Under consideration for publication in Formal Aspects of Computing

But What If I Don't Want To Wait Forever?

Colin Fidge[1] and Carron Shankland[2]

[1]Software Verification Research Centre, The University of Queensland, Australia
[2]Department of Computing Science and Mathematics, University of Stirling, United Kingdom

Keywords: IEEE 1394; IEEE 1394a; Formal specification; Verification; Probabilistic reasoning

Abstract.
We present an abstract model of the leader election protocol used in the IEEE 1394 High Performance Serial Bus standard. The model is expressed in the *probabilistic Guarded Command Language*. By formal reasoning based on this description, we establish the probability of the root contention part of the protocol successfully terminating in terms of the number of attempts to do so. Some simple calculations then allow us to establish an upper bound on the time taken for those attempts.

1. Introduction

Traditional computing formalisms allow us to reason about the functional properties of algorithms. Here we show how the *probabilistic Guarded Command Language* additionally allows us to determine the likelihood of an algorithm successfully terminating within a certain number of iterations, where the computation's progress relies on a random choice.

We focus on a specific aspect of the IEEE 1394 'FireWire' standard [Ins95, Ins00], namely *contention resolution* in the *network leader election* protocol of the Tree Identify Phase. The leader election protocol requires a set of asynchronously communicating nodes to elect a unique leader. In certain cases two nodes may simultaneously nominate one another as leader. The contention resolution algorithm solves this by having each node make a choice based on a randomly generated value. However, this choice may result in re-entering the contentious state, so it is not immediately obvious that contention will always be resolved.

It is already well known that the root contention resolution algorithm is correct. A number of previous studies have verified that the protocol will terminate successfully *eventually* [SV99]. In practice, however, a bounded solution would be more useful: for *how many* attempts will we have to wait before a leader is elected, or, given that the user has no real feedback as to the number of attempts, for *how long* will we have to wait? Here we formally derive a relationship between the number of attempts to resolve contention and the probability of a successful outcome in a particular worst-case scenario. Moreover, we use the timing

Correspondence and offprint requests to: Carron Shankland, Department of Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, United Kingdom. E-mail: ces@cs.stir.ac.uk

parameters of the amendment to the IEEE standard [Ins95] to determine an upper bound on the elapsed time, based on number of attempts.

2. Related Work

There have already been many correctness proofs for various aspects of the FireWire protocol at various levels of abstraction. Romijn [Rom01] and Stoelinga [Sto02] summarise several recent papers in the field. The studies closest to ours are those that focus on the protocol's probabilistic or timing properties.

Tools-based approaches aim to automatically explore the state space defined by the protocol. For instance, automata and the tools HyTech and Prism [KNS02] were used to establish an absolute deadline for successful termination of the protocol. Similarly, automata and the model checker LPMC [BSdRT01] were used to find bounds for the timing parameters within which the protocol can be successful. The Upaal2k tool was used to explore multiple instances of the protocol using different timing values in order to identify the timing constraints necessary for correct operation [SS01]. In a quite different approach, simulation based on a stochastic process algebra [D'A99] was used to produce a probability density function, answering the question "on *average*, how many attempts to solve contention are required?" However, all approaches based on state space exploration risk being limited by the size of the state machine which can be explored. For example, deadlines initially up to only 6μs were analysable in the first of these studies [KNS02], although larger deadlines of up to 1ms were analysed by applying abstraction techniques.

Given the practical limitations of exploring large state spaces, we instead prefer to concentrate on proof theoretic techniques. Some such proofs have also already been undertaken for the FireWire protocol. For instance, Romijn models the protocol using timed I/O automata in order to prove not only that the protocol can elect a leader correctly, but also that it will signal an error if the network contains a cycle and no leader can be elected [Rom01]. Earlier, a series of probabilistic timed I/O automata were used by Stoelinga and Vaandrager to prove the protocol's correctness [SV99]. Typically, these models represent the communications behaviour of the protocol in considerable detail. For instance, Romijn models all the internal and external actions of each node. The resulting proofs are therefore quite complex and many are reduced to proof 'sketches' in the published paper [Rom01].

By contrast, our goal in this paper is to devise a highly abstract model which defines nothing but the probabilistic properties of the protocol. Based on the results of the previous studies cited above, we take it for granted that the nodes behave and interact correctly, and that the overall protocol will eventually succeed, and instead concentrate on the probabilities of the overall algorithm terminating after a certain number of attempts in a worst case timing scenario.

3. Background: The *pGCL* Formalism

Dijkstra's *Guarded Command Language* (*GCL*) is a widely recognised notation for describing computations [Dij76]. It provides familiar programming language-like statements for declaring constants and variables (**con** and **var**), assignment (:=), sequencing (;), choice (**if** ... **fi**) and iteration (**do** ... **od**). To introduce syntactic abbreviations, we also allow '**let**' declarations below, with the obvious meaning.

Semantically, a *GCL* statement S is defined by its *weakest precondition*, $\mathrm{wp}\,.S.R$, a predicate characterising those states from which statement S will terminate in a state satisfying postcondition R. For instance, consider the assignment statement '$v := E$' for some variable v and expression E. Given an arbitrary postcondition predicate R, the semantics of assignment is defined as $\mathrm{wp}\,.(v := E).R \equiv R[E/v]$, where $R[E/v]$ is expression R with all free occurrences of 'v' replaced by 'E', and '$\equiv$' denotes predicate equivalence [Mor90]. As a concrete example, consider the particular assignment that increments variable x, and the postcondition that x is positive. Then $\mathrm{wp}\,.(x := x+1).(x > 0) \equiv x \geqslant 0$. Thus, if x is initially 2 this weakest precondition is true and the assignment will always terminate in a state where postcondition $x > 0$ holds. However, if x is initially -1, the weakest precondition expression $x \geqslant 0$ is false, so the assignment cannot make the postcondition true.

The basic *GCL* language can be extended with constructs not normally found in programming languages. An *assumption* statement '$\{P\}$' expresses the belief that predicate P is true at this point in the program [Mor90]. This information can be used for reasoning, but the statement does not compile to executable code. A demonic choice '$S \sqcap T$' may choose nondeterministically to either behave like statement S

or statement T; we cannot predict which [BvW98]. We also allow assignment statements to be generalised to multiple assignments, '$v, w := E, F$', in which the expressions on the right hand side are evaluated simultaneously.

The *probabilistic Guarded Command Language* (*pGCL*) [MM99] then extends this language even further by treating predicates not as Boolean-valued, but as probabilities in the closed interval $[0, 1]$. Rather than weakest preconditions, the semantics is defined in terms of *weakest pre-expectations*, i.e., the probability that a statement will achieve a desired postcondition. The formalism also introduces a probabilistic choice operator '$S\,{}_p\!\oplus T$' to the programming notation. This construct chooses to behave like statement S with probability p and like statement T with probability $(1 - p)$. Some weakest pre-expectation definitions are shown below [MM99]. They have a strong similarity to standard weakest precondition definitions. Let expression R be a probability, rather than a predicate.

Definition D1 $\mathrm{wp}.(v := E).R \equiv R[E/v]$
Definition D2 $\mathrm{wp}.(S\ ;\ T).R \equiv \mathrm{wp}.S.(\mathrm{wp}.T.R)$
Definition D3 $\mathrm{wp}.(S \sqcap T).R \equiv (\mathrm{wp}.S.R) \min (\mathrm{wp}.T.R)$
Definition D4 $\mathrm{wp}.(S\ {}_p\!\oplus\ T).R \equiv p * (\mathrm{wp}.S.R) + (1 - p) * (\mathrm{wp}.T.R)$

There is also a pleasing relationship between predicates and probabilities. For a particular predicate P, *pGCL* defines expression '$[P]$' to equal 1 if the predicate is true and 0 otherwise. This then allows logical operators in predicates to be related to arithmetic operators on probabilities. For instance, predicate implication '$\Rrightarrow$' [Mor90] is interpreted as the '$\leqslant$' relation on the corresponding probabilities [MM99]. A number of helpful laws are also available for manipulating logical and arithmetic operators, including the following [SMM97]. Let Q and R be predicates.

Law L1 $Q \Rightarrow R$ iff $[Q] \Rrightarrow [R]$
Law L2 $[Q \wedge R] \equiv [Q] * [R]$
Law L3 $[Q \wedge R] \equiv [Q] \min [R]$
Law L4 $[Q \vee R] \equiv [Q] \max [R]$

The two different arithmetic interpretations of logical conjunction offered by Laws L2 and L3 seem contradictory, but can be reconciled by recalling that probabilities $[Q]$ and $[R]$ can have values 0 or 1 only.

For example, the *pGCL* statement $(x := x+1)_{0.8}\!\oplus(x := x-1)$ describes a system that with probability 0.8 increments x, but otherwise decrements x. Let probability $[x > 0]$ express the likelihood that x is positive in the current state. Then the weakest pre-expectation that this *pGCL* statement will achieve such a state can be calculated using Definitions D4 and D1 above as follows.

$$
\begin{aligned}
&\mathrm{wp}.((x := x + 1)\ {}_{0.8}\!\oplus\ (x := x - 1)).[x > 0] \\
\equiv\ &0.8 * (\mathrm{wp}.(x := x + 1).[x > 0]) + 0.2 * (\mathrm{wp}.(x := x - 1).[x > 0]) \\
\equiv\ &0.8 * [x \geqslant 0] + 0.2 * [x > 1]
\end{aligned}
$$

To interpret this outcome, consider some possible initial states. If x equals 2 then the weakest pre-expectation that the statement will make x positive is $0.8 * 1 + 0.2 * 1 = 1$. The result is assured, no matter which way the probabilistic choice goes. If x is initially -1, then the pre-expectation is $0.8 * 0 + 0.2 * 0 = 0$. There is no possibility that the statement can make x positive. However, if x is initially 1, then the pre-expectation is $0.8 * 1 + 0.2 * 0 = 0.8$. In other words, there is a 0.8 probability that the statement will terminate with x positive. This ability to model probabilistic outcomes is the defining characteristic of *pGCL*.

4. Protocol Specification and Proof

The *leader election protocol* of IEEE 1394 [Ins95, Ins00], also known as the *tree identify protocol* [Ins95, pp. 100–102], may occur in two phases. In the initial phase, each node negotiates leadership by interacting with its neighbours until either a leader is elected, or two nodes remain in contention for leadership. Each node waits to receive messages relinquishing leadership from all but one of its neighbours, and then sends such a message to the remaining neighbour, thereby building a spanning tree of the network. (Each node behaves deterministically in the initial phase, although the overall algorithm is nondeterministic due to race conditions between the nodes.) In the case that two nodes remain in contention for leadership, a second

$$\textbf{con } N : \mathbb{N}_1 \bullet \tag{1}$$
$$\textbf{let } \mathit{Index} = 1 \mathinner{..} N \bullet \tag{2}$$
$$\textbf{var } \mathit{nodes} : \mathit{Index} \to \mathbb{B} \bullet \tag{3}$$
$$\textbf{let } \mathit{contenders} = \{x \mid (x \mapsto \text{true}) \in \mathit{nodes}\} \bullet \tag{4}$$
$$\{\forall\, i : \mathit{Index} \bullet \mathit{nodes}(i) = \text{true}\}\,; \tag{5}$$
$$\textbf{do } \#\mathit{contenders} > 2 \to \tag{6}$$
$$\quad \sqcap_{n \in \{c \mid c \subset \mathit{contenders} \wedge c \neq \{\}\}} (\mathit{nodes} := \mathit{nodes} \uplus \{(i \mapsto \text{false}) \mid i \in n\}) \tag{7}$$
$$\textbf{od} \tag{8}$$

Fig. 1. Abstract model of the IEEE 1394 root contention protocol's initial phase in *pGCL*.

probabilistic phase then resolves contention between these two nodes by using an algorithm based on a random choice.

In this section we model and verify properties of both of these phases, at a high level of abstraction, using the *pGCL* formalism. For the initial phase we prove that it is guaranteed to terminate in a state with either one or two contenders for leadership remaining. The reasoning here is standard, and does not involve probabilities, but serves to introduce the *pGCL* notation. We then present a formal model of the probabilistic phase and use it to determine the likelihood of the contention resolution algorithm terminating within a certain number of attempts, based on the given probability of success at each attempt.

In practice, contention resolution depends on a probabilistic choice and on timing. When contention is detected each node probabilistically chooses to wait a 'short' or 'long' time before trying again. Timing issues are significant in contention resolution. Simons and Stoelinga [SS01] derive inequalities relating *delay* (time to transmit messages), *rc_fast_min* (minimum 'short' wait time in contention resolution), *rc_fast_max* (maximum 'short' wait time in contention resolution) and *rc_slow_min* (minimum 'long' wait time in contention resolution). If these inequalities are not satisfied the protocol will not operate correctly. Furthermore, small differences in the timing parameters in each node can increase the likelihood that contention is resolved.

For example, consider two nodes, A and B, choosing to wait a short time. If A has a smaller setting for 'short' than B then there is a chance that A's message relinquishing leadership will arrive before B's wait time has expired, and thus contention is resolved and B becomes leader. Conversely, if A and B both have the same setting for 'short' then contention is not resolved and another attempt must be made. This is representative of a particular worst-case scenario for this protocol, in which the effects of probabilistic choice (whether to wait 'short' or 'long') determine contention resolution. So, to focus on just the effects of probabilistic choice, we make the strong assumption that all nodes use exactly the same values for the time periods 'short' and 'long'. We also assume that *computation speed* and *time to transmit messages* are equivalent in all nodes, again to eliminate the effects of time on contention resolution. Finally, we also ignore the protocol's 'force root' parameter which affects leader election through timing.

4.1. Initial Phase Model

Figure 1 presents a high-level view of the protocol's initial phase. It describes properties of the network as a whole rather than describing the parallel behaviour of individual nodes.

Let $\mathbb{N}_1$ be the set of positive integers. The model assumes there are N nodes in the network (line 1), indexed from 1 to N (line 2). Let $\mathbb{B}$ be the Boolean type. Each node's status as a contender is represented by a Boolean which is true only if it has the potential to become leader. The overall system state is a total function, *nodes*, from node identifiers to contender status (line 3). An array of Booleans can be more succinctly represented as a set. In fact, the abbreviation '*contenders*' is exactly that (the set of all node identifiers with contender status 'true' (line 4)). Both representations are convenient in Section 4.3. Let functions be represented by sets of mappings $(d \mapsto r)$ from domain values d to range values r in the usual way [Spi92].

Operationally, the protocol is described as a simple iterative behaviour. Initially, it is assumed that every node i has the potential to be elected leader, so each one has 'contender' status (line 5). Let $\#S$ be the number of elements in set S. The protocol then iterates as long as the number of contenders exceeds two (line 6). Thus, the initial phase of the protocol may terminate with either one or two contenders remaining.

To define the action within the loop, let '$\mathrm{dom}\, f$' be the domain of a function f, as usual [Spi92].

$$\mathrm{dom}\, f \stackrel{\mathrm{def}}{=} \{x \mid (\exists\, y \bullet (x \mapsto y) \in f)\}$$

Also, let $f \uplus g$ denote overriding of mappings in function f with mappings with the same domain value from function g [Spi92].

$$f \uplus g \stackrel{\mathrm{def}}{=} \{(x \mapsto y) \mid (x \mapsto y) \in f \wedge x \notin \mathrm{dom}\, g\} \cup g$$

At each iteration, we require that one or more nodes drop out of contention (line 7). This is represented by a (generalised) nondeterministic choice over a non-empty, strict subset n of the remaining contenders. Thus one or more nodes, but not *all* the remaining nodes, will be eliminated at each iteration. The nondeterminism in the choice of subset models the unpredictable effect of race conditions on the actual protocol. Having chosen to eliminate some node i, the assignment statement then changes its contender status in the *nodes* function to 'false'.

Although intimidating, the statement on line 7 offers a reasonable abstraction of the actual protocol's behaviour. In practice, the protocol makes progress when a node yields the right to become leader to its neighbour. Our model allows several nodes to do this in one step, to represent concurrency in the network. A simpler model, such as the following, would not be sufficient.

$$\sqcap_{i \in contenders}(nodes := nodes \uplus \{(i \mapsto \mathrm{false})\})$$

This statement would always decrease the number of contenders by one at each iteration, and unrealistically cause the loop to always exit with exactly two contenders. Our model allows the loop to exit with either one or two contenders, i.e., sometimes timing in the network causes a leader to be elected with no contention, while in other cases two nodes contend to be leader.

We do *not* model here the connectivity of a particular network. Therefore we do not model *why* a node changes state. Thus, the model captures the set of moves for all possible spanning trees, over all possible topologies for networks of this size.

4.2. Initial Phase Proof

Our goal now is to prove that the model in Figure 1 always terminates in a particular state, satisfying the property that only one or two contenders remain.

$$\#contenders \in \{1, 2\}$$

Although the individual nodes behave deterministically, the network-wide result depends on the communications topology and propagation delays, and is nondeterministic. This proof could be done using conventional reasoning, but we use the *pGCL* proof rules [MM99] here, in order to introduce the formalism before the more complicated probabilistic proof in Section 4.4.

The **Probabilistic Loops Rule** [MM99] can be used to determine a lower bound on the probability that a loop of the form '**do** $G \rightarrow B$ **od**', with guard G and body B, will terminate in a certain state. Let T be the probability that the loop will terminate. In our examples this is always 1, but in other cases loop termination may not be guaranteed [Mor96]. Let I be a loop invariant *probability* (rather than the usual predicate). The rule is then as follows [MM99].

$$\begin{array}{rl} \text{If} & [G] * I \Rrightarrow \mathrm{wp}\,.B.I \\ \text{and} & I \Rrightarrow T \\ \text{then} & I \Rrightarrow \mathrm{wp}\,.(\mathbf{do}\ G \rightarrow B\ \mathbf{od}).([\neg\, G] * I)\,. \end{array}$$

Firstly we must show that the probability of the guard G and the invariant I both being true does not exceed the weakest pre-expectation of the loop body B with respect to the invariant. Next we must show that the invariant may never exceed the termination probability T. We may then finally conclude that the weakest pre-expectation of the whole loop, with respect to the probability of the guard being false and the invariant true, is at least as great as the probability of the loop invariant.

An obvious loop invariant in our case is as follows.

$$J \stackrel{\text{def}}{=} [\#contenders > 0]$$

Furthermore, it is clear that the negation of the guard, and this invariant, equals the goal that we want to achieve, using the algebraic properties of probabilistic predicates [SMM97].

$$\begin{aligned}
& [\neg\,(\#contenders > 2)] * [\#contenders > 0] \\
\equiv\ & [\#contenders \leqslant 2] * [\#contenders > 0] \\
\equiv\ & \text{`by Law L2'} \\
& [\#contenders \leqslant 2 \wedge \#contenders > 0] \\
\equiv\ & [\#contenders \in \{1, 2\}]
\end{aligned}$$

Now we apply the loop rule itself. Looking at the second condition first, we consider the probability that the loop in Figure 1 terminates. For this we appeal to *pGCL*'s **Probabilistic Variant Rule** [MM99]. Normally a loop variant is a natural number-valued function on the program state whose value is decreased by each iteration of a loop—this ensures that the loop must terminate because the number cannot be decreased indefinitely. In the probabilistic version, the aim is to find a loop variant which will be decreased by the loop body with some non-zero probability. In our example there is an obvious loop variant expression, '$\#contenders$'. Indeed, the whole purpose of the loop is to decrease this value at each iteration. By its definition in Section 4.1, this expression is bounded above by N, the number of nodes. It is bounded below by 1 because the only statement that can decrease it is statement 7, and the constraint on the nondeterministic choice that $c \subset contenders$ ensures that there is always at least one contender left each time the assignment is performed. Furthermore, statement 7 is guaranteed to decrease the variant at each iteration, thanks to the constraint that $c \neq \{\}$. Therefore, the 'termination probability' [MM99] of the loop is 1. The second condition of the loop rule is therefore satisfied immediately because $I \Rrightarrow 1$ for any invariant probability I, so in particular $J \Rrightarrow 1$.

The first loop rule condition then requires us to show that the weakest pre-expectation of the loop body on line 7, with respect to invariant J, is no less than the loop guard on line 6 times J. The proof proceeds as follows.

$$\begin{aligned}
& \text{wp}\,.(\text{line } 7).J \\
\equiv\ & \text{wp}\,.(\sqcap_{n \in \{c \mid c \subset contenders \wedge c \neq \{\}\}}(nodes := nodes \uplus \{(i \mapsto \text{false}) \mid i \in n\})).(\#contenders > 0) \\
\equiv\ & \text{`by Definition D3, generalised from the binary case'} \\
& \min_{n \in \{c \mid c \subset contenders \wedge c \neq \{\}\}}(\text{wp}\,.(nodes := nodes \uplus \{(i \mapsto \text{false}) \mid i \in n\}).(\#contenders > 0)) \\
\equiv\ & \text{`by Definition D1 and by unfolding the definition of } contenders\text{'} \\
& \min_{n \in \{c \mid c \subset contenders \wedge c \neq \{\}\}}[\#\{x \mid (x \mapsto \text{true}) \in (nodes \uplus \{(i \mapsto \text{false}) \mid i \in n\})\} > 0] \\
\equiv\ & \min_{n \in \{c \mid c \subset contenders \wedge c \neq \{\}\}}[(\#\{x \mid (x \mapsto \text{true}) \in nodes\} - \#n) > 0] \\
\equiv\ & \text{`by folding the definition of } contenders\text{'} \\
& \min_{n \in \{c \mid c \subset contenders \wedge c \neq \{\}\}}[(\#contenders - \#n) > 0] \\
\equiv\ & \min_{0 < m < \#contenders}[(\#contenders - m) > 0] \\
\equiv\ & \text{`by Law L3'} \\
& [(\#contenders - 1) > 0 \wedge \cdots \wedge (\#contenders - (\#contenders - 1)) > 0] \\
\equiv\ & [\#contenders > 1] \\
\Lleftarrow\ & \text{`by Law L1'} \\
& [\#contenders > 2] \\
\equiv\ & [\#contenders > 2 \wedge \#contenders > 0] \\
\equiv\ & \text{`by Law L2'} \\
& [\#contenders > 2] * [\#contenders > 0]
\end{aligned}$$

This completes the proof. Most steps are obvious applications of the definitions and laws. The interesting

$$
\begin{array}{lr}
\textbf{con}\ i, j : \mathit{Index}\ \bullet & (9)\\
\textbf{var}\ \mathit{short\_i}, \mathit{short\_j} : \mathbb{B}\ \bullet & (10)\\
\textbf{con}\ M : \mathbb{N}_1\ \bullet & (11)\\
\textbf{var}\ k : 1 \mathinner{\ldotp\ldotp} M\ \bullet & (12)\\
\textbf{con}\ p : (0,1)\ \bullet & (13)\\
 & \\
\{\#\mathit{contenders} \in \{1,2\}\}\ ; & (14)\\
\{\mathit{contenders} \subseteq \{i,j\}\}\ ; & (15)\\
\{i \neq j\}\ ; & (16)\\
k := 0\ ; & (17)\\
\textbf{do}\ k < M \wedge \#\mathit{contenders} > 1 \rightarrow & (18)\\
\quad (\mathit{short\_i} := \mathrm{true})\ {}_p\!\oplus (\mathit{short\_i} := \mathrm{false})\ ; & (19)\\
\quad (\mathit{short\_j} := \mathrm{true})\ {}_p\!\oplus (\mathit{short\_j} := \mathrm{false})\ ; & (20)\\
\quad \mathit{nodes}, k := \mathit{nodes} \uplus \{(i, \neg \mathit{short\_i} \vee \mathit{short\_j}), (j, \neg \mathit{short\_j} \vee \mathit{short\_i})\}, k+1 & (21)\\
\textbf{od} & (22)
\end{array}
$$

Fig. 2. Abstract model of the IEEE 1394 root contention protocol's probabilistic phase in *pGCL*.

steps are transforming an expression in '$\sqcap$' into one in 'min' parameterised over subsets c (those nodes changing state in this step). However, the actual members of c are not important—what matters is the size of c, represented by m above.

From the Probabilistic Loops Rule, we may now conclude that the probability of the loop achieving our desired goal, i.e., that there remain one or two contenders, cannot be less than invariant probability J. Assumption 5 in Figure 1 tells us that J's probability when the loop is entered is 1. Thus we can be certain that the loop will successfully terminate and will achieve the desired goal from this starting state.

4.3. Probabilistic Phase Model

We now consider the second phase of the protocol. Figure 2 presents our abstract *pGCL* model of the protocol's probabilistic phase. It relies on declarations 1 to 4 from Figure 1. It begins by assuming the property proven in Section 4.2 that there are only one or two contenders remaining (line 14). We introduce constants i and j to identify the remaining contender, or contenders (line 9), and formally state additional assumptions that no nodes other than i and j remain in contention (line 15), and that i and j are distinct (line 16). If there is only one node in contention then either i or j will be a contender, but not both, and the model terminates trivially.

In the IEEE standard [Ins95, Ins00], contention between nodes i and j is resolved as follows. When contention is detected, node i chooses a random Boolean and uses it to decide whether to wait for either a 'short' or 'long' time. After waiting its chosen time, node i checks for a 'RX_PARENT_NOTIFY' message from node j, meaning that node j would like node i to be the leader. If no such message has arrived then node i resends its own 'TX_PARENT_NOTIFY' message, indicating that node i wants node j to be the leader. Node j behaves similarly.

Our model abstracts from the details of contention resolution, while maintaining its basic characteristics. It allows the two nodes to attempt to resolve contention until either a single leader is elected or a given time limit expires. The time limit is modelled by a maximum number M of attempts to elect a leader (line 11). Loop counter k (line 12) keeps track of the number of attempts and is initialised to 0 (line 17). Both k and M are auxiliary features of our model allowing us to reason about the number of attempts to resolve contention. In the IEEE standard there is no time limit on contention resolution.

An attempt to resolve contention is modelled by the decision to wait for a short or long time, and the subsequent updating of each node's status. Recall that we assume that 'short' means the same time value in each node, and similarly 'long'. We ignore the effects of any other timing issues, such as time to transmit messages. This assumption allows us to concentrate on the absolute worst case, in which timing details have no effect on contention resolution. The assumption further leads to the conclusion that if both nodes wait the same length of time they are guaranteed to re-enter the contention phase. The possible outcomes of an attempt to resolve contention are therefore either that

1. both nodes wait a short time and yield to each other,
2. one node waits a short time and yields to its opponent (which is waiting a long time), or
3. both nodes wait a long time and yield to each other.

Note that outcome 2 is twice as likely as either cases 1 or 3 since there are two ways to achieve it—either node i waits a short time and j waits a long time, or vice versa. From this description, readers versed in probability theory may recognise already that the probability of the algorithm successfully terminating in M steps is $1-(p^2+(1-p)^2)^M$. Below we show how this result can be determined by mechanical application of *pGCL* reasoning. Having gained experience of this technique on a small, well understood example, we can have more confidence in applying it to larger, more complex situations.

Variable *short_i* (line 10) indicates whether node i has decided to wait for a short period of time before offering leadership to the other node. Similarly for *short_j*. At each attempt to resolve contention, node i decides probabilistically whether to wait for a short time or a long time (line 19). Constant p (line 13), between 0 and 1 exclusive, represents the choice's probability. Similarly, node j makes the same choice with the same probability (line 20). A multiple assignment then models the actions of the two nodes updating their state in response to these decisions, and the attempt counter k being incremented (line 21). A given node i remains a contender for leadership either if it decides to wait for a long time ($\neg$ *short_i*) or its opponent waits for a short time (*short_j*). The loop guard (line 18) tells us that the protocol terminates when either the time limit expires ($k = M$), or the number of contenders no longer exceeds one, in which case a leader has been successfully elected.

4.4. Probabilistic Phase Proof

We now aim to use the model in Figure 2 and the Probabilistic Loops Rule to determine a lower bound on the probability that a leader will be elected, given that the number of attempts is bounded by constant M. Our desired post-expectation is the probability that exactly one contender remains.

$$R \stackrel{\text{def}}{=} [\#\mathit{contenders} = 1]$$

By appealing to Definition D2, our overall strategy will be to find a probabilistic invariant K, at least as likely as R, that is maintained by the loop on lines 18 to 22, and to then show that the assumptions and initialisation on lines 14 to 17 establish this invariant.

4.4.1. Loop Termination

The loop termination requirement is trivial. We can use the Probabilistic Variant Rule [MM99] to immediately show that the loop on lines 18 to 22 always terminates. Let expression '$M - k$' be the loop variant. This expression is bounded above by M and below by 0, thanks to the type of k (line 12), and it decreases with probability 1 at every iteration since k is guaranteed to be incremented by the assignment on line 21.

4.4.2. Probabilistic Loop Invariant

The key to the proof is identifying a satisfactory invariant. We define a *probabilistic invariant* [MM99] which represents the exact probability of the loop successfully terminating from an arbitrary state. Let q denote expression '$p^2 + (1-p)^2$'.

$$K \stackrel{\text{def}}{=} [\mathit{contenders} = \{i\} \vee \mathit{contenders} = \{j\}] \max ((1 - q^{M-k}) * [\mathit{contenders} = \{i, j\}])$$

This is the maximum of two expressions denoting a 'successful' state and a state from which success can be achieved, respectively.

The left-hand argument to 'max' in invariant K represents the successful state where the goal of having only one contender left has already been achieved. If this predicate is already true then the probability of success is immediately 1; we have elected a leader.

The right-hand argument to 'max' is the probability of taking a 'good' step, i.e., a step which can eventually result in a leader being elected, in a state where success has not yet been achieved. Its right-hand term states that nodes i and j are the only remaining contenders (and i and j are distinct due to assumption 16). If this was not true, i.e., if some other node was also a contender, then the loop in Figure 2 cannot make progress towards termination. The left-hand term is a probability representing the likelihood of making a good choice in one of the remaining $M - k$ iterations, expressed as 1 minus the probability of always making 'bad' choices. A bad decision, i.e., one that does not make progress, occurs either when both nodes wait a short time, with probability p^2, or when both nodes wait a long time, with probability $(1-p)^2$. Thus the probability of a bad step in an iteration is $p^2 + (1-p)^2$, called q for convenience. The probability of a bad step in all remaining iterations is then q^{M-k}, and therefore the probability of a good step at some point is $1 - q^{M-k}$.

4.4.3. Invariant Achieves Goal

We want to use the Probabilistic Loops Rule [MM99], described in Section 4.2, which finds a pre-expectation for the whole loop with respect to the negation of the loop guard and the invariant. Therefore, we first need to show that the negation of the loop guard (line 18) times invariant K 'implies' our goal R.

$$
\begin{array}{ll}
 & [\neg(\text{line } 18)] * K \\
\equiv & [\neg(k < M \wedge \#\mathit{contenders} > 1)] * \\
 & ([\mathit{contenders} = \{i\} \vee \mathit{contenders} = \{j\}] \max ((1 - q^{M-k}) * [\mathit{contenders} = \{i,j\}])) \\
\equiv & \text{`since } k \text{ is bounded by } M \text{ (line 12)'} \\
 & [k = M \vee \#\mathit{contenders} \leqslant 1] * \\
 & ([\mathit{contenders} = \{i\} \vee \mathit{contenders} = \{j\}] \max ((1 - q^{M-k}) * [\mathit{contenders} = \{i,j\}])) \\
\equiv & \text{`by Law L2 and since `}\wedge\text{' distributes through `max''} \\
 & [(k = M \vee \#\mathit{contenders} \leqslant 1) \wedge (\mathit{contenders} = \{i\} \vee \mathit{contenders} = \{j\})] \max \\
 & ((1 - q^{M-k}) * [(k = M \vee \#\mathit{contenders} \leqslant 1) \wedge \mathit{contenders} = \{i,j\}]) \\
\Rrightarrow & \text{`by Law L1'} \\
 & [\#\mathit{contenders} = 1] \max \\
 & ((1 - q^{M-k}) * [\mathit{contenders} = \{i,j\} \wedge (k = M \vee \#\mathit{contenders} \leqslant 1)]) \\
\equiv & \text{`by Law L4'} \\
 & [\#\mathit{contenders} = 1] \max \\
 & (1 - q^{M-k}) * [\mathit{contenders} = \{i,j\} \wedge k = M] \max \\
 & (1 - q^{M-k}) * [\mathit{contenders} = \{i,j\} \wedge \#\mathit{contenders} \leqslant 1] \\
\equiv & \text{`by assumption 16 and since [false] equals 0'} \\
 & [\#\mathit{contenders} = 1] \max ((1 - q^{M-k}) * [\mathit{contenders} = \{i,j\} \wedge k = M]) \\
\equiv & \text{`since } k = M \text{ makes } (1 - q^{M-k}) \text{ equal 0'} \\
 & [\#\mathit{contenders} = 1]
\end{array}
$$

Therefore, finding the probability that the loop terminates with the guard false and invariant true will give us a lower bound on the probability of achieving our goal R, thanks to the monotonicity of weakest pre-expections, i.e., that if $P \Rrightarrow Q$ then $\mathrm{wp}.S.P \Rrightarrow \mathrm{wp}.S.Q$, for any statement S and probabilities P and Q [MM99].

4.4.4. Loop Body

Next we show that the loop body preserves the invariant. This is done by proving that the weakest pre-expectation of the loop body (lines 19–21) is 'implied by' the loop guard (line 18) multiplied by invariant K.

As we work backwards through the assignment statements in the loop body, the values in function *nodes* corresponding to nodes i and j are frequently overridden. To accommodate this, it is convenient to define the following parameterised version of *contenders* in which particular Boolean values b and c are associated with nodes i and j.

$$cont(b, c) \stackrel{\text{def}}{=} \{x \mid (x \mapsto \text{true}) \in (nodes \uplus \{(i \mapsto b), (j \mapsto c)\})\}$$

We then define some specific instances of this abbreviation which will occur in the proof, derived from the expression on line 21 and substitutions into that expression corresponding to the choices in lines 19 and 20.

$$\begin{aligned}
contenders_1 &\stackrel{\text{def}}{=} cont((\neg\ short_i \vee short_j), (\neg\ short_j \vee short_i)) \\
contenders_2 &\stackrel{\text{def}}{=} cont(\text{true}, short_i) \\
contenders_3 &\stackrel{\text{def}}{=} cont(\neg\ short_i, \text{true}) \\
contenders_4 &\stackrel{\text{def}}{=} cont(\text{true}, \text{true}) \\
contenders_5 &\stackrel{\text{def}}{=} cont(\text{false}, \text{true}) \\
contenders_6 &\stackrel{\text{def}}{=} cont(\text{true}, \text{false})
\end{aligned}$$

The proof itself is lengthy but merely involves expanding the weakest pre-expectation definitions and then simplifying the resulting expression.

$$\begin{aligned}
& \text{wp}.(\text{lines 19–21}).K \\
\equiv\ & \text{‘by Definition D2’} \\
& \text{wp}.(\text{line 19}).(\text{wp}.(\text{line 20}).(\text{wp}.(\text{line 21}).K)) \\
\equiv\ & \text{‘by Definition D1’} \\
& \text{wp}.(\text{line 19}).(\text{wp}.(\text{line 20}).([contenders_1 = \{i\} \vee contenders_1 = \{j\}] \max \\
& \qquad ((1 - q^{M-(k+1)}) * [contenders_1 = \{i, j\}]))) \\
\equiv\ & \text{‘by Definitions D4 and D1’} \\
& \text{wp}.(\text{line 19}).(p * ([contenders_2 = \{i\} \vee contenders_2 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_2 = \{i, j\}])) + \\
& \quad (1 - p) * ([contenders_3 = \{i\} \vee contenders_3 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_3 = \{i, j\}]))) \\
\equiv\ & \text{‘by Definitions D4 and D1’} \\
& p * (p * ([contenders_4 = \{i\} \vee contenders_4 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_4 = \{i, j\}])) + \\
& \quad (1 - p) * ([contenders_5 = \{i\} \vee contenders_5 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_5 = \{i, j\}]))) + \\
& (1 - p) * (p * ([contenders_6 = \{i\} \vee contenders_6 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_6 = \{i, j\}])) + \\
& \quad (1 - p) * ([contenders_4 = \{i\} \vee contenders_4 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_4 = \{i, j\}]))) \\
\equiv\ & \text{‘by assumption 16’} \\
& p * (p * ([\text{false}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_4 = \{i, j\}])) + \\
& \quad (1 - p) * ([\text{false} \vee contenders_5 = \{j\}] \max \\
& \qquad ((1 - q^{M-k-1}) * [\text{false}]))) + \\
& (1 - p) * (p * ([contenders_6 = \{i\} \vee \text{false}] \max \\
& \qquad ((1 - q^{M-k-1}) * [\text{false}])) + \\
& \quad (1 - p) * ([\text{false}] \max \\
& \qquad ((1 - q^{M-k-1}) * [contenders_4 = \{i, j\}])))
\end{aligned}$$

$$
\begin{array}{ll}
\equiv & \text{`since [false] equals 0'} \\
& p * (p * ((1 - q^{M-k-1}) * [contenders_4 = \{i,j\}]) + \\
& \quad (1-p) * [contenders_5 = \{j\}]) + \\
& (1-p) * (p * [contenders_6 = \{i\}] + \\
& \qquad (1-p) * ((1 - q^{M-k-1}) * [contenders_4 = \{i,j\}])) \\
\equiv & p^2 * (1 - q^{M-k-1}) * [contenders_4 = \{i,j\}] + \\
& p * (1-p) * [contenders_5 = \{j\}] + \\
& p * (1-p) * [contenders_6 = \{i\}] + \\
& (1-p)^2 * (1 - q^{M-k-1}) * [contenders_4 = \{i,j\}] \\
\Lleftarrow & \text{`by the definition of overriding `}\uplus\text{' and Law L1'} \\
& p^2 * (1 - q^{M-k-1}) * [contenders = \{i,j\}] + \\
& p * (1-p) * [contenders = \{i,j\}] + \\
& p * (1-p) * [contenders = \{i,j\}] + \\
& (1-p)^2 * (1 - q^{M-k-1}) * [contenders = \{i,j\}] \\
\equiv & (p^2 * (1 - q^{M-k-1}) + 2 * p * (1-p) + (1-p)^2 * (1 - q^{M-k-1})) * [contenders = \{i,j\}] \\
\equiv & \text{`by the definition of } q\text{'} \\
& (1 - q^{M-k}) * [contenders = \{i,j\}] \\
\Lleftarrow & \text{`by Law L1'} \\
& (1 - q^{M-k}) * [contenders = \{i,j\} \wedge k < M \wedge \#contenders > 1] \\
\equiv & \text{`since [false] equals 0'} \\
& [\text{false}] \max ((1 - q^{M-k}) * [contenders = \{i,j\} \wedge k < M \wedge \#contenders > 1]) \\
\equiv & [(contenders = \{i\} \vee contenders = \{j\}) \wedge k < M \wedge \#contenders > 1] \max \\
& ((1 - q^{M-k}) * [contenders = \{i,j\} \wedge k < M \wedge \#contenders > 1]) \\
\equiv & \text{`by Law L2 and since `}*\text{' distributes through `max''} \\
& [k < M \wedge \#contenders > 1] * \\
& ([contenders = \{i\} \vee contenders = \{j\}] \max ((1 - q^{M-k}) * [contenders = \{i,j\}]))
\end{array}
$$

Therefore we achieve the loop guard times the invariant, as desired. This result, the fact that loop termination will achieve the desired goal as shown in Section 4.4.3, and the termination argument of Section 4.4.1, allow us to use the Probabilistic Loops Rule [MM99] to conclude that the probability of the loop in Figure 2 achieving postcondition R is no less than the probabilistic invariant K.

4.4.5. Loop Initialisation

To complete the proof, we must find a lower bound on the probability of the loop initialisation code establishing invariant K. Here loop initialisation just consists of the assignment on line 17, in the context of the assumptions on lines 14 to 16.

$$
\begin{array}{ll}
& \text{wp.(line 17)}.K \\
\equiv & \text{`by Definition D1'} \\
& [contenders = \{i\} \vee contenders = \{j\}] \max ((1 - q^M) * [contenders = \{i,j\}]) \\
\Lleftarrow & \text{`arithmetic'} \\
& ((1 - q^M) * [contenders = \{i\} \vee contenders = \{j\}]) \max ((1 - q^M) * [contenders = \{i,j\}]) \\
\equiv & \text{`by Law L4'} \\
& (1 - q^M) * [contenders \in \{\{i\}, \{j\}, \{i,j\}\}] \\
\equiv & \text{`by assumptions 14 and 15, and since [true] equals 1'} \\
& 1 - q^M
\end{array}
$$

Thus, the probability of the entire algorithm successfully electing a leader in M attempts is *at least* $1 - q^M$, that is, $1 - (p^2 + (1-p)^2)^M$. A similar proof can be performed to show that the probability of the entire

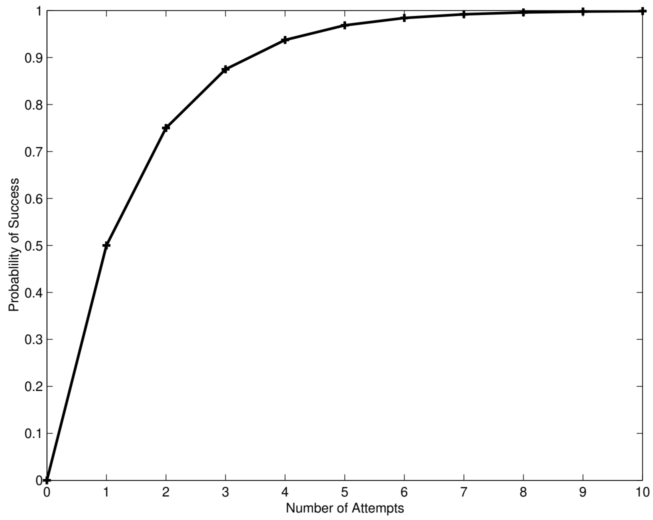

Fig. 3. Probability of solving root contention within a fixed number of attempts.

Table 1. Worst-case probabilities of electing a leader within some specific deadlines.

Attempts (M)	**Deadline** ($M * 1.67\,\mu$s)	**Probability** ($1 - q^M$)
1	1.67μs	0.5
2	3.34μs	0.75
5	8.35μs	0.96875
10	16.7μs	0.99902724

algorithm *failing* to elect a leader (that is, terminating with $contenders = \{i, j\}$ and $k = M$) is at least q^M. Therefore we know that the probability of success in our worst-case model is *exactly* $1 - (p^2 + (1-p)^2)^M$.

4.4.6. Relationship to Timing

The analysis above produced a probability of the protocol's success, with respect to the number of attempts. Recall that here we are concerned with the *worst case* duration, arising when the timing values are exactly the same in all nodes. We can also use this result to predict the worst case likelihood of successful termination of the contention resolution algorithm within a certain period of time by relating the number of attempts to the worst case time required for each attempt.

Figure 3 shows the calculated probabilities of success against the number of attempts, illustrating the expected result that as M tends to infinity, so the probability of success tends to 1. This implies the required safety and liveness properties, that exactly one leader is elected, and a leader is eventually elected. We use 0.5 for probability p since contention is resolved by choosing a random Boolean, and we assume the choice is unbiased.

Table 1 shows similar results against duration. The duration of each attempt to resolve contention is either 'short' or 'long'. From the amendment to the IEEE 1394 standard [Ins00], the wait times for nodes in contention are in the ranges 0.76μs–0.85μs ('short', or 'ROOT_CONTEND_FAST') and 1.59μs–1.67μs ('long', or 'ROOT_CONTEND_SLOW'). Processing and transmission times are negligible compared to these times and are therefore ignored, even for the earlier deterministic phase of the protocol. Therefore, we take the duration for each attempt to resolve contention to be 1.67μs. In establishing the upper bound we assume that all attempts take this time (i.e., choosing 'long' each time). This represents the most pessimistic situation imaginable because, for a given interval of time, it allows the smallest number of attempts to resolve contention, and hence the lowest probability of success.

The question of resolving contention within a given deadline was explored by Kwiatkowska et al [KNS02].

They calculated the probability of successful resolution within a time limit in the more general case where each contention round may take either a short or a long time. Our results are broadly in line with theirs, although direct comparison is difficult because they allow the more optimistic possibility that contention rounds can be short as well as long, which increases the probability of succeeding within a given deadline. For example, if the deadline is 3340ns, then contention may be resolved by two long rounds, or two short rounds and a long round. Hence, our probability of success for a given time is lower than theirs. This is to be expected since we are analysing worst case behaviour and therefore being as pessimistic as possible.

5. Evaluation

Comparative studies, in which different formal methods are applied to the same specification problem, have an important role in the wider dissemination and understanding of formal methods. Our primary contribution here is a formal analysis of the root contention algorithm, complementing the existing body of case studies based on the IEEE 1394 tree identify protocol [BSdRT01, D'A99, KNS02, Rom01, SS01, SV99]. This section attempts to rate our experience of using formal methods for this particular study to facilitate easier comparison with the other studies of this volume [FACJ].

The level of abstraction in our model is high; the structure of the network is ignored, nodes are modelled as Boolean values, and they have no detail of internal processing. Instead we view the system as a whole and simply record the changing states from one iteration of the protocol to the next. The focus of this article is the root contention resolution algorithm. Accordingly this was described in more detail. The *pGCL* formalism captures this abstract model clearly and succinctly.

The specification was analysed to establish a relation between the number of attempts to resolve contention and the probability of leader election. For completeness, we also analysed the initial phase of the protocol. All analysis was completely formal and carried out by hand. Corollaries of our result are the safety and liveness requirements as stated in the introductory paper [MRS02].

Of course, the value of any formal proof depends on how well the formal model matches reality. We assumed above, for instance, that during contention resolution the two nodes make their choices with the same probability p. If, however, this were not true then the proof could be redone using different values; only the arithmetic simplifications would differ.

We also ignored the influence of *absolute* timing on the protocol. In fact, if precise timing is considered, then typically success will be achieved in fewer attempts. This is because in our model if two contenders decide to wait the "same" length of time they will both remain in contention. In a practical implementation, however, the nodes' timing parameters or computation speeds may differ, allowing the possibility that, even in this case, contention will be resolved. Our approach abstracted the probabilistic behaviour of the protocol from timing concerns, which allowed us to reason in terms of the number of attempts, rather than the passage of time. As a result, the timing relationship discussed in Section 4.4.6 represents a worst-case situation. A practical implementation of the protocol may resolve contention more quickly but, by being conservative, our outcomes are always safe to rely on.

Our assumptions are well documented in the text of the paper, although not in the specification itself. No assumptions are made about the topology of the network or about the way nodes function.

We cannot speculate on whether the specification is easy to change, without knowing the nature of the change; however, since the specification is abstract, low level changes, such as changing the probability of choosing short over long delays, would have no effect. Changes to the main descriptions will invalidate the proofs, but if the changes are small then we anticipate that much of the proof could be reused easily.

Before starting this work neither author was familiar with *pGCL*. It took 3 months to learn *pGCL* and work out several (less abstract) versions of the specification before the final one was developed and the results proved. Given this experience we believe that anyone with a computing background could readily understand our solution, without detailed knowledge of *pGCL*.

Finally, we believe that such abstract specifications, while a useful analysis aid, cannot replace the IEEE standard. Standards have to be rather more precise and prescriptive about the operation of protocols used. We believe our more abstract approach allows more efficient reasoning about the protocol. Certainly it was easy to prove overall properties of the protocol in this case, without the need to consider the internal behaviour of nodes at all.

6. Conclusion

We have presented a probabilistic Guarded Command Language model of a leader election protocol and used it to analyse probabilistic properties of the protocol. Since the *pGCL* language builds on traditional semantics and reasoning techniques, we were able to conveniently perform both conventional reasoning about iterative behaviour and probabilistic reasoning about random choices in the same formalism. Furthermore, we were able to achieve useful results quickly and easily by taking a highly abstract view of the protocol.

Acknowledgements

We wish to thank Carroll Morgan for advice on devising probabilistic loop invariants, and Antonio Cerone for reviewing a draft of this paper. We also wish to thank the anonymous reviewers for their many helpful comments that improved the clarity of this article. This work was funded in part by a University of Queensland ECR Grant, *Development of Real-Time Distributed Systems*, and by grants from the University of Stirling Faculty of Management Research Fund, and the Carnegie Trust for the Universities of Scotland.

References

[BSdRT01] G. Bandini, R. Spelberg, R. de Rooij, and W. Toetenel. Application of parametric model checking—the root contention protocol. In *34th Annual Hawaii International Conference on System Sciences (HICSS-34)*. IEEE Computer Society, January 2001.

[BvW98] R.-J. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer-Verlag, 1998.

[D'A99] P. R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.

[Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[Ins95] Institute of Electrical and Electronics Engineers. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995*, August 1995.

[Ins00] Institute of Electrical and Electronics Engineers. *IEEE Standard for a High Performance Serial Bus (Amendment). Std 1394a-2000.*, June 2000.

[KNS02] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 13(6):pages, 2002.

[MRS02] S. Maharaj, J. Romijn and C. Shankland. IEEE 1394 Tree Identify Protocol: Introduction to the Case Study. *Formal Aspects of Computing*, 13(6):pages, 2002.

[MM99] C. Morgan and A. McIver. pGCL: Formal reasoning for random algorithms. *South African Computer Journal*, (22):14–27, March 1999. Special issue on the 1998 Winter School on Formal and Applied Computer Science.

[Mor90] C. Morgan. *Programming from Specifications*. Prentice-Hall, 1990.

[Mor96] C. Morgan. Proof rules for probabilistic loops. In He Jifeng, John Cooke, and Peter Wallis, editors, *BCS-FACS Seventh Refinement Workshop*, Electronic Workshops in Computing. Springer-Verlag, 1996. `http://www.ewic.org.uk/ewic/`.

[Rom01] J. Romijn. A timed verification of the IEEE 1394 leader election protocol. *Formal Methods in System Design*, 19(2):165–194, September 2001.

[SMM97] K. Seidel, C. Morgan, and A. McIver. Probabilistic imperative programming: A rigorous approach. In L. Groves and S. Reeves, editors, *Formal Methods Pacific '97*. Springer-Verlag, 1997.

[Spi92] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice-Hall, second edition, 1992.

[SS01] D. Simons and M. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. *International Journal on Software Tools for Technology Transfer*, 3(4):469–485, 2001.

[SV99] M. I. A. Stoelinga and F. W. Vaandrager. Root contention in IEEE 1394. In *5th AMAST Workshop on Real-Time and Probabilistic Systems*, volume 1601 of *Lecture Notes in Computer Science*, pages 53–74. Springer-Verlag, 1999.

[Sto02] M. Stoelinga. Fun with FireWire: Experiences with Verifying the IEEE 1394 Root Contention Protocol. *Formal Aspects of Computing*, 13(6):pages, 2002.

[FACJ] J. Cooke, S. Maharaj, J. Romijn and C. Shankland (eds). Special Issue on the IEEE 1394 Tree Identify Phase. *Formal Aspects of Computing*, 13(6), 2002.