5+\varepsilon) S(m-1) \end{aligned}$$

from part (iii) of Lemma 4.3.

The term $1 + \tau\Lambda(g(R_\varepsilon), R_\varepsilon)$ is a constant depending only on the value ε. Furthermore if ε is chosen so that $\varepsilon < \sigma/(8\omega - 2\sigma)$ then

$$\lambda_\varepsilon =_{def} \tau(0.5+\varepsilon)$$

is less than 1. In which case we obtain the recurrence,

$$S(m, \Omega) < (1 + \tau\Lambda(g(R_\varepsilon), R_\varepsilon)) + \lambda_\varepsilon S(m-1)$$

which is valid for all $m \ge g(R_\varepsilon)$. Solving this recurrence gives,

$$S(m, \Omega) < (1 + \tau\Lambda(g(R_\varepsilon), R_\varepsilon)) \left[\frac{1 - \lambda_\varepsilon^{m-g(R_\varepsilon)}}{1 - \lambda_\varepsilon} \right] + \lambda_\varepsilon^{m-g(R_\varepsilon)} S(g(R_\varepsilon), \Omega)$$

Substituting $\lambda_\varepsilon = \tau(0.5+\varepsilon)$, $\tau = (4\omega - \sigma)/2\omega$ and simplifying this expression, we obtain

$$S(m, \Omega) < \frac{4\omega}{\sigma - 2\varepsilon(4\omega - \sigma)} \left[1 + \frac{4\omega - \sigma}{2\omega} \Lambda(g(R_\varepsilon), R_\varepsilon) \right] + o(1)$$

For the lower bound,

$$\begin{aligned} S(m, \Omega) &= 1 + \tau \sum_{i=1}^{m-1} \zeta_m(i) S(i, \Omega) \\ &> (1 + \tau\Lambda(m, R_\varepsilon)) + \frac{\tau}{2} S\left[\frac{m-1}{2} \right] \\ &> (1 + \tau\Lambda(\infty, R_\varepsilon)) + \frac{\tau}{2} S\left[\frac{m-1}{2} \right] \\ &> \frac{4\omega}{\sigma} \left[1 + \frac{4\omega - \sigma}{2\omega} \Lambda(\infty, R_\varepsilon) \right] - o(1) \end{aligned}$$ □

With the basis $\Omega = \{\wedge, \vee, \neg\wedge, \neg\vee\}$, for which $\omega = \sigma = 4$, by explicit calculation we obtain the following table:

m	ε	$\Lambda(m^2, m)$	Upper	Lower	$S(m^2)$
10	0.08810	0.36269	13.10148	6.00959	8.79662
20	0.06269	0.48628	11.08798	6.83040	9.65762
30	0.05129	0.55672	10.60334	7.28271	9.84362
40	0.04446	0.60396	10.39771	7.58155	9.91116

Table 1: Behaviour of $S(m, \{\wedge, \vee, \neg\wedge, \neg\vee\})$ in the Uniform Distribution.

Similarly, with the basis $\Omega = \hat{B}_2$, for which $\omega = 10$ and $\sigma = 8$, we get

m	ε	$\Lambda(m^2, m)$	Upper	Lower	$S(m^2)$
10	0.08810	0.38701	27.42472	7.85423	12.21727
20	0.06269	0.53554	18.62393	9.15125	14.14229
30	0.05129	0.62424	16.94783	9.90333	14.60016
40	0.04446	0.68545	16.27163	10.41578	14.77108

Table 2: Behaviour of $S(m, \hat{B}_2)$ in the Uniform Distribution.

From which we conclude,

Corollary 4.2: $\forall\, m \geq 1600$, under the uniform distribution over m-leaf binary trees, the expected evaluation complexity of Ω-formula of size m satisfies:

i. $9.91116 < S(m, \Omega) < 10.39771$, if $\Omega = \{\wedge, \vee, \neg\wedge, \neg\vee\}$.

ii. $14.77108 < S(m, \Omega) < 16.27163$, if $\Omega = \hat{B}_2$.

Proof: Immediate from Theorem 4.3, and the two tables above. The lower bounds being taken from the final column of each table. □

5 Discussion and Further Work

Our principal aim in this paper has been to present an analytic study of the typical behaviour of a particular paradigm for digital simulation: demand driven lazy evaluation. In this section we discuss the limitations of these results and outline potential ideas for their development.

We can identify two immediate limitations as regards the general applicability of our analyses: the condition S1 which sets quite stringent conditions on the set of operations considered for gates, and the fact that we deal with formulae rather than logic circuits, i.e. gates have fanout restricted to 1.

As regards the restriction on gate operations, while this appears to rule out rather more complex structures, e.g. multiplexors, in practice these could be simulated at the level of some 2-input gate realisation: the restriction does not exclude a logically complete basis such as $\{$ *NAND*, *NOR* $\}$. Of course, such an approach would mean that a realisation is considered at a lower level of granularity than might be considered in a 'standard' simulation environment, however, given that *some* form of restriction will be inevitable in formulating an analytically tractable model, it may well be the case that similar objections can be raised for other conditions. It should be noted that the condition used does admit the set of all 2-input Boolean operations.

Extending the analytic approach to general circuits presents significant technical problems in view of the fact that the model of information flow existing in formulae, i.e. that each gate provides an input for exactly one other gate, is not valid in a circuit context. This is not a problem in terms of a practical implementation of the paradigm but it is unclear to what extent the analysis of formulae extends to consideration of circuits. Experimental implementation of the demand-driven paradigm with randomly generated circuits, albeit in a multiprocessor distributed context, do provide some support for the analytic study, e.g. [Gittings, 1992], [Dunne, *et al.*, 1997].

Finally, it is worth considering to what extent similar methods can be employed in formulating a detailed analytic study of event-driven methods. The specific drawbacks of such approaches, in terms of redundant calculation, have been noted by, among others, [Smith *et al.*, 1987] and [Jennings, 1991]. The range of event-scheduling mechanisms that could be employed in implementing a specific approach, creates difficulties in attempting sensibly to define an event-driven model.

6 Conclusions

In this paper we have considered various performance characteristics of the demand-driven simulation paradigm. We have analysed how a specific algorithm within this class, performs on average when evaluating Boolean formulae of size m. Among the features of interest arising from these analyses, are that the performance measures obtained depend only on the logical bases from which formulae operations are chosen and the probability distribution used to determining the likelihood of particular m-leaf binary trees defining the graph-structure of a formula occurring. Two such distributions have been examined: the binary search tree distribution, for which an average case evaluation complexity bound of $O(m^{0.5})$ was obtained in the case of logical bases containing only $\wedge$-type operations; and the uniform distribution over binary trees, for which it has been demonstrated that the average case evaluation complexity of size m Ω-formulae is bounded above by a constant depending only on Ω, provided that Ω contains some $\wedge$-type functions. These results provide analytic quantification of the performance achievable with the demand-driven approach.

References

[Baeza-Yates *et al.* 1992] Baeza-Yates, R., Casas, R., Díaz, J., and Martínez, J. "On the average size of the intersection of binary trees," *SIAM Jnl. of Computing*, vol. 21, no. 1, pp. 24-32, 1992

[Brent *et al.* 1973] Brent, R., Kuck, D.J., Maruyama, K. "The parallel evaluation of arithmetic expressions without division", *IEEE Trans. Comp.*, C-22, pp. 532-534, 1973

[Charlton *et al.* 1991] Charlton, C.C., Jackson, D., and Leng, P.H. "Lazy simulation of digital logic," *Computer Aided Design*, vol. 23, no. 7, pp. 506-513, 1991

[Dunne and Leng 1992] Dunne, P.E., and Leng, P.H. "An algorithm for optimising signal selection in demand-driven circuit simulation," *Transactions of the Society for Computer Simulation*, vol. 8, no.4, pp. 269-280, 1992

[Dunne *et al.* 1995] Dunne, P.E., Gittings, C.J., and Leng, P.H. "Multiprocessor simulation strategies with optimal speed-up," *Inf. Proc. Letters*, vol. 54, no. 1, pp. 23-33, April 1995

[Dunne *et al.* 1997] Dunne, P.E, Leng, P.H., and Nwana, G. "Demand-driven logic simulation using a network of loosely coupled processors", (submitted)

[Flajolet and Odlyzko 1982] Flajolet, P. and Odlyzko, A. "The average height of binary trees and other simple trees," *Jnl. of Comp. and Syst. Sci.*, vol. 25, pp. 171-213, 1982

[Gittings, 1992] Gittings, C.J. "Parallel demand-driven simulation of logic networks", Ph.D. Dissertation, Univ. of Liverpool, 1992

[Henderson and Morris 1976] Henderson, P. and Morris, J. "A lazy evaluator," *Proc. 3rd ACM Symp. on Principles of Programming Languages*, pp. 95-103, 1976

[Jackson 1986] Jackson, D. "The generation of systems for microarchitectural simulation," Ph.D Dissertation, Univ. of Liverpool, 1986

[Jackson *et al.* 1987] Jackson, D., Charlton, C.C., and Leng, P.H. "Modelling and simulation of digital logic: an alternative approach," *Proc. 16th IASTED Int. Symp., 'Identification, Modelling and Simulation'*, pp. 86-90, Paris, 1987

[Jakoby *et al.* 1994] Jakoby, A., Reischuk, R., and Schindelhauer, C. "Circuit Complexity: from the Worst Case to the Average Case," *Proc. 26th ACM Symp. on Theory of Comp.*, pp. 58-67, 1994

[Jennings 1991] Jenning, G. "A case against event-driven simulation for digital system design", *Proc. 24th IEEE Annual Simulation Symposium*, IEEE, pp. 170-176, 1991

[Jolley 1961] Jolley, L.B.W. "*Summation of series*", (2nd revised edition), Dover Publications, New York, 1961

[Knuth 1973] Knuth, D.E. "*The Art of Computer Programming: Sorting and Searching*", Addison-Wesley, 1973

[McColl, 1978] McColl, W.F. "The circuit depth of symmetric Boolean functions", *Jnl. of Comp. and Syst. Sci.*, vol. 17, pp. 108-115, 1978

[Robson 1979] Robson, J.M. "The height of binary search trees," *Austral. Comput. Jnl.*, vol. 11, pp. 151-153, 1979

[Smith *et al.* 1987] Smith, S.P., Mercer, M.R., and Brock, B. "Demand driven simulation: BACKSIM," *Proc. 24th ACM/IEEE Design Automation Conference*, pp. 181-187, 1987

[Szygenda and Thompson 1975] Szygenda S.A. and Thompson, E.W. "Digital logic simulation in a time-based, table-driven environment, Part 1: Design verification," Computer, pp. 24-36, 1975

[Ulrich 1969] Ulrich, E.G. "Exclusive simulation of activity in digital networks," *Comm. ACM*, vol. 12, no. 2, pp. 102-110, 1969