Formal Aspects of Computing (1998) 3: 1–000

The Tree Identify Protocol of IEEE 1394 in μCRL

Carron Shankland[1] and Mark van der Zwaag[2]

[1]Department of Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK; [2]CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Keywords: process algebra; verification of distributed systems; leader election protocols

Abstract. We specify the tree identify protocol of a high performance serial multimedia bus (IEEE Standard 1394 [IEE96]) at three different levels of detail using μCRL [GP95]. We use the cones and foci verification technique of Groote and Springintveld [GS95] to show that the descriptions are equivalent under branching bisimulation, thereby demonstrating that the protocol behaves as expected.

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. Case studies therefore have a valuable role to play both in promoting and demonstrating particular verification techniques, and providing practical examples of their application. This paper presents one such case study. We apply 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 [IEE96], 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

Correspondence and offprint requests to: Carron Shankland, Department of Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK, carron@cs.stir.ac.uk

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 lead 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 main operation of the standard, concerned with sending packets of information across 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 identify 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 identify 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 described using I/O automata in [DGRV97]. Verification is by (manual) manipulation of a number of invariants, phrased in predicate calculus. Also discussed is the mechanisation of this verification in the theorem prover PVS.

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

- a specification of the external behaviour of the protocol,
- Implementation A, and
- Implementation B.

The specification merely announces a single leader has been chosen. In Implementation A nodes are specified individually and negotiate with their neighbours to determine the parent-child relationship. Communication is by hand shaking. Implementation B has 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 2.2 respectively. They were derived with reference to the transition diagram in Section 4.4.2.2 of the standard [IEE96]. Section 3 gives an informal overview of the cones and foci technique of [GS95], together with some common definitions. The formal details of this technique are repeated in the appendix for convenience.

We prove, using the cones and foci technique, that Implementation A and Implementation B have the same behaviour with respect to branching bisimulation as the simple specification, therefore showing that these descriptions behave as required, i.e. a single leader is chosen. The proofs may be found in Section 4 and Section 5, respectively.

We conclude with some remarks about the success of this case study and about verification using the technique of [GS95] in general.

2. Description of the Tree Identify Protocol

The descriptions are given in μCRL, which is roughly ACP [BW90] extended with a formal treatment of data. Familiarity is assumed with this formalism; an introduction may be found in [GP95].

Briefly, the main features of the formalism are as follows. δ represents deadlock, $p \cdot q$ indicates sequential composition and $p + q$ indicates alternative composition. The process $\sum_{x:D} p(x)$ behaves as the possibly infinite choice between processes $p(d)$ where d is any data term of sort D. The parallel composition of processes p and q is written $p \parallel q$. We have a sort $\mathbb{B}$ of booleans with two elements t and f and the usual boolean operators. Conditionals are written $p \lhd b \rhd q$, meaning if b holds behave as p, otherwise behave as q. The operator τ_I hides all those actions in the set I, by converting them to silent τ actions, and ∂_H restricts enabled actions, by renaming actions in H to δ. We choose H such that the ∂_H operator forces the enclosed processes to communicate with each other. For booleans we assume the following binding conventions: $\neg$ binds stronger than $\wedge, \vee$, which bind stronger than $\rightarrow$.

The μCRL data definitions used (e.g. $\mathbb{N}$, $\mathbb{N}\mathit{Set}$, $\mathbb{N}\mathit{SetList}$) are assumed and 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. We define

$$\mathit{Spec} = \mathit{leader} \cdot \delta.$$

2.2. Implementation A

A more fine grained model is given by representing each node in the network by a separate process. Individual nodes are specified below as processes NodeA. Each node has three parameters:

- i:$\mathbb{N}$ is the identification number of the node. This is used to parameterise communications between nodes, and is not changed during the protocol.
- p:$\mathbb{N}\mathit{Set}$ is the set of node identifiers of potential parents of the node. The initial value is the set of all neighbours, decreasing to either a singleton (containing the parent node) or the empty set (indicating that the node is the elected leader).
- s:$\mathbb{N}$ is the current state of the node. We use two state values: 0 corresponds to "still working" and 1 to "finished". The initial value is 0.

A node can send and receive messages: an action $s(i, j, \mathit{par})$ is the sending of the parent request par by node i to node j, and an action $r(i, j, \mathit{par})$ is the receiving of a parent request from node i by node j. When the nodes of the network are composed in parallel, these two actions synchronise with each other to produce c actions. An action $c(i, j, \mathit{par})$ is the establishment of a child-parent relation between node i and node j. In this case, the type $\mathbb{M}$ of messages has only one element, i.e. the parent request message *par*.

We define the set of actions

$$Act = \{s, r, c : \mathbb{N} \times \mathbb{N} \times \mathbb{M}, leader\}$$

and the communication $s|r = c$. There are no other communications defined.

If a node is still active and its set of potential parents is empty, it declares itself leader by the execution of the *leader* action. By definition, nodes in state 1 are equivalent to deadlock. Individual nodes are defined as follows:

$$\begin{aligned}
&NodeA(i{:}\mathbb{N}, p{:}\mathbb{N}Set, s{:}\mathbb{N}) = \\
&\quad leader \cdot NodeA(i,p,1) \lhd s = 0 \wedge empty(p) \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} r(j,i,par) \cdot NodeA(i, p \setminus \{j\}, s) \lhd s = 0 \wedge j \in p \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} s(i,j,par) \cdot NodeA(i,p,1) \lhd s = 0 \wedge p = \{j\} \rhd \delta.
\end{aligned}$$

The process $ImpA(n, P_0)$ is the parallel composition of $n+1$ nodes, with P_0 describing the configuration of the network:

$$ImpA(n{:}\mathbb{N}, P_0{:}\mathbb{N}SetList) = \partial_H(NodesA(n, P_0)),$$

where $H = \{s, r\}$ and

$$\begin{aligned}
&NodesA(n, P_0) = \\
&\quad NodeA(0, P_0[0], 0) \lhd n = 0 \rhd \\
&\qquad (NodeA(n, P_0[n], 0) \parallel NodesA(n-1, P_0)).
\end{aligned}$$

P_0 is a list of sets of connections for all nodes, indexed by node number, and initially all nodes are in state 0. Note that since node identifiers start at 0, the process $ImpA(n, P_0)$ is the composition of $n+1$ nodes.

2.3. Implementation B

Implementation A 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 Implementation B unidirectional one-element buffers are introduced to model communication between nodes; there are two buffers for each pair of nodes. The communication events also become more complex: in addition to the parent requests, nodes must also send acknowledgements (since a node cannot assume its parent request is successful until an acknowledgement is received). Therefore we introduce the acknowledgement message *ack*. Let $\mathbb{M}$ be the sort of messages with two elements *par* and *ack*.

The parallel composition of all buffers is defined in Figure 1 as process *Buffers*. The names of actions in this definition may be confusing; for a buffer an s action is a *read* action and a $\bar{r}$ action is a *send* action. This is a consequence of the names used in the specification of nodes defined below.

Again individual nodes of the network are specified by separate processes.

$$Buffer(i{:}\mathbb{N}, j{:}\mathbb{N}) = \sum_{m:\mathbb{M}} s(i,j,m) \cdot \bar{r}(i,j,m) \cdot Buffer(i,j)$$
$$Buffers(index{:}\mathbb{N}, n{:}\mathbb{N}) =$$
$$BList(0,n) \lhd index = 0 \rhd (BList(index,n) \parallel Buffers(index-1,n))$$
$$BList(row{:}\mathbb{N}, col{:}\mathbb{N}) =$$
$$Buffer(row,0) \lhd col = 0 \rhd (Buffer(row,col)) \parallel BList(row,col-1))$$

Fig. 1. The process *Buffers*

The parameters are similar to those for Implementation A, except there are now three more states, and there is an extra parameter $c{:}\mathbb{N}Set$ that is used to keep track of children that have to be acknowledged.

In state 0, a node receives parent requests setting up the parent-child relationship. When it has received requests from all or all but one of its neighbours, it moves into state 1. In state 1 a node acknowledges its children. A node can leave state 1 by sending a parent request to its only remaining potential parent (if any). Leaf nodes can skip state 1, and go to state 2 immediately. In state 2, if a node has an empty potential parent set it is the leader and it can do a *leader* action. If not, a node waits for an acknowledgement from its parent. In state 2, a node may receive a parent request instead of an acknowledgement from its requested parent; it then moves into state 3, attempting to resolve contention.

In the standard, contention is resolved by waiting a randomly chosen time before checking for a offer to be a child from the other node, and, if there is none, resending its own parent request. There is no time in μCRL so here there is a choice between sending the parent request again and waiting to receive a child request. Note that there is the possibility of an internal loop if the nodes in contention keep sending each other parent requests. Contention is resolved if in the state where both nodes are in state 3, one of the nodes sends a parent request and the other node does not retransmit its own request, but waits to receive the request from the other node. After the contention has been resolved one of the nodes returns to state 1; this node has received a parent request from the other node and it has to acknowledge this new child. The other node moves into state 2 and waits to be acknowledged. State 4 corresponds to finished.

As for Implementation A, there is the special case where $n = 0$, i.e. there is only one node in the network. In this case this one node can do the leader action immediately.

An action $\bar{s}(i,j,par)$ is the sending of a parent request from i to j . Through the buffer, the $\bar{s}$ action is transformed into a $\bar{r}$ action, synchronising with r actions in other nodes. An action $r(j,i,par)$ is therefore the receiving of a parent request from j by i. Acknowledgements $\bar{s}(i,j,ack)$ from i to j acknowledge that i will be j's parent.

We define the set of actions

$$Act = \{r, \bar{r}, r^*, s, \bar{s}, s^*{:}\mathbb{N} \times \mathbb{N} \times \mathbb{M}, leader\}$$

and the communications $r|\bar{r} = r^*$ and $s|\bar{s} = s^*$. There are no other communications defined.

Individual nodes *NodeB* are specified in Figure 2. The complete process *ImpB* is the parallel composition of all nodes and buffers. Note that buffers not required for communication will simply not be used because of the requirement for synchronisation between *NodesB* and *Buffers*. We define

$$\mathit{ImpB}(n{:}\mathbb{N}, P_0{:}\mathbb{N}\mathit{SetList}) = \partial_H(\mathit{NodesB}(n, P_0) \parallel \mathit{Buffers}(n, n)),$$

where $H = \{r, \bar{r}, s, \bar{s}\}$ and

$$\begin{aligned}
&\mathit{NodesB}(n, P_0) = \\
&\quad \mathit{NodeB}(0, P_0[0], \emptyset, 0) \lhd n = 0 \rhd \\
&\qquad (\mathit{NodeB}(n, P_0[n], \emptyset, 0) \parallel \mathit{NodesB}(n-1, P_0)).
\end{aligned}$$

$$\begin{aligned}
&\mathit{NodeB}(i{:}\mathbb{N}, p{:}\mathbb{N}\mathit{Set}, c{:}\mathbb{N}\mathit{Set}, s{:}\mathbb{N}) = \\
&\quad \mathit{leader} \cdot \mathit{NodeB}(i, p, c, 4) \lhd (s = 0 \lor s = 2) \land \mathit{empty}(p) \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} r(j, i, \mathit{par}) \cdot \mathit{NodeB}(i, p \setminus \{j\}, c \cup \{j\}, \mathit{if}(\mathit{singleton}(p), 1, 0)) \\
&\qquad \lhd s = 0 \land j \in p \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} \bar{s}(i, j, \mathit{ack}) \cdot \mathit{NodeB}(i, p, c \setminus \{j\}, 1) \\
&\qquad \lhd s = 0 \land \mathit{singleton}(p) \land j \in c \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} \bar{s}(i, j, \mathit{par}) \cdot \mathit{NodeB}(i, p, c, 2) \\
&\qquad \lhd s = 0 \land p = \{j\} \land \mathit{empty}(c) \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} \bar{s}(i, j, \mathit{ack}) \cdot \mathit{NodeB}(i, p, c \setminus \{j\}, \mathit{if}(\mathit{empty}(p) \land \mathit{singleton}(c), 2, 1)) \\
&\qquad \lhd s = 1 \land j \in c \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} \bar{s}(i, j, \mathit{par}) \cdot \mathit{NodeB}(i, p, c, 2) \lhd s = 1 \land p = \{j\} \land \mathit{empty}(c) \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} r(j, i, \mathit{ack}) \cdot \mathit{NodeB}(i, p, c, 4) \lhd s = 2 \land p = \{j\} \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} r(j, i, \mathit{par}) \cdot \mathit{NodeB}(i, p, c, 3) \lhd s = 2 \land p = \{j\} \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} r(j, i, \mathit{par}) \cdot \mathit{NodeB}(i, p \setminus \{j\}, c \cup \{j\}, 1) \lhd s = 3 \land p = \{j\} \rhd \delta + \\
&\quad \textstyle\sum_{j:\mathbb{N}} \bar{s}(j, i, \mathit{par}) \cdot \mathit{NodeB}(i, p, c, 2) \lhd s = 3 \land p = \{j\} \rhd \delta
\end{aligned}$$

Fig. 2. The process *NodeB*

3. Correctness

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 bisimulation, to a more abstract specification. When data is introduced to the descriptions proving equivalence is more complex since data can considerably alter the flow of control in the process. The cones and foci technique of [GS95] addresses this problem. 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. A state of the implementation where no internal actions are enabled is called a *focus point*, and there may be several such points in the implementation. In Implementation A the focus comes when the implementation can perform the *leader* action, because the *leader* action is always the last action to be performed. In Implementation B there may be internal actions enabled in states where the *leader* action is enabled, and the focus comes when the *leader* action is the only enabled action. Focus points are characterised by a boolean condition on the data of the process called the *focus condition*. The focus condition is the negation of the condition which allows τ actions to occur. The *cone* belonging to a focus point is the part of the state space from which the focus can be reached by internal actions; imagine the transition system forming a cone or funnel pointing towards the focus. There may also be unreachable states in the implementation; these can be excluded by use of a data invariant.

The final element in the technique is a mapping between the data states of the implementation and the data states of the specification. This mapping is surjective, but almost certainly not injective, since the data of the specification is likely to be simpler than that of the implementation.

Equivalence between the two systems can then be shown by proving six "matching criteria" to 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 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 of [GS95] (repeated in the appendix here as Theorem A.1). The general forms of the matching criteria are given in Definition A.3. Given the particular actions, conditions and mapping for a system, the matching criteria can be mechanically derived. Of course, the choice of mapping requires some thought, as does the subsequent proof of the criteria.

In Section 5 we will see that for Implementation B, the procedure is more complicated. In this case contention results in internal loops within the cone (therefore the implementation is not convergent). Fortunately, [GS95] has, in addition to the General Equality Theorem, a version which is extended by notions of progression and fairness to counteract the problem of implementations with internal loops (this is Theorem A.2). Fairness allows that we abstract from *progressing* internal actions only. This abstraction is obtained by the application of a *pre-abstraction* function. We will use a focus condition and matching criteria relative to this pre-abstraction (Definitions A.4 and A.6).

A requirement of the cones and foci proof method is that the process be defined by a linear equation (Definition A.1). The linearisation of process terms is a common transformation in process algeba. The linearisation technique of [Gro96] provides rules for the transformation in the special case that the system is composed of similar processes (as in *ImpA* and *ImpB*).

As mentioned earlier, the protocol 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 *GoodTopology* is formalised below.

Definition 3.1 Given n:$\mathbb{N}$, the maximal node identifier in the network, and a list P_0:$\mathbb{N}$*SetList* giving a set of neighbours for all nodes in the network, the conjunction of the following properties is called $GoodTopology(n, P_0)$:

- P_0 is symmetric: $\forall i, j.(i \in P_0[j] \Leftrightarrow j \in P_0[i])$.
- P_0 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. $\forall k, j \leq n. \exists s = i_0 \ldots i_m.(i_0 = k \land i_m = j)$
  - no loops : $\forall i. \neg\exists$ direct path $s = i_0 i_1 \ldots i_m.(i = i_0 \land i = i_m)$.

As a preliminary step to applying this proof method for either Implementation A or Implementation B, the process *Spec* defined in Section2.1 must be translated into linear form. Additionally, a data parameter must be added on which to base a mapping from the data of the *ImpA* or *ImpB*.

Definition 3.2 (*Linear Specification*)

$$L\text{-}Spec(b{:}\mathbb{B}) = leader \cdot L\text{-}Spec(\mathsf{f}) \lhd b \rhd \delta$$

Clearly $L\text{-}Spec(\mathsf{t}) = Spec$.

4. Correctness of Implementation A

4.1. Linearisation

The linearisation of *ImpA* is given here as *L-ImpA*. For recursive calls of *L-ImpA* only those arguments which are updated are given, e.g. $L\text{-}ImpA(1/S[k])$ means replace the kth element of S by 1, leaving all other elements as they are.

Definition 4.1 (*Linearearisation of ImpA*)

$$\begin{aligned}
&L\text{-}ImpA(n{:}\mathbb{N}, P{:}\mathbb{N}SetList, S{:}\mathbb{N}List) = \\
&\quad \textstyle\sum_{k:\mathbb{N}} \ leader \cdot L\text{-}ImpA(1/S[k]) \\
&\qquad \lhd S[k] = 0 \land empty(P[k]) \land k \leq n \rhd \delta + \\
&\quad \textstyle\sum_{k,j:\mathbb{N}} c(j, k, par) \cdot L\text{-}ImpA((P[k] \setminus \{j\})/P[k], 1/S[j]) \\
&\qquad \lhd S[j] = 0 \land P[j] = \{k\} \land S[k] = 0 \land j \in P[k] \land \\
&\qquad\quad k \neq j \land k, j \leq n \rhd \delta
\end{aligned}$$

This linearisation can be derived straightforwardly from the definition of individual nodes using the linearisation technique of [Gro96]. We assert

$$ImpA(n, P_0) = L\text{-}ImpA(n, P_0, S_0),$$

where S_0 is the list of initial state values for the nodes, so $\forall i. S_0[i] = 0$.

[1] Define *paths*, $s{:}\mathbb{N}List = i_0 i_1 \ldots i_m$ such that $\forall k < m. i_{k+1} \in P_0[i_k]$. *Direct paths* are paths which do not backtrack down an edge already followed (remember P_0 is symmetric). A path s is direct if $\forall k < m.(i_k i_{k+1} \in s) \Rightarrow (i_{k+1} i_k \notin s)$.

4.2. Invariants

The proof of correctness also requires a number of invariants. The invariants listed below hold in every state (n, P, S) that can be reached from the initial state (n, P_0, S_0). The variables k and l are universally quantified over $\{0, \ldots, n\}$.

$\mathrm{I}_1: \ S[k] = 0 \vee S[k] = 1$
$\mathrm{I}_2: \ l \in P_0[k] \leftrightarrow l \in P[k] \vee k \in P[l]$
$\mathrm{I}_3: \ l \in P_0[k] \wedge l \notin P[k] \rightarrow S[l] = 1$
$\mathrm{I}_4: \ S[k] = 1 \rightarrow \mathit{singleton}(P[k]) \vee \mathit{empty}(P[k])$
$\mathrm{I}_5: \ l \in P[k] \wedge S[k] = 0 \rightarrow S[l] = 0 \wedge k \in P[l]$

The proofs of these are straightforward, and omitted here.

4.3. Some intermediate steps

The linearisation *L-ImpA* is not sufficient to allow us to apply Theorem A.1. The indices of the sums preceding any visible actions must be the same in both the specification and the implementation; clearly this is not the case. The sum over k preceding the *leader* action in *L-ImpA* correctly reflects that any node can be the root, i.e. there are multiple foci. However, it is not important *which* node is the root, only that one is chosen, and the boolean condition guarding the *leader* action in *L-ImpA* ensures that this is the case, summed up in Lemma 4.1. This lemma says that if a node can do the *leader* action, then all other nodes are in state 1. So if a node declares itself leader then it is the first one to do so, and because after this action all states will be in state 1, there will be no *leader* action, or any other action, after it.

Lemma 4.1 (*Uniqueness of Root*)

$$\forall k \leq n. \ \mathit{empty}(P[k]) \rightarrow \forall l \leq n. \ l \neq k \rightarrow S[l] = 1$$

Proof. We assume nodes $k, l \leq n$ such that $k \neq l \wedge \mathit{empty}(P[k]) \wedge S[l] = 0$, and derive a contradiction. By *GoodTopology* there is a path of distinct nodes $k = k_0 \ldots k_m = l$, such that $\forall i < m. \ k_{i+1} \in P_0[k_i]$. By I_2 and $\mathit{empty}(P[k_0])$ we see that $k_0 \in P[k_1]$. Then by I_3 $S[k_1] = 1$, and by I_4 $\mathit{singleton}(P[k_1])$. In a similar way we derive for all $0 < i \leq m$ that $P[k_i] = \{k_{i-1}\}$ and $S[k_i] = 1$. So in particular $S[l] = 1$. $\square$

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 *ImpA*. Introduce the function *pr* on data states of the implementation, which is the minimal node identifier of nodes in state 0, and if there is no node in state 0 it is defined to be 0. The variables k are quantified over $\{0, \ldots, n\}$.

$$\mathit{pr}(n, P, S) = \mathit{if}(\exists k. \ S[k] = 0, \mathit{min}(\{k \mid S[k] = 0\}), 0)$$

By Lemma 4.1, only one node will perform the *leader* action. We see that if a node k can perform the *leader* action, i.e. if it satisfies $S[k] = 0 \wedge \mathit{empty}(P[k])$, then it will be the value of *pr*. So it is safe to eliminate the summation over k in the first summand of *L-ImpA*, by instantiating it with $\mathit{pr}(n, P, S)$. This elimination yields the redefinition of process *L-ImpA* defined below. We often write *pr* to denote the value of *pr* in the current state.

Definition 4.2 (*Linearisation of ImpA (redefined)*)

$$\begin{aligned}
&\textit{L-ImpA}(n{:}\mathbb{N}, P{:}\mathbb{N}\textit{SetList}, S{:}\mathbb{N}\textit{List}) = \\
&\quad \textit{leader} \cdot \textit{L-ImpA}(1/S[pr]) \lhd S[pr = 0 \wedge \textit{empty}(P[pr]) \rhd \delta + \\
&\quad \textstyle\sum_{k,j:\mathbb{N}} c(j,k,par) \cdot \textit{L-ImpA}((P[k] \setminus \{j\})/P[k], 1/S[j]) \\
&\qquad\quad \lhd S[j] = 0 \wedge P[j] = \{k\} \wedge S[k] = 0 \wedge j \in P[k] \wedge \\
&\qquad\qquad k \neq j \wedge k,j \leq n \rhd \delta
\end{aligned}$$

4.4. Verification

The theorem to be demonstrated can now be stated as:

Theorem 4.1 Under the assumption of *GoodTopology* and the invariants it holds that

$$\tau \cdot \textit{L-Spec}(\mathsf{t}) = \tau \cdot \tau_{\{c\}} \textit{L-ImpA}(n, P_0, S_0).$$

In the special case where $n = 0$ (there is only one node in the network) we have

$$\textit{L-Spec}(\mathsf{t}) = \tau_{\{c\}} \textit{L-ImpA}(n, P_0, S_0).$$

This is a direct instantiation of Theorem A.1 with the initial state, because in the initial state the focus condition (defined below) is true if and only if $n = 0$. In order to prove Theorem 4.1 the matching criteria must be satisfied. To show that the matching criteria hold we first define the focus condition and the state mapping for $\tau_{\{c\}} \textit{L-ImpA}$. 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:

$$FC = \forall k,l \leq n.\ S[k] = 1 \vee P[k] \neq \{l\} \vee S[l] = 1 \vee k \notin P[l] \vee\ k = l.$$

The state mapping h is a function mapping data states of the implementation into data states of the simple specification. In this case h is defined so that it is t before the visible *leader* action occurs and f afterwards:

$$h(n, P, S) = (S[pr] < 1).$$

Intuitively h says that as long as the possible root, pr, introduced in the last section, has not moved to state 1 then the *leader* action has not yet occurred.

The matching criteria Given the particulars of *L-ImpA*, *L-Spec*, FC and h, the matching criteria are mechanically derived from the general forms of Definition A.3. The instantiated matching criteria are stated below, together with the proof that they hold.

1. The implementation is convergent.
   Using the number of nodes k for which $S[k] = 0$ as a measure, then each τ step decreases that measure by one.
2. In any state $d = (n, P, S)$ of the implementation, the execution of an internal step leads to a state with the same h-image.
   Suppose an internal action is possible, i.e. there are nodes $k, l \leq n$ such that

$$S[k] = 0 \wedge P[k] = \{l\} \wedge S[l] = 0 \wedge k \in P[l] \wedge\ k \neq l$$

We see that $S[pr] = 0$. We have to show that if we reach a state $d' = (n, P', S')$ by the communication between nodes k and l, then $S'[pr'] = 0$, where pr' is the value of pr in state d'. It holds that $S = S'$ except that $S'[k] = 1$. By definition of pr, $pr' \neq k$ because there is at least one node, i.e. l, with a state value equal to 0.

3. $S[pr] < 1 \wedge \mathit{empty}(P[pr]) \rightarrow S[pr] < 1$. Trivial.
4. $S[pr] < 1 \wedge FC \rightarrow S[pr] < 1 \wedge \mathit{empty}(P[pr])$
   Assume $S[pr] < 1 \wedge FC$. Then trivially $S[pr] < 1$. We prove $\mathit{empty}(P[pr])$ by assuming $\neg \mathit{empty}(P[pr])$ and deriving a contradiction. Let $k_1 \in P[pr]$. By I_5 we have $S[k_1] = 0$ and $pr \in P[k_1]$. By FC we see that $\neg \mathit{singleton}(P[k_1])$, so there is a $k_2 \neq pr$ in $P[k_1]$ such that $S[k_2] = 0$ and $k_1 \in P[k_2]$. We see that proceeding in this way we can construct an infinite path $k_0 k_1 k_2 \ldots$, where $pr = k_0$, such that for all i it holds that $S[k_i] = 0$, $k_i \in P[k_{i+1}]$ and $k_i \neq k_{i+2}$. By I_2 we see that this contradicts *GoodTopology*.
5. Trivial. The action *leader* involves no data.
6. If the implementation can reach state d' by the execution of the *leader* action, then $h(d') = \mathsf{f}$.
   Assume $S[pr] = 0 \wedge \mathit{empty}(P[pr])$ (the *leader* action can be executed).
   Then by Lemma 4.1 we see that all nodes other than pr are in state 1. We also see that the execution of the *leader* action brings also the node that is the value of pr in state 1. So after the action all nodes are in state 1, so then the value of h will be f.

By Theorem A.1 it follows that Theorem 4.1 holds.

5. Correctness of Implementation B

In Figure 3 we give a new definition for individual nodes *NodeB*. The definition in Figure 2 is easier to read, but we will use this definition because it is more compact. Using $s > 0 \rightarrow \mathit{empty}(p) \vee \mathit{singleton}(p)$ and $s = 3 \rightarrow \mathit{empty}(c)$, that hold in every state reachable from the initial state, it is easy to check that these definitions are equivalent (cf. I_4 and I_8 of Section 5.2).

5.1. Linearisation

The linearisations of the processes *NodesB* and *Buffers* are defined in Figures 4 and 5 as N and *L-Buffers* respectively. We left out the linearisation of the process *Buffer*. Individual buffers are modelled by the identifiers of their source and target nodes, a natural 0 or 1 giving the *state* of the buffer – where 0 means the buffer is empty and 1 means the buffer is full, and a message value of type $\mathbb{M}$. The parameters BS and BM in the definition of *L-Buffers* are tables containing entries for pairs of naturals: for all naturals i and j, $BS[i,j]$ of type $\mathbb{N}$ is the state value of the buffer from i to j and $BM[i,j]$ of type $\mathbb{M}$ is the message value of the buffer from i to j. Let the initial values of the parameters be such that $\mathit{GoodTopology}(n, P_0)$ and

$$\forall i,j.\ C_0[i] = \emptyset \wedge S_0[i] = 0 \wedge BS_0[i,j] = 0 \wedge BM_0[i,j] = \mathit{ack}.$$

We took the initial message values to be acknowledgements for convenience; this is not essential.

$$
\begin{array}{l}
\mathit{NodeB}(i{:}\mathbb{N}, p{:}\mathbb{N}\mathit{Set}, c{:}\mathbb{N}\mathit{Set}, s{:}\mathbb{N}) = \\
\quad \mathit{leader} \cdot \mathit{NodeB}(i,p,c,4) \lhd (s=0 \lor s=2) \land \mathit{empty}(p) \rhd \delta + \\
\quad \sum_{j:\mathbb{N}} r(j,i,\mathit{par}) \cdot \mathit{NodeB}(i, \mathit{if}(s=2,p,p \setminus \{j\}), \mathit{if}(s=2,c,c \cup \{j\}), \\
\qquad\qquad\qquad\qquad\qquad \mathit{if}(s=2,3,\mathit{if}(\mathit{singleton}(p),1,0))) \\
\qquad \lhd (s=0 \lor s=2 \lor s=3) \land j \in p \rhd \delta + \\
\quad \sum_{j:\mathbb{N}} r(j,i,\mathit{ack}) \cdot \mathit{NodeB}(i,p,c,4) \lhd s=2 \land p=\{j\} \rhd \delta + \\
\quad \sum_{j:\mathbb{N}} \bar{s}(i,j,\mathit{par}) \cdot \mathit{NodeB}(i,p,c,2) \\
\qquad \lhd (s=0 \lor s=1 \lor s=3) \land p=\{j\} \land \mathit{empty}(c) \rhd \delta + \\
\quad \sum_{j:\mathbb{N}} \bar{s}(i,j,\mathit{ack}) \cdot \mathit{NodeB}(i,p,c \setminus \{j\}, \mathit{if}(\mathit{empty}(p) \land \mathit{singleton}(c),2,1)) \\
\qquad \lhd ((s=0 \land \mathit{singleton}(p)) \lor s=1) \land j \in c \rhd \delta
\end{array}
$$

Fig. 3. New definition of process *NodeB*

The implementation *ImpB* is given by

$$\mathit{ImpB}(n{:}\mathbb{N}, P_0{:}\mathbb{N}\mathit{SetList}) = \partial_H(N(n,P_0,C_0,S_0) \parallel \mathit{L\text{-}Buffers}(n, \mathit{BS}_0, \mathit{BM}_0)).$$

Linearisation of *ImpB* is the process *L-ImpB* defined in Figure 6.

$$
\begin{array}{l}
N(n{:}\mathbb{N}, P{:}\mathbb{N}\mathit{SetList}, C{:}\mathbb{N}\mathit{SetList}, S{:}\mathbb{N}\mathit{List}) = \\
\quad \sum_{k:\mathbb{N}} \mathit{leader} \cdot N(4/S[k]) \\
\qquad \lhd (S[k]=0 \lor S[k]=2) \land \mathit{empty}(P[k]) \land k \le n \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} r(j,k,\mathit{par}) \cdot N(\mathit{if}(S[k]=2, P[k], P[k] \setminus \{j\})/P[k], \\
\qquad\qquad\qquad\qquad\qquad \mathit{if}(S[k]=2, C[k], C[k] \cup \{j\})/C[k], \\
\qquad\qquad\qquad\qquad\qquad \mathit{if}(S[k]=2,3,\mathit{if}(\mathit{singleton}(P[k]),1,0))/S[k]) \\
\qquad \lhd (S[k]=0 \lor S[k]=2 \lor S[k]=3) \\
\qquad\quad \land j \in P[k] \land k,j \le n \land k \ne j \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} r(j,k,\mathit{ack}) \cdot N(4/S[k]) \\
\qquad \lhd S[k]=2 \land P[k]=\{j\} \land k,j \le n \land k \ne j \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} \bar{s}(k,j,\mathit{par}) \cdot N(2/S[k]) \\
\qquad \lhd (S[k]=0 \lor S[k]=1 \lor S[k]=3) \land P[k]=\{j\} \land \\
\qquad\quad \mathit{empty}(C[k]) \land k,j \le n \land k \ne j \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} \bar{s}(k,j,\mathit{ack}) \cdot N((C[k] \setminus \{j\})/C[k], \\
\qquad\qquad\qquad\qquad\qquad \mathit{if}(\mathit{empty}(P[k]) \land \mathit{singleton}(C[k]),2,1)/S[k]) \\
\qquad \lhd ((S[k]=0 \land \mathit{singleton}(P[k])) \lor S[k]=1) \land j \in C[k] \land \\
\qquad\quad k,j \le n \land k \ne j \rhd \delta
\end{array}
$$

Fig. 4. The linearisation of *NodesB*

$$
\begin{array}{l}
\mathit{L\text{-}Buffers}(n{:}\mathbb{N}, BS{:}\mathbb{N}\mathit{Table}, BM{:}\mathbb{M}\mathit{Table}) = \\
\quad \sum_{i,j:\mathbb{N}} \sum_{m:\mathbb{M}} s(i,j,m) \cdot \mathit{L\text{-}Buffers}(m/BM[i,j], 1/BS[i,j]) \\
\qquad \lhd\, BS[i,j] = 0 \wedge i,j \leq n \,\rhd\, \delta + \\
\quad \sum_{i,j:\mathbb{N}} \bar{r}(i,j,BM[i,j]) \cdot \mathit{L\text{-}Buffers}(0/BS[i,j]) \lhd BS[i,j] = 1 \wedge i,j \leq n \rhd \delta
\end{array}
$$

Fig. 5. The linearisation of *Buffers*

$$
\begin{array}{l}
\mathit{L\text{-}ImpB}(n{:}\mathbb{N}, P{:}\mathbb{N}\mathit{SetList}, C{:}\mathbb{N}\mathit{SetList}, S{:}\mathbb{N}\mathit{List}, BS{:}\mathbb{N}\mathit{Table}, BM{:}\mathbb{M}\mathit{Table}) = \\
\quad \sum_{k:\mathbb{N}} \mathit{leader} \cdot \mathit{L\text{-}ImpB}(4/S[k]) \\
\qquad \lhd\, (S[k] = 0 \vee S[k] = 2) \wedge \mathit{empty}(P[k]) \wedge k \leq n \,\rhd\, \delta + \\
\quad \sum_{k,j:\mathbb{N}} r^*(j,k,\mathit{par}) \cdot \mathit{L\text{-}ImpB}(\mathit{if}(S[k] = 2, P[k], P[k] \setminus \{j\})/P[k], \\
\qquad\qquad\qquad \mathit{if}(S[k] = 2, C[k], C[k] \cup \{j\})/C[k], \\
\qquad\qquad\qquad \mathit{if}(S[k] = 2, 3, \mathit{if}(\mathit{singleton}(P[k]), 1, 0))/S[k], \\
\qquad\qquad\qquad 0/BS[j,k]) \\
\qquad \lhd\, (S[k] = 0 \vee S[k] = 2 \vee S[k] = 3) \wedge j \in P[k] \wedge k,j \leq n \wedge k \neq j \wedge \\
\qquad\quad BS[j,k] = 1 \wedge BM[j,k] = \mathit{par} \,\rhd\, \delta + \\
\quad \sum_{k,j:\mathbb{N}} r^*(j,k,\mathit{ack}) \cdot \mathit{L\text{-}ImpB}(4/S[k], 0/BS[j,k]) \\
\qquad \lhd\, S[k] = 2 \wedge P[k] = \{j\} \wedge k,j \leq n \wedge k \neq j \wedge \\
\qquad\quad BS[j,k] = 1 \wedge BM[j,k] = \mathit{ack} \,\rhd\, \delta + \\
\quad \sum_{k,j:\mathbb{N}} s^*(k,j,\mathit{par}) \cdot \mathit{L\text{-}ImpB}(2/S[k], 1/BS[k,j], \mathit{par}/BM[k,j]) \\
\qquad \lhd\, (S[k] = 0 \vee S[k] = 1 \vee S[k] = 3) \wedge P[k] = \{j\} \wedge \mathit{empty}(C[k]) \wedge \\
\qquad\quad k,j \leq n \wedge k \neq j \wedge BS[k,j] = 0 \,\rhd\, \delta + \\
\quad \sum_{k,j:\mathbb{N}} s^*(k,j,\mathit{ack}) \cdot \mathit{L\text{-}ImpB}((C[k] \setminus \{j\})/C[k], \\
\qquad\qquad\qquad \mathit{if}(\mathit{empty}(P[k]) \wedge \mathit{singleton}(C[k]), 2, 1)/S[k], \\
\qquad\qquad\qquad 1/BS[k,j], \mathit{ack}/BM[k,j]) \\
\qquad \lhd\, ((S[k] = 0 \wedge \mathit{singleton}(P[k])) \vee S[k] = 1) \wedge j \in C[k] \wedge \\
\qquad\quad k,j \leq n \wedge k \neq j \wedge BS[k,j] = 0 \,\rhd\, \delta
\end{array}
$$

Fig. 6. The linearisation of *ImpB*

5.2. Invariants

The invariants given below hold in every state (n, P, C, S, BS, BM) that is reachable from the initial state. The variables k and l are universally quantified over $\{0, \ldots, n\}$.

I_1 : $S[k] \leq 4$

I_2 : $l \in P_0[k] \leftrightarrow l \in P[k] \vee k \in P[l]$

I_3 : $S[k] = 0 \wedge \mathit{empty}(P[k]) \rightarrow \mathit{empty}(P_0[k])$

I_4 : $S[k] > 0 \rightarrow \mathit{empty}(P[k]) \vee \mathit{singleton}(P[k])$

I_5 : $S[k] = 0 \wedge l \in P[k] \rightarrow (BS[l,k] = 0 \leftrightarrow BM[l,k] = \mathit{ack})$

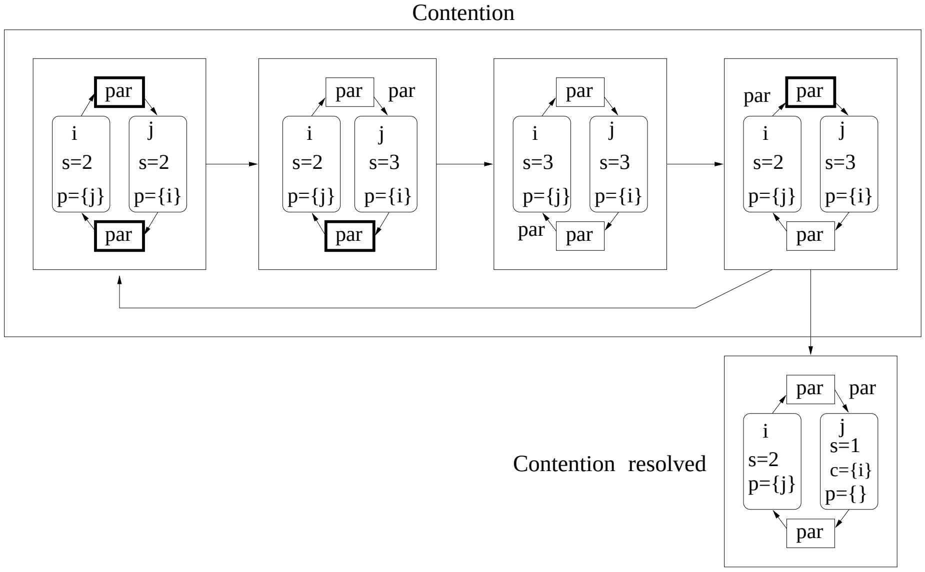

Fig. 7. Contention illustrated

$$
\begin{array}{lll}
\mathrm{I}_6 & : & S[k] \leq 1 \wedge (l \in P[k] \vee l \in C[k]) \rightarrow BS[k,l] = 0 \wedge BM[k,l] = ack \\
\mathrm{I}_7 & : & S[k] = 1 \rightarrow \neg(empty(P[k]) \wedge empty(C[k])) \\
\mathrm{I}_8 & : & S[k] = 3 \rightarrow empty(C[k]) \wedge singleton(P[k]) \\
\mathrm{I}_9 & : & S[k] = 3 \wedge P[k] = \{l\} \rightarrow BM[l,k] = par \\
\mathrm{I}_{10} & : & S[k] = 3 \wedge P[k] = \{l\} \rightarrow P[l] = \{k\} \wedge (S[l] = 2 \vee S[l] = 3) \\
\mathrm{I}_{11} & : & S[k] > 0 \wedge l \in P_0[k] \rightarrow P[k] = \{l\} \vee (S[l] > 0 \wedge P[l] = \{k\}) \\
\mathrm{I}_{12} & : & S[k] = 4 \wedge P[k] = \{l\} \rightarrow k \notin P[l] \\
\mathrm{I}_{13} & : & S[k] = 0 \wedge l \in P[k] \rightarrow k \in P[l] \wedge \\
 & & \quad (S[l] = 0 \vee S[l] = 1 \vee (S[l] = 2 \wedge BS[l,k] = 1)) \\
\mathrm{I}_{14} & : & S[k] = 3 \wedge P[k] = \{l\} \wedge S[l] = 3 \rightarrow BS[k,l] = 0 \\
\mathrm{I}_{15} & : & S[k] = 2 \wedge S[l] = 2 \wedge P[k] = \{l\} \wedge P[l] = \{k\} \rightarrow BS[k,l] = 1 \\
\mathrm{I}_{16} & : & S[k] = 2 \wedge S[l] = 3 \wedge P[k] = \{l\} \wedge P[l] = \{k\} \rightarrow \\
 & & \quad (BS[l,k] = 0 \rightarrow BS[k,l] = 1)
\end{array}
$$

Most of these invariants are easy to check. The last three invariants relate to contention in the system, i.e. two nodes have each sent a parent request to each other. There are four distinct states associated with contention; these are illustrated in Figure 7. We hope the picture is self-explanatory. It shows nodes i and j, and the buffers between them. A thick box indicates a buffer is in state 1, i.e. it holds a message that is to be transmitted.

5.3. Some intermediate steps

Linearisation of *ImpB* yields an expression where the summand starting with the external *leader* action is preceded by a summation. We eliminate this summation in the same way as in Section 4. Here the function *pr* on data states of the implementation is defined as taking the minimum of the set

$$\mathit{if}(\exists k.\mathit{empty}(P[k]), \{k \mid \mathit{empty}(P[k])\}, \{k \mid \neg\exists l.S[l] < S[k]\}),$$

where variables k and l are quantified over $\{0, \ldots, n\}$. We again need a 'uniqueness of root' lemma. Lemma 5.1 says that if a node k can declare itself leader or has declared itself leader, then there cannot be another node that can do the *leader* action. We also see that this k will then be the value of the function *pr*. Given the function *pr*, the new linearisation of *ImpB* is as presented in Figure 8.

Lemma 5.1 (*Uniqueness of Root*)

$$\forall k \leq n.\ \mathit{empty}(P[k]) \rightarrow \neg\exists l \leq n.\ l \neq k \wedge \mathit{empty}(P[l])$$

Proof. By I_1 $S[k] \leq 4$. If $S[k] = 0 \wedge \mathit{empty}(P[k])$, then $\mathit{empty}(P_0[k])$ by I_3, and by *GoodTopology* there is only one node in the network, so the lemma trivially holds.

Now assume $S[k] > 0 \wedge \mathit{empty}(P[k])$. Take an $l \leq n$ such that $l \neq k$. By *GoodTopology*, there is a path of distinct nodes $k_0 k_1 \ldots k_m$ with $k = k_0$, $l = k_m$ and $\forall i < m.\ k_{i+1} \in P_0[k_i]$. By I_{11} we see that since $P[k_0] \neq \{k_1\}$, it holds that $S[k_1] > 0$ and $P[k_1] = \{k_0\}$. Also by I_{11} it now holds that $P[k_i] = \{k_{i-1}\}$ for all $0 < i \leq m$. So $\neg\mathit{empty}(P[l])$. □

Lemma 5.2 $S[pr] > 1 \wedge \neg\mathit{empty}(P[pr]) \rightarrow \mathit{contention}$
where *contention* abbreviates

$$\begin{gathered}\exists k, l \leq n.(S[k] = 2 \vee S[k] = 3) \wedge (S[l] = 2 \vee S[l] = 3) \wedge \\ P[k] = \{l\} \wedge P[l] = \{k\}.\end{gathered}$$

Proof. Suppose $S[pr] > 1$ and $\neg\mathit{empty}(P[pr])$. Since $\neg\mathit{empty}(P[pr])$, there are at least two nodes. By definition of *pr* all nodes k have $S[k] > 1$ and $\neg\mathit{empty}(P[k])$. Then by I_4 $\mathit{singleton}(P[k])$ for all nodes k. Now supposing there is no pair of nodes that have each other as potential parent leads to a contradiction: Take any node k_0. Construct a path $k_0 k_1 \ldots$ such that $\forall i.\ P[k_i] = \{k_{i+1}\}$. By assumption $\neg\exists i.\ P[k_{i+1}] = \{k_i\}$. Now *GoodTopology* and I_2 tell us $\forall i.k_i \notin \{k_0, \ldots, k_{i-1}\}$. So this path must visit infinitely many nodes. This contradicts *GoodTopology*.

So there is a pair of nodes k, l such that $S[k] > 1 \wedge S[l] > 1 \wedge P[k] = \{l\} \wedge P[l] = \{k\}$. By I_{12} we know that $S[k] \neq 4$ and $S[l] \neq 4$. The lemma follows by I_1. □

Corollary 5.3 $S[pr] = 4 \rightarrow \mathit{empty}(P[pr])$

Proof. Suppose $S[pr] = 4$ and $\neg\mathit{empty}(P[pr])$. By the definition of *pr* it holds that $S[k] = 4$ for all nodes k. So $\neg\mathit{contention}$, contradicting Lemma 5.2. □

5.4. Verification

The correctness of Implementation B is stated by the following theorem.

$$
\begin{array}{l}
\textit{L-ImpB}(n{:}\mathbb{N}, P{:}\mathbb{N}\textit{SetList}, C{:}\mathbb{N}\textit{SetList}, S{:}\mathbb{N}\textit{List}, LS{:}\mathbb{N}\textit{Table}, LM{:}\mathbb{M}\textit{Table}) = \\
\quad \textit{leader} \cdot \textit{L-ImpB}(4/S[pr]) \lhd (S[pr]=0 \vee S[pr]=2) \wedge \textit{empty}(P[pr]) \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} r^*(j,k,\textit{par}) \cdot \textit{L-ImpB}(\textit{if}(S[k]=2, P[k], P[k]\setminus\{j\})/P[k], \\
\qquad\qquad \textit{if}(S[k]=2, C[k], C[k]\cup\{j\})/C[k], \\
\qquad\qquad \textit{if}(S[k]=2, 3, \textit{if}(\textit{singleton}(P[k]),1,0))/S[k], \\
\qquad\qquad 0/LS[j,k]) \\
\qquad \lhd (S[k]=0 \vee S[k]=2 \vee S[k]=3) \wedge j \in P[k] \wedge \\
\qquad\quad k,j \le n \wedge k \ne j \wedge LS[j,k]=1 \wedge LM[j,k]=\textit{par} \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} r^*(j,k,\textit{ack}) \cdot \textit{L-ImpB}(4/S[k], 0/LS[j,k]) \\
\qquad \lhd S[k]=2 \wedge P[k]=\{j\} \wedge k,j \le n \wedge k \ne j \wedge \\
\qquad\quad LS[j,k]=1 \wedge LM[j,k]=\textit{ack} \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} s^*(k,j,\textit{par}) \cdot \textit{L-ImpB}(2/S[k], 1/LS[k,j], \textit{par}/LM[k,j]) \\
\qquad \lhd (S[k]=0 \vee S[k]=1 \vee S[k]=3) \wedge P[k]=\{j\} \wedge \textit{empty}(C[k]) \\
\qquad\quad \wedge\, k,j \le n \wedge k \ne j \wedge LS[k,j]=0 \rhd \delta + \\
\quad \sum_{k,j:\mathbb{N}} s^*(k,j,\textit{ack}) \cdot \textit{L-ImpB}((C[k]\setminus\{j\})/C[k], \\
\qquad\qquad \textit{if}(\textit{empty}(P[k]) \wedge \textit{singleton}(C[k]),2,1)/S[k], \\
\qquad\qquad 1/LS[k,j], \textit{ack}/LM[k,j]) \\
\qquad \lhd ((S[k]=0 \wedge \textit{singleton}(P[k])) \vee S[k]=1) \wedge j \in C[k] \wedge \\
\qquad\quad k,j \le n \wedge k \ne j \wedge LS[k,j]=0 \rhd \delta
\end{array}
$$

Fig. 8. The process *L-ImpB* (redefined)

Theorem 5.1 Under the assumption of *GoodTopology* and the invariants, it holds that

$$\tau \cdot \textit{L-Spec}(\mathsf{t}) = \tau \cdot \tau_{\{r^*,s^*\}} \textit{L-ImpB}(n, P_0, C_0, S_0, BS_0, BM_0).$$

We will prove this theorem by application of Theorem A.2 (taking $\textit{Int} = \{r^*, s^*\}$ and $\textit{Ext} = \{\textit{leader}\}$).

Pre-abstraction, State Mapping and Focus Condition As explained in Section 2.3, the process *ImpB* is not convergent. Theorem A.2 requires that we distinguish between progressing and non-progressing internal actions. We define a pre-abstraction function on actions and their data, that yields f on non-progressing internal actions only. In this case, non-progressing actions occur when two nodes that are in contention send each other a parent request. More precisely: if one of the nodes has sent a parent request, and has moved into state 2, then the sending of a parent request by the other node is non-progressing. The pre-abstraction function ξ is defined by

$$\xi(a) = \begin{cases} \neg(S[k]=3 \wedge S[j]=2) & \text{if } a = s^*(k,j,\textit{par}), \\ \mathsf{t} & \text{otherwise.} \end{cases}$$

We define a state mapping h from data states of the implementation to data states of the specification. As before, this mapping is only concerned with values of states:

$$h(n, P, C, S, BS, BM) = (S[pr] < 4).$$

In Implementation A the *leader* action was the last action to occur. In Implementation B it is possible that some nodes are still waiting for acknowledgements after the *leader* action has occurred.

The focus condition of *L-ImpB* relative to ξ is the conjunction of the negations of the conditions for performing a progressing internal action (cf. Definition A.4):

$$\begin{array}{l} FC_\xi = \forall k, l \leq n.\ k \neq l \rightarrow \\ \quad \neg((S[k] = 0 \vee S[k] = 2 \vee S[k] = 3) \wedge l \in P[k] \wedge \\ \qquad BS[l, k] = 1 \wedge BM[l, k] = par) \\ \quad \wedge \\ \quad \neg(S[k] = 2 \wedge P[k] = \{l\} \wedge BS[l, k] = 1 \wedge BM[l, k] = ack) \\ \quad \wedge \\ \quad \neg((S[k] = 0 \vee S[k] = 1 \vee S[k] = 3) \wedge P[k] = \{l\} \wedge empty(C[k]) \wedge \\ \qquad BS[k, l] = 0 \wedge \neg(S[k] = 3 \wedge S[l] = 2)) \\ \quad \wedge \\ \quad \neg(((S[k] = 0 \wedge singleton(P[k]) \vee S[k] = 1) \wedge l \in C[k] \wedge BS[k, l] = 0). \end{array}$$

Using Invariants 4–9 we can simplify this formula to

$$\begin{array}{l} FC_\xi = \forall k, l \leq n.\ k \neq l \rightarrow \\ \qquad (S[k] = 0 \rightarrow (l \in P[k] \rightarrow BS[l, k] = 0) \wedge \neg singleton(P[k])) \\ \quad \wedge\ \ S[k] \neq 1 \\ \quad \wedge\ (S[k] = 2 \wedge P[k] = \{l\} \rightarrow BS[l, k] = 0) \\ \quad \wedge\ (S[k] = 3 \wedge P[k] = \{l\} \rightarrow BS[l, k] = 0 \wedge (BS[k, l] = 1 \vee S[l] = 2)). \end{array}$$

Before we prove the matching criteria, we add the following lemma.

Lemma 5.4 *contention* $\rightarrow \neg FC_\xi$

Proof. Suppose *contention*. So there are nodes $k, l \leq n$ such that

$$(S[k] = 2 \vee S[k] = 3) \wedge (S[l] = 2 \vee S[l] = 3) \wedge P[k] = \{l\} \wedge P[l] = \{k\}.$$

If one of these nodes, say k' – call the other node l', is in state 2, then we distinguish cases

- $BS[l', k'] = 1$. This contradicts the third conjunct of FC_ξ.
- $BS[l', k'] = 0$. Now by I_{15} it must be the case that $S[l'] = 3$. By I_{16} we see that $BS[k', l'] = 1$. This contradicts the last conjunct of FC_ξ.

So both nodes are in state 3. Then by I_{14} it holds that $BS[k', l'] = BS[l', k'] = 0$. This contradicts the last conjunct of FC_ξ. □

The Matching Criteria We instantiate Definition A.6 with the processes *L-ImpB* and *L-Spec*, the state mapping h and the pre-abstraction ξ.

1. The process *L-ImpB* is convergent w.r.t. ξ.
   Let rq be $\sum_{k \leq n} |P[k]|$; ac be $\sum_{k \leq n} |C[k]|$; s_i be the number of nodes in state

i; and l_2 be the number of requests sent to nodes in state 2, but not received yet. In other words: the number of lines such that its state equals 1 and the receiving node is in state 2.
We define the following measure on data states:

$$\mathit{Measure} = \langle rq, ac, s_0, s_1, l_2, s_3, s_2 \rangle.$$

Let $\prec$ be the lexicographical ordering on $\mathbb{N}^7$. Now $\prec$ is a well-founded ordering on the data states of *L-ImpB* such that the measure decreases at every execution of a progressing internal step.

2. In any state d of the implementation, the execution of an internal step leads to a state with the same h-image.
Suppose $S[pr] < 4$. The only internal action that can change the state of a node k to 4, is the receiving of an acknowledgement by k, where $S[k] = 2$ and $\mathit{singleton}(P[k])$.
Suppose in the state d' reached by this action, k becomes the value of pr, then $S'[pr'] = 4 \wedge \mathit{singleton}(P'[pr'])$. This contradicts Corollary 5.3.
So in every state d' reachable by an internal action $S'[pr'] < 4$.
Suppose $S[pr] \not< 4$. By I_1 and Corollary 5.3 it holds that $\mathit{empty}(P[pr])$. Now we see by Lemma 5.1 that pr will keep the same value.

3. $(S[pr] = 0 \vee S[pr] = 2) \wedge \mathit{empty}(P[pr]) \rightarrow S[pr] < 4$. Trivial.

4. $FC_\xi \wedge S[pr] < 4 \rightarrow (S[pr] = 0 \vee S[pr] = 2) \wedge \mathit{empty}(P[pr])$.
Suppose FC_ξ and $S[pr] < 4$. $S[pr] \neq 1$ by FC_ξ. If $S[pr] = 3$, then we have by I_8 and I_{10} that *contention*, contradicting the assumption FC_ξ by Lemma 5.4. So $(S[pr] = 0 \vee S[pr] = 2)$. We have to show $\mathit{empty}(P[pr])$. We distinguish cases $S[pr] = 0$ and $S[pr] = 2$ and show that the assumption $\neg\mathit{empty}(P[pr])$ leads to a contradiction.

   - $S[pr] = 0$. Assume $\neg\mathit{empty}(P[pr])$. Let $pr = k_0$ and $k_1 \in P[k_0]$. By I_{13} we can make the following case distinction, where $S[k_1] \neq 1$ by FC_ξ:

   $$S[k_1] = 0 \quad \text{or} \quad S[k_1] = 2 \wedge BS[k_1, k_0] = 1$$

     In the second case $\neg FC_\xi$ because $\neg(k_1 \in P[k_0] \rightarrow BS[k_1, k_0] = 0)$ and $S[k_0] = 0$. Contradiction. In the first case we see by FC_ξ that $\neg\mathit{singleton}(P[k_1])$, so there is a $k_2 \neq k_0$ in $P[k_1]$. We can repeat the argument above for k_1 and k_2. But we cannot construct an infinite path $k_0 k_1 \ldots$ where $\forall i. S[k_i] = 0 \wedge k_{i+1} \in P[k_i] \wedge k_i \neq k_{i+2}$, as this would violate *GoodTopology* by I_2. So for some i we get $S[k_i] = 0$ and $\neg(k_{i+1} \in P[k_i] \rightarrow BS[k_{i+1}, k_i] = 0)$, contradicting FC_ξ as above.
   - $S[pr] = 2$. Suppose $\neg\mathit{empty}(P[pr])$. Then we find $\neg FC_\xi$ by Lemma 5.2 and Lemma 5.4. Contradiction.

5. Trivial, because the *leader* action takes no data.

6. If from a state d, state d' is reached by the execution of the *leader* action, then $h(d') = \mathsf{f}$.
We see by Lemma 5.1 that the value of pr will be the same for d and d'. It holds that $S = S'$ except that $S'[pr] = 4$. So $h(d') = 4 < 4$, which is false.

Now Theorem 5.1 follows by Theorem A.2.

6. 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 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 *ImpA*. 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 achievement is that our study 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 similar processes to describe the system, state-based descriptions, the use of the state parameter to define the mapping function, a simple boolean in the specification and an invariant on the topology of the network). 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. [GK95, 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.

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. The first author thanks the Programming Research Group at the University of Amsterdam, EXPRESS project partners for providing a pleasant working environment, and the EC HCM Fellowship scheme for funding her visit. The second author was supported by the Netherlands Organization for Scientific Research (NWO) under contract SION-2854/612-61-002.

References

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.

[DGRV97] M.C.A. Devillers, W.O.D. Griffioen, J.M.T Romijn, and F.W. Vaandrager. Verification of a leader election ppotocol — formal methods applied to IEEE 1394. Technical report, Computing Science Institute, University of Nijmegen, December 1997.

[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 LNCS, 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):237–440, 1997.

[GK95] 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 '94*, Workshops in Computing, pages 63–86. Springer-Verlag, 1995.

[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 μCRL. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes '94*, 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.

[GS95] J.F. Groote and J. Springintveld. Focus points and convergent process operators. Technical Report 142, University of Utrecht, Logic Group Preprint Series, 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.

[IEE96] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus. Std 1394-1995, August 1996.

[KS96] H.P. Korver and M.P.A. Sellink. On automating process algebra proofs. In V. Atalay et al, editor, *Proceedings of the 11-th International Symposium on Computer and Information Sciences, ISCIS XI Antalya, Turkey*, volume 2, 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.

A. Theorems and Definitions

We repeat here the most important definitions and theorems from [GS95]. For the formulation we rely in part on the appendix of [GS96].

A.1. General Equality Theorem

Definition A.1 Let $A \subseteq Act \cup \{\tau\}$ be a finite set of actions, and let D be a data type. A *linear process equation* (LPE) over Act and D is an equation of the form

$$X(d{:}D) = \sum_{a\in A} \sum_{e:E_a} a(f_a(d,e)) \cdot X(g_a(d,e)) \lhd b_a(d,e) \rhd \delta$$

for some data types E_a, D_a, and functions $f_a{:}D \to E_a \to D_a$, $g_a{:}D \to E_a \to D$, $b_a{:}D \to E_a \to \mathbb{B}$. (We assume that τ has no parameter.)

A summand $a(f_a(d,e)) \cdot X(g_a(d,e)) \lhd b_a(d,e) \rhd \delta$ means that if for some e of type E_a the guard $b_a(d,e)$ is satisfied, the action a can be performed with parameter $f_a(d,e)$, followed by a recursive call of X with new value $g_a(d,e)$. The main feature of LPEs is that for each action a there is at most one summand in the alternative composition. Note that therefore the definition of process *L-ImpB* in Figure 8 does not directly fit into this format. We made sure that theorems were applied correctly.

Definition A.2 An LPE X written as in Definition A.1 is called *convergent* if it does not admit infinite τ-paths, i.e. there is a well-founded ordering $<$ on D such that for all $e{:}E_\tau$ and $d{:}D$ we have that $b_\tau(d,e)$ implies $g_\tau(d,e) < d$.

An *invariant* of an LPE X written as in Definition A.1 is a function $I{:}D \to \mathbb{B}$ such that for all $a \in A$, $e{:}E_a$, and $d{:}D$ we have $b_a(d,e) \wedge I(d) \to I(g_a(d,e))$.

Definition A.3 Let X and Y be LPEs given as follows:

$$X(d{:}D_X) = \sum_{a\in A} \sum_{e:E_a} a(f_a(d,e)) \cdot X(g_a(d,e)) \lhd b_a(d,e) \rhd \delta$$

$$Y(d{:}D_Y) = \sum_{a\in A\setminus\{\tau\}} \sum_{e:E_a} a(f'_a(d,e)) \cdot Y(g'_a(d,e)) \lhd b'_a(d,e) \rhd \delta$$

Let FC_X be a formula over $d{:}D_X$ describing exactly the states of X from which no τ-action is enabled (i.e. equivalent to $\neg\exists e_\tau{:}E_\tau b_\tau(d,e_\tau)$). Let $h{:}D_X \to D_Y$ be a state mapping. The following 6 conditions are called the *matching criteria* and their conjunction is denoted by $C_{X,Y,h}(d)$.

1. X is convergent
2. $\forall e{:}E_\tau (b_\tau(d,e) \to h(d) = h(g_\tau(d,e)))$
3. $\forall a \in A \setminus \{\tau\}\ \forall e{:}E_a\ (b_a(d,e) \to b'_a(h(d),e))$
4. $\forall a \in A \setminus \{\tau\}\ \forall e{:}E_a\ (FC_X(d) \wedge b'_a(h(d),e) \to b_a(d,e))$
5. $\forall a \in A \setminus \{\tau\}\ \forall e{:}E_a\ (b_a(d,e) \to f_a(d,e) = f'_a(h(d),e))$
6. $\forall a \in A \setminus \{\tau\}\ \forall e{:}E_a\ (b_a(d,e) \to h(g_a(d,e)) = g'_a(h(d),e))$

Theorem A.1 (*General Equality Theorem)* Let X, Y, FC_X, and h be as above. Suppose I is an invariant of X and, for all $d{:}D_X$, $I(d) \to C_{X,Y,h}(d)$. Assume that r and q are solutions of X and Y, respectively, then

$$\forall d{:}D_X\ I(d) \to r(d) \lhd FC_X(d) \rhd \tau r(d) = q(h(d)) \lhd FC_X(d) \rhd \tau q(h(d)).$$

A.2. Abstraction and idle loops

Let X and Y be LPEs given as follows:

$$X(d{:}D_X) = \sum_{a \in Ext \cup Int \cup \{\tau\}} \sum_{e:E_a} a(f_a(d,e)) \cdot X(g_a(d,e)) \lhd b_a(d,e) \rhd \delta$$

$$Y(d{:}D_Y) = \sum_{a \in Ext} \sum_{e:E_a} a(f'_a(d,e)) \cdot Y(g'_a(d,e)) \lhd b'_a(d,e) \rhd \delta$$

where *Ext*, *Int* and $\{\tau\}$ are mutually disjoint.

Definition A.4 Let ξ be a pre-abstraction function. The *focus condition* of X relative to ξ is defined by:

$$FC_{X,Int,\xi}(d) = \forall a \in Int \cup \{\tau\} \forall e{:}E_a \neg(b_a(d,e) \wedge \xi(a)(d,e))$$

Definition A.5 X is *convergent w.r.t.* ξ iff there is a well-founded ordering $<$ on D_X such that for all $a \in Int \cup \{\tau\}, d{:}D_X$ and all $e{:}E_a$ we have that $b_a(d,e)$ and $\xi(a)(d,e)$ imply $g_a(d,e) < d$.

Definition A.6 Let X, Y be as above. Let $h{:}D_X \to D_Y$ and let ξ be a pre-abstraction function. The following 6 conditions are called the *matching criteria for idle loops* and their conjunction is denoted by $CI_{X,Y,\xi,h}(d)$.

1. X is convergent w.r.t. ξ
2. $\forall a \in Int \cup \{\tau\}\ \forall e{:}E_a\ (b_a(d,e) \to h(d) = h(g_a(d,e)))$
3. $\forall a \in Ext\ \forall e{:}E_a\ (b_a(d,e_a) \to b'_a(h(d),e))$
4. $\forall a \in Ext\ \forall e{:}E_a\ (FC_{X,Int,\xi} \wedge b'_a(h(d),e) \to b_a(d,e))$
5. $\forall a \in Ext\ \forall e{:}E_a\ (b_a(d,e) \to f_a(d,e) = f'_a(h(d),e))$
6. $\forall a \in Ext\ \forall e{:}E_a\ (b_a(d,e) \to h(g_a(d,e)) = g'_a(h(d),e))$

Theorem A.2 Let X, Y, ξ and h be as above. Let p and q be solutions of X and Y, respectively. If I is an invariant of X and $\forall d{:}D_X\ (I(d) \rightarrow CI_{X,Y,\xi,h}(d))$, then

$$\forall d{:}D_X\ I(d) \rightarrow \tau\tau_{Int}(p(d)) = \tau q(h(d)).$$