The Tree Identify Protocol of IEEE 1394

Carron Shankland
Department of Computing Science and Mathematics,
University of Stirling*

Abstract

We specify the tree identify protocol of a high performance serial multimedia bus (IEEE standard 1394 [IEE95]) in three different levels of detail using μCRL [GP95]. We propose using the cones and foci verification technique of Groote and Springintveld [GS95] to show the descriptions equivalent under branching bisimulation. The proof of the equivalence of the two more abstract specifications is shown in detail and the proof of the equivalence of the most abstract and the more complex description, which is work in progress, is sketched.

1 Introduction

Much time and effort is expended in the development of new techniques for description and analysis of (computer) systems; however, many of these techniques remain the preserve only of their inventors, and are never widely used. This is often due to the sharp learning curve required to adopt them; many verification techniques have complex theoretical underpinnings, and require sophisticated mathematical skills to apply them. The study presented in this paper applies one such technique, the cones and foci technique of Groote and Springintveld [GS95], to a fragment of the software for a high performance serial multimedia bus, the IEEE standard 1394 [IEE95], also known as "Firewire".

Briefly, IEEE 1394 connects together a collection of systems and devices in order to carry all forms of digitized video and audio quickly, reliably, and inexpensively. Its architecture is scalable, and it is "hot-pluggable", so a designer or user can add or remove systems and peripherals easily at any time. The only requirement is that the form of the network should be a tree (other configurations leads to errors). Much effort has been expended on the description and verification of various parts of the standard, using several different formalisms and proof techniques.

*The bulk of this work was carried out while visiting the Programming Research Group, University of Amsterdam, under the EC HCM Fellowship scheme on the EXPRESS project.

The main operation of the standard, concerned with sending packets of information accross the network, is described using μCRL in [Lut97] and using E-LOTOS in [SM97]. The former is essentially a description only, with five correctness properties stated informally, but not formalised or proved. The exercise of [SM97] is based on the μCRL description, adding another layer of the protocol and carrying out the verification suggested, using the tool CADP [FGK+96].

In this paper we concentrate on the tree idenification protocol which occurs after a bus reset in the system, e.g. when a node is added to or removed from the network. The purpose of the tree identification protocol is to assign a (new) root, or leader, to the network. Essentially, the protocol consists of a set of negotiations between nodes to establish the direction of the parent-child relationship. Potentially, a node can be a parent to many nodes, but a child of at most one node. A node with no parent (after the negotiations are complete) is the leader. The tree identify protocol must ensure that a leader is chosen, and that it is the only leader chosen.

This part of the 1394 is also described elsewhere in the literature. The most abstract description is that using I/O automata by Vaandrager [Vaa97], which specifies the interactions of the tree identify protocol on a network level (so individual nodes are not specified, only the behaviour of the network as a whole). Verification is by (manual) manipulation of a number of invariants, phrased in predicate calculus. In contrast, a very fine grained, low level view is taken in [GRV97]. The system is specified in PVS, with real time introduced, and the overall behaviour of the network is given by specifying the behaviour of each node individually. A number of properties of the network are demonstrated.

The descriptions presented here fit inbetween these two levels, and analyses are carried out by hand.

There are three descriptions of the protocol, written using μCRL [GP95], in this paper:

- specification - merely announces a leader has been chosen.
- abstract description - nodes are specified individually and negotiate with their neighbours to determine the parent-child relationship. Communication is by hand shaking.
- complex description - negotiation as above, but communication between nodes occurs via two unidirectional channels (therefore messages may pass each other, causing conflicts in assigning the leader).

These descriptions may be found in Sections 2.1, 2.2 and 2.3 respectively.

We prove, using the cones and foci technique, that the abstract description has the same behaviour with respect to branching bisimulation as the simple specification. The proof may be found in Section 3.3. Currently in progress is the proof, using the same technique, of the equivalence of the complex description and the specification, therefore only the outline is sketched in Section 3.4.

2 Description of the Root Finding Protocol

The descriptions are given in μCRL. Familiarity is assumed with this formalism, or similar process algebras; an introduction may be found in [GP95]. Briefly, the main features of the formalism are: δ is the null process, $a.P$ indicates action sequencing and $P+Q$ indicates choice. Choice over a number of indexed processes, for some index set I, is written $\sum_{i:I} P_i$. Novel in μCRL is the "if then else" construct, written $P \lhd b \rhd Q$, meaning if b holds behave as P, otherwise behave as Q. Finally, the operator τ_I hides all those actions in the set I, by converting them to silent τ actions, and ∂_H restricts those actions in H, forcing the enclosed processes to communicate with each other on actions in H, and forbidding communication with the environment via those actions.

The μCRL data definitions used are not presented here; these are straightforward and examples of the appropriate types or similar may be found in [GP95, Lut97].

2.1 Specification

The most abstract specification of the tree identify protocol is the one which merely reports that a leader has been found. The network is viewed as a whole, and no communications between nodes are specified.

Definition 1 *(Specification of 1394)*

$$Spec = \overline{leader}.\delta$$

2.2 Abstract Description of the Protocol

We now describe the protocol in slightly finer detail. In this model, each node in the network is modelled by a separate process, and the whole is given by the parallel composition of these nodes.

Each node has seven parameters:

- $i : Nat$ the identification number of the node. This is used to parameterise communications between nodes, and is not changed during the operation of the protocol.

- $conns : Nat$Set the set of nodes (natural numbers) to which the current node is connected). This does not change during the protocol.

- $p : Nat$ the identifier of the (potential) parent of this node, initially 0, a dummy node.

- $child : Nat$Set the set of identifiers of children of this node, initially empty.

- s : *Nat* the current state of the node. In *1394A* there are two states: state 0 corresponds to "still working" and state 1 to "finished". Initial state is 0.
- w : *Nat* the number of processes still to communicate with this node (waiting). While the node is in state 0 $w = |Conns| - |Child|$, therefore w has initial value $|Conns|$.
- *root* : *Bool* true if this node is, or has the potential to be, the leader and false otherwise. Initially true for all nodes.

Appropriate data type definitions for these parameters are assumed.

Definition 2 *(Individual Nodes in 1394A)*
Formal specification of the actions and their communications:

$$\begin{array}{ll}
\textbf{act} & bmc, \overline{bmp}, c^* : Nat \times Nat \\
 & \overline{leader} \\
\textbf{comm} & bmc|\overline{bmp} = c^*
\end{array}$$

Individual nodes are specified thus:

$$\begin{array}{l}
N(i, conns, p, child, s, w, root) = \\
\quad \sum_{j:Nat} \; bmc(i, j).N(i, conns, p, child \cup \{j\}, s, w-1, root) \\
\qquad\qquad \lhd\; s = 0 \wedge w \geq 1 \wedge j \in conns \setminus child \;\rhd \delta + \\
\quad \sum_{j:Nat} \; \overline{bmp(j, i)}.N(i, conns, j, child, 1, w-1, false) \\
\qquad\qquad \lhd\; s = 0 \wedge w = 1 \wedge j \in conns \setminus child \rhd\; \delta + \\
\quad \overline{leader}.N(i, conns, p, child, 1, w, root) \lhd\; s = 0 \wedge w = 0 \rhd\; \delta
\end{array}$$

Intuitively, $bmc(i, j)$ is a request from node i to node j to "be my child". The converse message, $\overline{bmp}(j, i)$ is a request from node j to node i to "be my parent". When all the nodes of the network are composed in parallel (as in Definition 3) these two actions synchronise with each other to produce c^* actions.

It is not specified what happens in state 1 (this is the next stage of the bus reset, not considered here). By definition, nodes in state 1 are equivalent to deadlock.

The network is specified for an undetermined but bounded number of nodes.

Definition 3 *(Complete 1394A)*
The complete system is given by:

$$1394A(n : Nat, Conns : NatSetList) = \tau_I \partial_H NodesA(n, Conns)$$

where

$$\begin{array}{l}
NodesA(n, Conns) = \\
\qquad\qquad N(1, Conns[1], 0, \emptyset, 0, |Conns[1]|, true) \\
\qquad\qquad\quad \lhd\; n = 1 \;\rhd \\
\qquad\qquad (N(n, Conns[n], 0, \emptyset, 0, |Conns[n]|, true) \parallel NodesA(n-1, Conns))
\end{array}$$

and

$$I = \{c^*\}$$
$$H = \{bmc, \overline{bmp}\}$$

Conns is a list of sets of connections for all nodes, indexed by node number. Again, the appropriate data definitions are assumed.

Correctness of the simple *1394A* is discussed in the Section 3.3.

2.3 Concrete Description of the Protocol

The description of *1394A* assumed hand-shaking communication between nodes; in reality messages are sent by variations in voltage along wires of various lengths and are therefore not received instantaneously, i.e. they are asynchronous communications. This means a node may ask to be a child of its neighbour, while that neighbour has already sent out a message asking to be *its* child (but the messages have crossed in transmission). That contention has to be resolved, and one node assigned to be the parent and the other the child.

In the following specification unidirectional buffers (*Lines*) are introduced to model communication between nodes. There are two channels for each pair of nodes (those channels not required for communication will simply not be used because of the requirement for synchronisation between *Lines* and *NodesC* in the complete description given in Definition 6). The communication events also become more complex: in addition to the "be my parent" and "be my child" requests, nodes must also send acknowledgements (since a node cannot assume its requests are successful until acknowledgements are received). Therefore the data type *MType:* $\{parent,\ ack\}$ is introduced to model the different kinds of communication.

Definition 4 *(Communication Buffers)*

$$
\begin{array}{ll}
Line(i : Nat, j : Nat) & = \sum_{t:MType} s(i,j,t).\overline{r}(i,j,t).Line(i,j) \\
Lines(index : Nat, n : Nat) & = \\
\quad LineList(1,n) \lhd\ index = 1\ \rhd (LineList(index,n) \parallel Lines(index-1,n)) & \\
LineList(row : Nat, col : Nat) = & \\
\quad Line(row,1) \lhd\ col = 1\ \rhd (Line(row,col)) \parallel LineList(row,col-1)) &
\end{array}
$$

Again individual nodes of the network are specified by separate processes. The parameters are similar to those for the simple specification in Definition 2, except there are now two more states. Intuitively, state 0 corresponds to sending and receiving requests setting up the parent-child relationship. After sending its "be my parent" request, a node moves into state 1, sending acknowledgements to all its children, and waiting for an acknowledgement from its parent. At this point it may receive a "be my parent" request instead of an acknowledgement from its requested parent, so it moves into state 2, attempting to resolve contention.

In the standard, contention is resolved by waiting a randomly chosen time (short or long) before checking for a offer to be a child from the other node, and, if there is none, resending its own "be my parent" request. There is no time in μCRL, so here there is a nondeterministic choice between sending the parent request again and waiting to receive a child request. (Nondeterminism is introduced by the use of τ actions. It is not possible to preface both branches by τ as that would lead to deadlock, so we either receive a "be my parent" request from the other node, or nondeterministically commit to sending our own "be my parent" request.)

After the resolution attempt both nodes return to state 1; there may be other acknowledgements still to be received. State 3 corresponds to finished.

The role of w is also extended: it is used both to count down the number of remaining requests, and then to count up the number of acknowledgements sent.

A message $\overline{s}(i, j, parent)$ is a sent request from i to j to "be my parent" (the analogous message in *1394A* is $\overline{bmp(j,i)}$). Through the buffer, the $\overline{s}$ message is transformed into a $\overline{r}$ message, synchronising with r messages in other nodes. A message $r(j, i, parent)$ is therefore a received request from j to i to "be my parent" (the analogous message in *1394A* is $bmc(i,j)$). Acknowledgement messages $\overline{s}(i, j, ack)$ from i to j acknowledge that i will be j's parent. Lastly, the $\overline{done}$ message is introduced. This occurs when a node which cannot be leader has received all acknowledgements and can move to the finished state.

There is a special case in this node for having no connections; we therefore assume a network of size 1, and the node can declare itself leader. This case was implicit in *1394A*.

Definition 5 *(Individual Nodes in 1394C)*
The actions and their communications are specified formally:

$$
\begin{array}{ll}
\textbf{act} & r, \overline{r}, r^*, s, \overline{s}, s^* : \mathit{Nat} \times \mathit{Nat} \times \mathit{MType} \\
 & \overline{leader}, done \\
\textbf{comm} & r|\overline{r} = r^* \\
 & s|\overline{s} = s^*
\end{array}
$$

Individual nodes are specified thus:

$$
\begin{array}{l}
P(i, conns, p, child, s, w, root) = \\
\quad \overline{leader}.P(i, conns, p, child, 3, w, root) \lhd\ s = 0 \wedge |conns| = 0 \ \rhd \delta + \\
\quad \sum_{j:Nat} r(j, i, parent).P(i, conns, p, child \cup \{j\}, s, w-1, root) \\
\qquad\quad \lhd\ s = 0 \wedge w > 1 \wedge j \in conns \setminus child \ \rhd \delta + \\
\quad \sum_{j:Nat} (r(j, i, parent).P(i, conns, p, child \cup \{j\}, 1, w-1, root) + \\
\qquad\qquad \overline{s}(i, j, parent).P(i, conns, j, child, 1, w-1, root)) \\
\qquad\quad \lhd\ s = 0 \wedge w = 1 \wedge j \in conns \setminus child \ \rhd\ \delta + \\
\quad \sum_{j:Nat} \overline{s}(i, j, ack).P(i, conns, p, child \setminus \{j\}, s, w+1, root) \\
\qquad\quad \lhd\ s = 1 \wedge j \in child \ \rhd \delta + \\
\quad (r(p, i, ack).P(i, conns, p, child, s, w, false) + \\
\qquad\quad r(p, i, parent).P(i, conns, 0, child, 2, w, root)) \lhd\ s = 1 \ \rhd \delta + \\
\quad \overline{leader}.P(i, conns, p, child, 3, w, root) \lhd\ s = 1 \wedge |conns| = w \ \rhd \delta + \\
\quad done.P(i, conns, p, child, 3, w, root) \lhd\ s = 1 \wedge \neg root \wedge empty(child) \ \rhd \delta + \\
\quad \sum_{j:Nat} (r(j, i, parent).P(i, conns, p, child \cup \{j\}, 1, w, root) + \\
\qquad\qquad \tau.\overline{s}(i, j, parent).P(i, conns, j, child, 1, w, root)) \\
\qquad\quad \lhd\ s = 2 \wedge j \in conns \setminus child \ \rhd \delta
\end{array}
$$

Definition 6 *(Complete 1394C)*
The system as a whole is given by

$$1394C(n : Nat, Conns : NatSetList) = \tau_I \partial_H(NodesC(n, Conns) \parallel Lines(n, Conns))$$

where

$$
\begin{array}{l}
NodesC(n : Nat, Conns : NatSetList) = \\
\quad P(1, Conns[1], 0, \emptyset, 0, |Conns[1]|, true) \\
\qquad \lhd\ n = 1 \ \rhd \\
\quad (P(n, Conns[n], 0, \emptyset, 0, |Conns[n]|, true) \parallel NodesC(n-1, Conns))
\end{array}
$$

and

$$
\begin{array}{l}
I = \{s^*, r^*, done\} \\
H = \{r, \overline{r}, s, \overline{s}\}
\end{array}
$$

Correctness of the *1394C* is discussed in the next section.

3 Correctness

Correctness of the *1394A* and *1394C* is given by showing them equivalent in some sense to the very simple specification of Definition 1. This paper gives a proof of correctness of the *1394A*, but only a sketch of the correctness of *1394C*; this is work still in progress. First, the technique used is introduced.

In process algebra it is common to verify the correctness of a description (the implementation) by proving it equivalent in some sense, e.g. with respect to strong

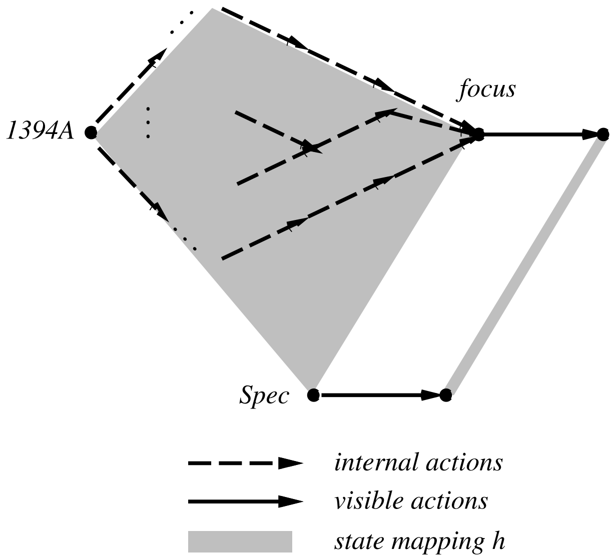

Figure 1: Cone and Focus for *1394A*

bisimulation, to a more abstract specification. When data is introduced to the descriptions proving equivalence is more complex; data can considerably alter the flow of control in the process.

To prove equivalence of two descriptions with data (call them the implementation and the specification) we use the cones and foci technique of [GS95]. The main idea of this technique is that there are usually many internal events in the implementation, but they are only significant in that they must progress somehow towards producing a visible event which can be matched with a visible event in the specification. The point at which the visible event can occur is the *focus*, and there may be several such points in the implementation. In the *1394* descriptions, the focus comes when the implementation, i.e. *1394A* or *1394C*, can perform the $\overline{leader}$ action. The focus point can be characterised by a boolean condition on data of the process. (Namely, the negation of the condition which allows τ actions to occur.)

The *cone* relates to the idea that no matter how divergent internal actions at the beginning of the implementation may seem, eventually the transitions will converge at the focus; imagine the transition system forming a cone or funnel pointing towards the focus.

The final element in the technique is a mapping between the data of the specification and the data of the implementation which captures the relation between the two specifically with respect to values in the cone before the focus point, just at the focus, and after the focus.

These elements are displayed pictorially in Figure 1.

Equivalence between the two systems can then be shown by proving six "matching criteria" hold. Informally, these say

1. The implementation must be convergent.

2. Internal actions in the implementation preserve the mapping.

3. If the implementation can do a visible action then so can the specification.

4. If the specification can do a visible action and the focus condition holds, then so can the implementation.

5. The implementation and the specification have the same data (modulo the mapping) on visible actions.

6. If the implementation does a visible action then the mapping is preserved afterwards.

If these six criteria are satisfied then the specification and the implementation can be said to be branching bisimilar under the General Equality Theorem (Theorem 3.3) of [GS95].

The general forms of the matching criteria are given in [GS95], and the user of the technique need only supply the particular actions, conditions and mapping for their system. This part of the verification is completely mechanical, although the choice of mapping requires some thought, as does the subsequent *proof* of the matching criteria.

A requirement of this method is that the process be in linear form, i.e. using only the choice and sequence operators. This is a common transformation is process algebra. The linearisation technique of [Gro96] provides rules for the transformation in the special case that the system is composed of n similar processes (as in *1394A* and *1394C*). Also, Theorem 3.3 of [GS95] mentions an invariant on data, which serves to eliminate the unreachable parts of the implementation. In this case the data invariant is *Good Topology* (Definition 7). This definition, and other preliminary definitions, are given in the next section.

3.1 Preliminaries

3.1.1 Good Topology

As mentioned earlier, the *1394* operates correctly only on tree networks, i.e. assuming the network has a good topology. Networks with loops will cause a timeout in the real protocol, and unconnected nodes will simply be regarded as another network. The property of *Good Topology* is formalised here.

The graph specified by the network connections is given by the information in *Conns*, an array indexed by naturals (node numbers) containing sets of connections (also naturals) for each node.

Definition 7 *(Trees)*
Given n, the number of nodes in the network:

- $Conns$ is symmetric: $\forall i, j.(i \in Conns[j] \Leftrightarrow j \in Conns[i])$
- $Conns$ is a tree, i.e. it is a connected graph with no loops.
  - connected: there exists a path[1] s between every pair of nodes.
    $\forall k, j \leq n. \exists s = i_0 \ldots i_m.(i_0 = k \wedge i_m = j)$
  - no loops : $\forall i. \neg\exists$ direct path $s = i_0 i_1 \ldots i_m.(i = i_0 \wedge i = i_m)$.

All these conjoined are called $GoodTopology(n : Nat, Conns : NatSetList)$. The parameters are often omitted since they are clear from the text.

3.1.2 Invariants

The proof of correctness also requires a number of invariants.

Lemma 1 *(1394A Invariants)*

1. $\forall i.(S[i] = 1 \ \wedge \ W[i] = 0) \Rightarrow Conns[i] = Child[i] \cup \{p\}$
2. $\forall i.(S[i] = 0 \ \wedge \ W[i] = 0) \Rightarrow Conns[i] = Child[i]$
3. $\forall i, j. j \in Child[i] \Rightarrow (S[j] = 1 \ \wedge \ W[i] = 0)$
4. $\forall i.(Root[i] \wedge \exists j.S[j] = 0) \Rightarrow S[i] = 0$
5. $\forall i, j.(S[i] = 0 \ \wedge \ j \in Conns[i] \setminus Child[i])$
   $\Rightarrow (i \in Conns[j] \setminus Child[j] \ \wedge \ S[j] = 0 \ \wedge \ W[j] \geq 1)$

The proofs of these are straightforward, and omitted here.

3.2 Linearisation of the Specification

As a preliminary step to applying Theorem 3.3 of [GS95], the specification of Definition 1 must be translated into a linear form. Additionally, a data parameter must be added on which to base a mapping from the data of the *1394A* or *1394C* (as required by the cones and foci verification technique).

Definition 8 *(Linear specification)*
$S(b : Bool) = \overline{leader}.S(false) \ \triangleleft \ b \ \triangleright \ \delta$

Clearly $S(true) = Spec$.

[1] Define *paths*, $s : NatList = i_0 i_1 \ldots i_m$ such that $\forall k < m. i_{k+1} \in Conns[i_k]$. *Direct paths* are paths which do not backtrack down an edge already followed (remember *Conns* is symmetric). A path s is direct if $\forall k < m.(i_k i_{k+1} \in s) \Rightarrow (i_{k+1} i_k \notin s)$.

3.3 Correctness of the Abstract 1394A

3.3.1 Linearisation of the Abstract 1394

The description of the implementation, *1394A*, must also be linearised.

The behaviour of the whole network is given by the linearisation, therefore the data for the individual nodes must be collated and given to the linearised process as a number of arrays. New list types are defined which correspond to the parameters of *1394A*, e.g. $S : NatList$ is a list of states for all nodes, indexed by node number. As before, *Conns* is given, and all the other lists are constructed from information in *Conns*. For recursive calls of *AbstractL* only those arguments which are updated are given. e.g. $AbstractL(S[k] := 1)$ means update only the kth element of S, all other elements remaining the same. The τ action arises from synchronisations of *bmp* and $\overline{bmc}$.

Definition 9 *(Linear Abstract Implementation)*

$$
\begin{array}{l}
AbstractL(n : Nat, Conns : NatSetList, P : NatList, Child : NatSetList, \\
\qquad S : NatList, W : NatList, Root : BoolList) = \\
\quad \sum_{k:Nat} \overline{leader}.AbstractL(S[k] := 1) \triangleleft S[k] = 0 \wedge W[k] = 0 \wedge k \leq n \triangleright \delta + \\
\quad \sum_{k_1:Nat} \sum_{k_2:Nat} \tau. \\
\qquad AbstractL(P[k_1] := k_2, Child[k_2] := Child[k_2] \cup \{k_1\}, S[k_1] := 1, \\
\qquad\qquad W[k_2] := W[k_2] - 1, W[k_1] := W[k_1] - 1, Root[k_1] := false) \\
\qquad \triangleleft (S[k_1] = 0 \wedge W[k_1] = 1 \wedge k_2 \in (Conns[k_1] \setminus Child[k_1]) \\
\qquad\quad \wedge\ S[k_2] = 0 \wedge W[k_2] \geq 1 \wedge k_1 \in (Conns[k_2] \setminus Child[k_2]) \\
\qquad\quad \wedge\ k_1 \neq k_2 \wedge k_1 \leq n \wedge k_2 \leq n) \triangleright \delta
\end{array}
$$

where initial values are

$$\forall i.(P[i] = 0 \ \wedge \ Child[i] = \emptyset \ \wedge \ S[i] = 0 \ \wedge \ W[i] = |Conns[i]| \ \wedge \ Root[i] = true)$$

By Theorem 3.5 of [Gro96] (linearisation of n similar processes)

$$AbstractL(n, Conns, P, Child, S, W, Root) = 1394A(n, Conns, P, Child, S, W, Root)$$

This linearisation is not sufficient to allow us to apply the General Equality Theorem of [GS95]. The indices of the sums preceding any visible actions must be the same in both the specification and the implementation; however, in the specification of Definition 8 there is no sum in the specification at all, and in the specification of Definition 9 there is a sum over k preceding the $\overline{leader}$ action. This correctly reflects that any node can be the root. There are multiple foci; one for each node, to be precise. See the diagram in Figure 2. However, it is not important *which* node is the root, only that one is chosen, and the boolean condition guarding the $\overline{leader}$ action in Definition 9 ensures that this is the case, summed up in Lemma 2.

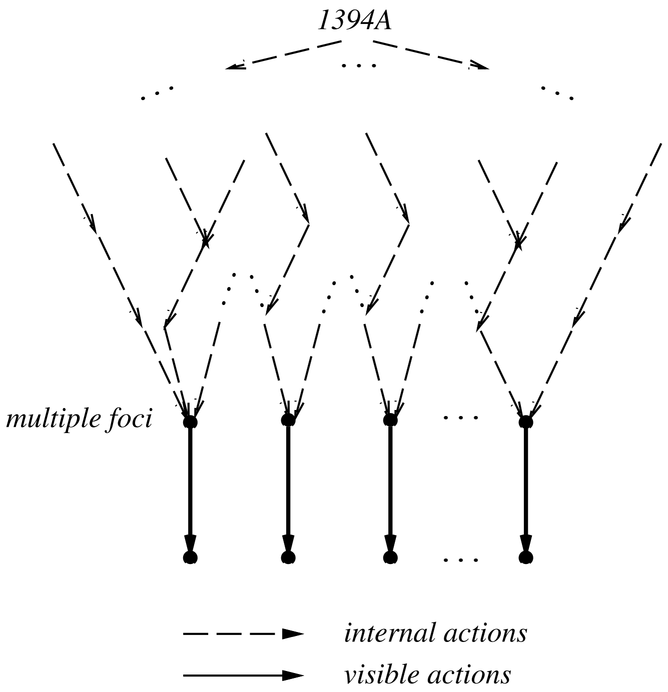

Figure 2: Cones of the *1394A*

Lemma 2 *(Uniqueness of Root)*

$$\forall k : Nat.(S[k] = 0 \wedge W[k] = 0 \wedge k \leq n) \Rightarrow \forall j : Nat.j \neq k, j \leq n.S[j] = 1$$

Proof. By contradiction. Assume a node j for which $S[j] = 0$. A path can be constructed $i_0 \ldots i_m$ from $k = i_0$ to $j = i_m$ (by *GoodTopology*). For all p, $0 \leq p < m$, there is a link from i_p to i_{p+1} in the path. By Invariant 2 $i_1 \in Child[i_0]$, and by Invariant 3 $S[i_1] = 1 \wedge W[i_1] = 0$. Thereafter, for $p > 0$, Invariant 1 (and knowing the parent of i_p is i_{p-1} by the previous step) gives $i_{p+1} \in Child[i_p]$ and Invariant 3 gives $S[i_{p+1}] = 1 \wedge W[i_{p+1}] = 0$. In particular, when $p = m-1$ then $S[i_m] = 1$, i.e. $S[j] = 1$, contradicting the original assumption. ∎

The information that the root node chosen is unique (once a certain point in the cone is reached) can be exploited to give a new definition of the linearisation of *1394A*.

Introduce $pr : Nat.pr \leq n$, the smallest index of S such that $Root[pr]$ holds, i.e. pr is a function ranging over $1..n$ which chooses the first i for which $Root[i]$ holds. If $\forall i.\neg Root[i]$ then $pr = 0$. At most points in the cone there will be several nodes for which *Root* holds, choosing the smallest turns out to be convenient in the proof, but does not affect the eventual choice of root; Lemma 2 shows that, at a certain point in the cone (at the focus) the root has been chosen and is unique.

Since the condition for the first summand of Definition 9 is the left hand side of Lemma 2, only one k will satisfy the condition, namely that denoted by pr at the focus, so substitute pr for k in $AbstractL$ to obtain AL.

Definition 10 *(New Linear Abstract Description)*

$$
\begin{array}{l}
AL(n, Conns, P, Child, S, W, Root) = \\
\quad \overline{leader}.AL(S[pr] := 1) \lhd S[pr] = 0 \wedge W[pr] = 0 \wedge pr \leq n \rhd \delta + \\
\quad \sum_{k_1:Nat} \sum_{k_2:Nat} \ \tau. \\
\qquad AL(P[k_1] := k_2, Child[k_2] := Child[k_2] \cup \{k_1\}, S[k_1] := 1, \\
\qquad\quad W[k_2] := W[k_2] - 1, W[k_1] := W[k_1] - 1, Root[k_1] := false) \\
\qquad \lhd (S[k_1] = 0 \wedge W[k_1] = 1 \wedge k_2 \in (Conns[k_1] \setminus Child[k_1]) \\
\qquad\ \ \wedge\ S[k_2] = 0 \wedge W[k_2] \geq 1 \wedge k_1 \in (Conns[k_2] \setminus Child[k_2]) \\
\qquad\ \ \wedge\ k_1 \neq k_2 \wedge k_1 \leq n \wedge k_2 \leq n) \rhd \delta
\end{array}
$$

By Lemma 2,

$$AL(n, Conns, P, Child, S, W, Root) = AbstractL(n, Conns, P, Child, S, W, Root)$$

The theorem to be demonstrated now can be stated as:

Theorem 1
Correctness of the abstract *1394A* under the assumption of *GoodTopology* is given by

$$GoodTopology \Rightarrow (\tau.S(true) = \tau.AL(n, Conns, P, Child, S, W, Root))$$

with initial values as before.

The τ steps have been added because the verification is with respect to branching bisimulation. If we had only $S = AL$ then, with $n > 1$ nodes in the network, AL can do several τ actions which S cannot match, so try $\tau.S = AL$. However, in that case, with $n = 1$ the specification can do a τ action which AL cannot match, hence the formulation of the theorem above.

In order to prove Theorem 1 using the General Equality Theorem of [Gro96] the matching criteria must be satisfied.

3.3.2 Focus Condition, State Mapping and Matching Criteria for 1394A

To show the matching criteria hold we first define the focus condition FC and the mapping h for AL (i.e. *1394A*).

The *Focus Condition*, FC, is the condition under which no more τ steps can be made, i.e. it is the negation of the condition for making a τ step.

Definition 11 *(Focus Condition)*

$$
\begin{array}{ll}
FC = \forall k_1, k_2. & S[k_1] \neq 0 \vee W[k_1] \neq 1 \vee k_2 \notin (Conns[k_1] \setminus Child[k_1]) \\
& \vee\ S[k_2] \neq 0 \vee W[k_2] < 1 \vee k_1 \notin (Conns[k_2] \setminus Child[k_2]) \\
& \vee\ k_1 = k_2 \vee k_1 > n \vee k_2 > n
\end{array}
$$

The state mapping h is a function, $NatList \Rightarrow Bool$, mapping (selected) data from the implementation into the data of the simple specification. In this case the data selected is S, the list of states of nodes, and h is defined so that it is *true* before the visible *leader* action occurs and *false* afterwards.

Definition 12 *(State Mapping for 1394A)*

$$h(S) = (S[pr] = 0)$$

Intuitively h says that as long as the possible root, pr, introduced in the last section, still has state 0 then the $\overline{leader}$ action has not yet occurred.

In order to prove Theorem 1 we must show the following six conditions hold (the matching criteria of [GS95]). In the following the notation S' is used to indicate the value of S after some action has occurred; similarly for pr.

The matching critera are mechanically derived from the general forms of [GS95], given the particulars of S, AL, FC and h.

Lemma 3 *(Matching Criteria)*

1. The implementation is convergent, i.e. there are no infinite τ loops.

2. τ steps in AL preserve h.

   $\forall k_1, k_2.(S[k_1] = 0 \;\wedge\; W[k_1] = 1 \;\wedge\; k_2 \in (Conns[k_1] \setminus Child[k_1]) \;\wedge\; S[k_2] = 0 \;\wedge\; W[k_2] \geq 1 \;\wedge\; k_1 \in (Conns[k_2] \setminus Child[k_2]) \;\wedge\; k_1 \neq k_2 \;\wedge\; k_1 \leq n \;\wedge\; k_2 \leq n) \Rightarrow h(S) = h(S')$

3. If AL can perform a $\overline{leader}$ action then so can S.

   $(S[pr] = 0 \wedge W[pr] = 0 \wedge pr \leq n) \Rightarrow h(S)$.

4. If S can perform a $\overline{leader}$ action and FC holds then AL can also perform a $\overline{leader}$ action.

   $(h(S) \wedge FC) \Rightarrow (S[pr] = 0 \wedge W[pr] = 0 \wedge pr \leq n)$

5. If AL can perform a $\overline{leader}$ action then S can perform a $\overline{leader}$ action with the same data.

6. If AL performs a $\overline{leader}$ action then the mapping h is preserved afterwards.

   $(S[pr] = 0 \wedge W[pr] = 0 \wedge pr \leq n) \Rightarrow \neg h(S')$

Proof.

1. For each τ step the number of nodes for which $S[i] = 0$ holds decreases by 1 (because $S[k_1] := 1$ in Definition 10). Eventually the τ condition (requiring $S[k_1] = 0 \ \wedge \ S[k_2] = 0$) will fail.

2. The result of h depends on pr, therefore we do a case analysis on values of pr:

   $pr \neq k_1$ Then pr is preserved since $S[pr]$ is still 0, and $Root[pr]$ still holds. It does not matter if $pr = k_2$ here because these parameters are not altered for k_2 either.

   $pr = k_1$ $Root'[k_1]$ is *false*, therefore $pr \neq pr'$. From Invariant 4 we have that whichever new pr is chosen, the state is 0 (because we know at least $S[k_2] = 0$).

3. Trivial.

4. Assume $h(S)$ and FC. Consider the right hand side of the implication. $h(s) \Rightarrow S[pr] = 0$, and by definition $pr \leq n$, therefore assume $W[pr] \neq 0$, i.e. $W[pr] > 0$, and try to derive a contradiction.

   Since $W[pr] > 0$, $\exists j.j \in Conns[pr] \setminus Child[pr]$, and, from Invariant 5, $W[j] \geq 1$. Substituting pr and j in FC gives $W[j] \neq 1$, therefore that condition can be strengthened to $W[j] > 1$. Following similar reasoning at each step, an infinite path can be constructed, violating *GoodTopology* and thus deriving a contradiction.

5. Trivial. The action $\overline{leader}$ involves no data.

6. From Definition 10, performing the $\overline{leader}$ action has no effect on $Root[pr]$. Therefore, $pr = pr'$, and $S'[pr'] = 1$, also from Definition 10, and from Lemma 2 no other node can be pr therefore $h(S') = false$.

∎

From these and Theorem 3.3 of [GS95] it follows that Theorem 1 holds.

3.4 Sketch of Correctness of the Complex 1394C

In this case the procedure is more complicated because contention results in τ loops within the cone (therefore the implementation is not convergent). Fortunately, [GS95] has, in addition to the General Equality Theorem used in the previous section, a version which is extended by notions of progression and fairness to counteract the problem of implementations with τ loops. (This is Theorem 4.9 of [GS95].)

3.4.1 Linearisation of 1394C

The procedure of linearisation for the more complex *1394C* is similar to that of the simple version. First the definitions of *Lines* and *NodesC* are linearised separately (using the technique of [Gro96], as before) and then the results are combined (manually) using standard process algebra rules for communicating processes.

The problem is that the resulting definition has a τ loop; namely, when there is contention during the negotiation of the parent-child relation the nodes enter state 2, some variables are reset and the nodes return to state 1 again. Rather than force a particular resolution of contention the description of Definition 5 relies on nondeterminism, so the nodes could easily enter the contention state again and again.

The solution of [GS95] is to introcude a **pre-abstraction function** ξ. This function distinguishes progressing from non-progressing internal actions. In this case, non-progressing actions occur when a "be my parent" request is sent when there is already a contradictory request in the system. The pre-abstraction function is formalised below.

Definition 13 *(Pre-abstraction Function)*
The pre-abstraction function is defined by

$$\xi(a) = \begin{cases} \neg(P[j] = k) & a = s^*(k, j, parent) \\ true & \text{otherwise} \end{cases}$$

There are nine different kinds of τ actions in the linearised *1394C* as given in Definition 14; each one results from different communications of s and r actions. The pre-abstraction function is used to modify the behaviour of the linearised *1394C*, by distinguishing progressing and non-progressing actions and adding the progression condition to the guards associated with progressing actions. The non-progressing actions have all been renamed to *dummy*, therefore all τ actions are progressing, and only *dummy* actions are non-progressing.

The defintion of the linear *1394C* is given here for information, although it is, admittedly, rather difficult to read. The statement of Theorem 4.9 of [GS95] (the General Equality Theorem for systems with τ loops) requires that we add similar *dummy* transitions to the specification. We do not show this here.

Definition 14 *(Linear 1394C)*
Assuming an appropriate linearisation of Nodes and Lines, and the pre-abstraction function above.

$$
\begin{array}{l}
\mathit{1394CL}(N, Conns, P, Child, S, W, Root) = \\
\quad \overline{leader}.1394CL(S[pr] := 3) \lhd\ S[pr] = 0 \wedge |Conns[pr]| = 0 \wedge pr \leq n \ \ \rhd\ \delta + \\
\quad \overline{leader}.1394CL(S[pr] := 3) \lhd\ S[pr] = 1 \wedge |Conns[pr]| = W[pr] \wedge pr \leq n \ \ \rhd\ \delta + \\
\quad \sum_{k:Nat} \sum_{j:Nat} \tau./*r^*(j,k,parent)*/ \\
\qquad 1394CL(Child[k] := Child[k] \cup \{j\}, W[k] := W[k]-1, B[j,k] := \neg B[j,k]) \\
\qquad\quad \lhd\ S[k] = 0 \wedge W[k] > 1 \wedge j \in Conns[k] \setminus Child[k] \wedge k \leq n \wedge j \leq n \\
\qquad\quad \wedge B[j,k] \wedge Type[j,k] = parent\ \rhd \delta + \\
\quad \sum_{k:Nat} \sum_{j:Nat} \tau./*r^*(j,k,parent)*/ \\
\qquad 1394CL(Child[k] := Child[k] \cup \{j\}, S[k] := 1, W[k] := W[k]-1, B[j,k] := \neg B[j,k]) \\
\qquad\quad \lhd\ S[k] = 0 \wedge W[k] = 1 \wedge j \in Conns[k] \setminus Child[k] \wedge k \leq n \wedge j \leq n \\
\qquad\quad \wedge B[j,k] \wedge Type[j,k] = parent\ \rhd\ \delta + \\
\quad \sum_{k:Nat} \sum_{j:Nat} /*s^*(k,j,parent)*/ \\
\qquad (\tau.1394CL(P[k] := j, S[k] := 1, W[k] := W[k]-1, \\
\qquad\quad Type[k,j] := parent, B[k,j] := \neg B[k,j]) \\
\qquad \lhd P[j] \neq k\ \rhd \\
\qquad dummy.1394CL(P[k] := j, S[k] := 1, W[k] := W[k]-1, \\
\qquad\quad Type[k,j] := parent, B[k,j] := \neg B[k,j])) \\
\qquad\quad \lhd S[k] = 0 \wedge W[k] = 1 \wedge j \in Conns[k] \setminus Child[k] \wedge k \leq n \wedge \neg B[k,j] \wedge j \leq n\ \rhd\ \delta + \\
\quad \sum_{k:Nat} \sum_{j:Nat} \tau./*s^*(k,j,ack)*/ \\
\qquad 1394CL(Child[k] := Child[k] \setminus \{j\}, W[k] := W[k]+1, \\
\qquad\quad Type[k,j] := ack, B[k,j] := \neg B[k,j]) \\
\qquad\quad \lhd\ S[k] = 1 \wedge j \in Child[k] \wedge k \leq n \wedge \neg B[k,j] \wedge j \leq n\ \rhd \delta + \\
\quad \sum_{k:Nat} \tau./*r^*(P[k],k,ack)*/ \\
\qquad 1394CL(Root[k] := \mathit{false}, B[P[k],k] := \neg B[P[k],k]) \\
\qquad\quad \lhd\ S[k] = 1 \wedge k \leq n \wedge B[P[k],k] \wedge P[k] \leq n \wedge Type[P[k],k] = ack\ \rhd \delta + \\
\quad \sum_{k:Nat} \tau./*r^*(P[k],k,parent)*/ \\
\qquad 1394CL(P[k] := 0, S[k] := 2, B[P[k],k] := \neg B[P[k],k]) \\
\qquad\quad \lhd\ S[k] = 1 \wedge k \leq n \wedge B[P[k],k] \wedge P[k] \leq n \wedge Type[P[k],k] = parent\ \rhd \delta + \\
\quad \sum_{k:Nat} done.1394CL(S[k] := 3) \\
\qquad \lhd\ S[k] = 1 \wedge \neg Root[k] \wedge Empty(Child[k]) \wedge k \leq n\ \rhd \delta + \\
\quad \sum_{k:Nat} \sum_{j:Nat} \tau./*r^*(j,k,parent)*/ \\
\qquad 1394CL(Child[k] := Child[k] \cup \{j\}, S[k] := 1, B[j,k] := \neg B[j,k]) \\
\qquad\quad \lhd\ S[k] = 2 \wedge j \in Conns[k] \setminus Child[k] \wedge k \leq n \wedge B[j,k] \wedge \\
\qquad\quad Type[j,k] = parent \wedge j \leq n\ \rhd \delta + \\
\quad \sum_{k:Nat} \sum_{j:Nat} /*s^*(k,j,parent)*/ \\
\qquad (\tau.1394CL(P[k] := j, S[k] := 1, B[k,j] := \neg B[k,j], Type[k,j] := parent) \\
\qquad \lhd P[j] \neq k\ \rhd \\
\qquad \tau.dummy.1394CL(P[k] := j, S[k] := 1, B[k,j] := \neg B[k,j], Type[k,j] := parent)) \\
\qquad\quad \lhd\ S[k] = 2 \wedge j \in Conns[k] \setminus Child[k] \wedge k \leq n \wedge \neg B[k,j] \wedge j \leq n\ \rhd \delta
\end{array}
$$

The state mapping is also more complex this time. In *1394A* the $\overline{leader}$ action was

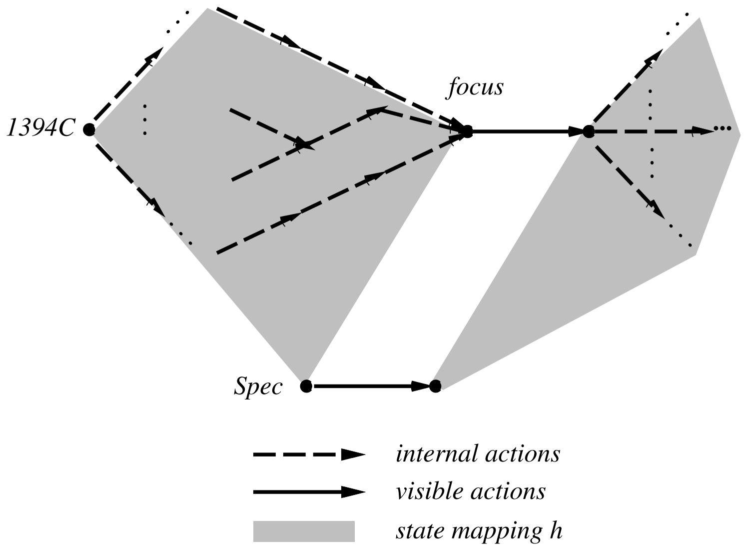

Figure 3: Cones and Foci

the last action to occur. In *1394C* it is possible that some nodes are still sending out acknowledgements after the $\overline{leader}$ action has occurred. See Figure 3.

Definition 15 *(State Mapping for 1394C)*
As before, the mapping is only concerned with values of states.

$$h(S) = (S[pr] \neq 3)$$

where *pr* is as before.

The correctness of *1394C* can be stated as follows.

Theorem 2
With initial values as for Theorem 1,

$$GoodTopology \Rightarrow (\tau.S'(true) = \tau.1394CL(n, Conns, P, Child, S, W, Root))$$

The main difficulty of the proof of correctness now is the sheer size of the focus condition (the negation of the conjunction of all conditions preceding progressing τ actions). As mentioned already, this is work in progress.

4 Conclusions

We have described the tree identify protocol of the 1394 multimedia serial bus. This was an exercise in specification using μCRL and in verification using the cones and foci technique. While no errors were identified in this view of the system, the exercise has been worthwhile for a number of reasons.

One of our original goals was to "test" the verification technique. We mentioned at the beginning that uptake of verification techniques is often slow due to their complexity. The cones and foci technique has a simple and appealing principle at its heart, and provides a useful structure for the verification, but, as has been seen here, is complex to apply. In particular it relies on expertise in the domain, experience in applying the technique to other examples, and some mathematical creativity! This is true of many formal methods.

To aid the verification process it is essential to have good tool support. It should be straightforward to automate parts of the technique of [GS95] used here. In particular, the initial linearisation can be generated automatically, and some development in this area is underway. In fact, computer checked proofs using this technique are described in [KS96]. Note, however, that in the study described here the proof process fed back into the description, in that it was impossible to prove the matching criteria held with the original linearisation of *1394A*. At that point experience and creativity stepped in and the function *pr* was introduced, altering the description of the system and therefore the matching criteria and making the proof possible.

The matching criteria can be automatically generated given the linear specification and implementation, and the state mapping h. Automation of this and linearisation would leave the verifier free to consider the more tricky questions of the definition of h and the proofs of the matching criteria. Several proof assistants exist which could be used to computer check such proofs, eliminating the possibility of manually introduced errors. If a more powerful tool such as HOL [GM93] were used then it may also be possible to use higher level tactics to aid the proof process. An interesting problem might be to examine a number of case studies using this verification technique to try to extract some general principles which could be coded in some specialised tactics.

In order for this to be possible, a number of studies must be carried out. Our second acheivement is that ours is one example, and adds to the body of experience in applying formal methods; however, at present there are too few examples of the application of [GS95] to allow us to draw any useful conclusions. From the limited set of examples available, we note that the verification of a distributed summation algorithm presented in [GS96] does have similar features (the use of n similar processes to describe the system, state-based descriptions, the use of state to define the mapping function, a simple boolean data in the specification and the *GoodTopology* invariant). With more case studies it may turn out that these are all common features of specification and verification of distributed systems in μCRL.

This proof technique compares favourably with earlier proofs in μCRL, e.g. [GK94, FGK97], which relied on much lower level proof rules (the usual rules for manipulating process algebra expressions), although we note that the proof given in [FGK97] contains some similar features to the specifications here and in [GS96] (state based specification, n similar processes). The cones and foci technique allows the verifier to concentrate on features of the data, and the structure of the proof technique takes care of the process algebra part.

This proof technique also contrasts with the approaches of [GM97] in which automated proofs of branching bisimulation are carried out using the CADP toolbox, and [SM97] which again uses the CADP toolbox, but this time to check the validity of modal formulae with respect to labelled transition systems generated from the descriptions. In both cases the size of the system must be restricted in order to allow automated checking. These may then be useful as a prototype stage; automated verification on a small number of nodes, followed by assisted verification on a bounded but undetermined number of nodes using techniques such as cones and foci.

Finally, we gained a greater understanding of how the tree identify protocol operates. In particular, an interesting observation is that if a node is extremely apathetic, sending no communications until absolutely necessary, then it will become the leader (assuming no other apathetic nodes in the network).

Acknowledgements

Thanks are due to Jan Friso Groote, who instigated this case study, for many helpful discussions regarding the application of the cones and foci verification technique. Thanks also to Judi Romijn and David Griffioen for discussions regarding the operation of the 1394 tree identify protocol. Finally, thanks to the PRG at the University of Amsterdam for providing a pleasant working environment, and to the EC for funding my visiting fellowship.

References

[FGK+96] J-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP (CAESAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox. In R. Alur and T.A. Henzinger, editors, *Proceedings of CAV'96*, number 1102 in Lecture Notes in Computer Science, pages 437–440. Springer-Verlag, 1996.

[FGK97] L. Fredlund, J.F. Groote, and H. Korver. Formal verification of a leader election protocol in process algebra. *Theoretical Computer Science*, 177(2):459–486, 1997.

[GK94] J.F. Groote and H. Korver. Correctness Proof of the Bakery Protocol in μCRL. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes*, Workshops in Computing, pages 63–86. Springer-Verlag, 1994.

[GM93] M.J.C. Gordon and T.F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[GM97] H. Garavel and L. Mounier. Specification and verification of various distributed leader election algorithms for unidirectional ring networks. *Science of Computer Programming*, 29(1-2):171–197, 1997.

[GP95] J.F. Groote and A. Ponse. The Syntax and Semantics of μ-CRL. In *Proceedings of Algebra of Communicating Processes, Utrecht 1994*, Workshops in Computing. Springer-Verlag, 1995.

[Gro96] J.F. Groote. A note on n similar parallel processes. Technical Report CS-R9626, Centrum voor Wiskunde en Informatica, Amsterdam, 1996.

[GRV97] W.O.D. Griffioen, J. Romijn, and F. Vaandrager. ColumBUS: An exercise in formal specification of real-time software. Draft manuscript, January 24, 1997.

[GS95] J.F. Groote and J. Springintveld. Focus Points and Convergent Process Operators. Technical Report 142, Logic Group of the University of Utrecht, 1995.

[GS96] J.F. Groote and J. Springintveld. Algebraic verification of a distributed summation algorithm. Technical Report CS-R9640, Centrum voor Wiskunde en Informatica, Amsterdam, 1996.

[IEE95] Institute of Electrical and Electronics Engineers. *P1394 Standard for a High Performance Serial Bus*, 1995. Draft 8.0v2a, July 7 1995.

[KS96] H.P. Korver and M.P.A. Sellink. On Automating Process Algebra Proofs. In V. Atalay et al., editors, *Proceedings of the 11-th International Symposium on Computer and Information Sciences, ISCIS XI, Antalya, Turkey,* volume II, pages 815–826, 1996.

[Lut97] S.P. Luttik. Description and formal specification of the Link Layer of P1394. Technical Report SEN-R9706, Centrum voor Wiskunde en Informatica, Amsterdam, 1997.

[SM97] M. Sighireanu and R. Mateescu. Validation of the Link Layer Protocol of the IEEE-1394 Serial Bus (FireWire): an Experiment with E-LOTOS. Technical Report 3172, INRIA, 1997.

[Vaa97] F. Vaandrager. Homework Assignment DSS: Verification of a Tree Identify Protocol. University of Nijmegen, 1997.