Parameterized Reachability Analysis of the IEEE 1394 Root Contention Protocol using TReX

A. Collomb-Annichini[1] and M. Sighireanu[2]

[1] VERIMAG, Centre Equation, 2 av. de Vignate, 38610 Gières, France
[2] LIAFA, University of Paris 7, 2 place Jussieu, 75251 Paris Cedex 5, France

Abstract. We report about the reachability analysis of fully parametrized models of the IEEE 1394 root contention protocol. This protocol uses timing constraints in order to elect a leader. The interesting point is that the timing constraints involve some parameters (transmission delay, bounds of waiting intervals), and the behavior of the protocol strongly depends on the relation between these parameters. In order to synthesize the relation ensuring the correct behavior of the protocol, we apply the symbolic reachability techniques implemented in the TReX tool. We take the unparameterized model of Root Contention protocol proposed in [23] and study different parametrized versions of this model. We are able to synthesize automatically all the relations already found by proof or experiments on the unparameterized versions. We compare our results with those reported or obtained using other tools for parametrized systems.

1 Introduction

Due to the progress done in the area of classical timed model checking, tools like Kronos [27] and Uppaal [19] are routinely used for the verification of industrial case studies. Now, important research effort is concentrated on the extension of this classical framework with other kind of infinite-domain variables (e.g., counters, queues) or with *parameters* (i.e., uninstantiated constants). In this way, the models verified are closer to the real implementations and it is possible to synthesize the constraints on parameters such that some properties are verified. Indeed, in the classical framework, the synthesis of constraints on parameters is done (e.g., [23]) by giving (manually) concrete values to parameters and then, extrapolate the results obtained. This manual process is very time consuming and error prone. Therefore, tool support for deriving the constraints *automatically* is very important. There are currently several tools that can do analysis of parametrized timed systems: HyTech [14], LPMC [16], TReX [6], and a prototype extension of Uppaal [15]. Now, these tools has to be confronted to realistic case studies in order to guide the research effort.

The IEEE 1394 root contention protocol [1] is a such case study. It is a real-time leader election protocol for two processes, used by the the physical level of the IEEE 1394 bus in order to determine the topology of the network. In this protocol, timing parameters like transmission delay and waiting intervals play an essential role. For some values of parameters, the protocol is correct and for others it fails either by electing two leaders or by infinite running. We are interested in synthesizing automatically the constraints that ensure the correctness of the protocol. This is

[3] This work was supported in part by the European Commission (FET project ADVANCE, contract No IST-1999-29082).

interesting since it has been shown by experiments [23] that the values for waiting intervals given in the standard [1] do not support the increase of the transmission delay, i.e., the length of the cable.

In order to synthesize the constraints, we use the TReX [6] tool on the different models of the protocol. TReX is a tool for automatic analysis of automata-based models equipped with variables belonging to different infinite/finite domains and with parameters. These models are, at the present time, timed automata extended with counters and parameters and communicating through unbounded lossy FIFO channels and shared variables. The techniques used in TReX are based on *symbolic reachability analysis*. Symbolic representation structures are used to represent infinite sets of configurations for the variables. Forward/backward exploration procedures are used to generate a finite abstraction of the input model representing the graph of reachable symbolic configurations. The termination is not guaranteed, but efficient *extrapolation techniques* [2, 4] are used to help it. These techniques are based on computing the (exact) effect of iterating control loops detected automatically during the exploration.

For timed automata, the symbolic representation structure used by TReX is an extension of classical DBM [10], Constrained Parametric DBM [4]. They allow to deal in a uniform way with counter/clock automata, parametric/non-parametric models, and systems generating linear or nonlinear arithmetical constraints between parameters.

Related work. The work on the verification of the IEEE 1394 standard is concentrated on the lower levels of the protocol: [17, 22] focus on the link layer, and [9, 21, 24, 26, 20, 23, 5, 15] study protocols used by the physical level. For this last level, except [23, 5, 15], the verification is done by manual proof, or using theorem proving tools like PVS. These proof approaches focus on the whole physical layer protocol and show its correction under several assumptions on the topology of the network. [20] provides a detailed overview of each approach.

The first attempt [23] to verify mechanically the root contention protocol uses Uppaal. By (manual) analysis of some scenarios of the protocol, they obtain two constraints on parameters: the first is necessary and sufficient to elect an unique leader (safety property), the second is sufficient for termination (liveness property). These constraints are shown to be correct by instantiating the parameters with values satisfying the constraints and then model-checking the models using Uppaal. We consider the same models for the protocol, but fully parameterized and slightly modified in order to easily verify the properties of the protocol. Although in [23] are not given execution performances, we think that due to the absence of parameters, their performances may be better than ours.

A first step to the automatic verification of the parametrized version of the protocol is made in [5]. They uses LPMC [16], an extension of the PMC model-checker for parameterized timed systems. LPMC uses partition refinement techniques and deals only with linear constraints between parameters. This seems to be sufficient for the root contention protocol. The model chosen is inspired by [26] and it is more complex than [23] because it models the probabilistic aspects of the protocol. However, they consider a wrong model for communications w.r.t. the IEEE 1394 standard [23]. In the experiments done with LPMC, not all the five timing constants of the protocol have been considered as parameters, but *only one or two* of them. For example, for the model with only one parameter, the verification of a safety property takes 60s and 45MB with LPMC. This result is comparable with the one we obtain for the good model (i.e., 3min and 15MB). However, we didn't try to apply LPMC on our model because of lack of time and expertise.

[15] is the first attempt to do fully parametrized model-checking of this protocol. They use the models given in [23] and apply a prototype extension of UPPAAL for linear parameterized model checking. This tool, although less general in the input model than TREX, uses the same symbolic representation structures for valuations of clocks, the Constrained Parametric DBM introduced in [4]. They limit the analysis to linear constraints, which is a restriction compared with TREX, but gives better performances in some cases. Also, they identify a subclass of parameterized timed automata (lower/upper bound) for which the symbolic reachability algorithm always terminates. The root contention protocol belongs to this class of automata. However, for more general classes of timed automata, they do not propose techniques to help termination. TREX is more general in this point, since it uses extrapolation techniques. For this case study, they model-check that, in the presence of the constraints proposed in [26, 23], the protocol satisfies the safety and liveness properties. We apply TREX on the same model and obtain greater values for execution time and memory consumption. We think that these bad performances of TREX are fully explained by the limitations of the input model of TREX (the rendez-vous must be modeled by shared variables) and by the use of a more general (non-linear) external constraint solving procedure. The positive point for TREX is that we are able to synthesize the safety constraints directly, without the use of additional lemma or the instantiation of parameters. Also, we are able to synthesize the constraints for the liveness properties and to model-check more complex properties.

To our knowledge, HYTECH [14] has not been used until now for the verification of the root contention protocol. HYTECH model-checks hybrid automata, a more general timed model than the one used by TREX and UPPAAL. It also uses the symbolic reachability analysis, but on different symbolic representations for variables, i.e. polyhedra. The constraints considered are linear. In this paper, we apply successfully HYTECH on the parametrized models provided in [23] and we compare the results (when possible) with TREX and PUPPAAL.

Overview of the paper. We begin by giving a short presentation of the TREX tool, where we explain mainly the techniques used by this particular case study. Then, Section 3 describes informally the IEEE 1394 root contention protocol and gives its specification. Section 4 presents five models considered for analysis with TREX and reports, for each model, the experimental results. Also, we compare, when possible, our results with those reported in [15] or obtained with HYTECH. Finally, we give our concluding remarks.

2 Symbolic Reachability Analysis with TReX

Figure 1 shows the overall environment and architecture of TREX.

The input model of TREX is given by a textual IF [7] file. This file can be obtained by translation (using IF tools) of a graphical SDL specification, or of a textual LOTOS one. Also, the IF file can be used (after instantiation of parameters) to do finite model-checking with the CADP toolbox [12] or KRONOS. The input model is the only mandatory input of TREX. Additionally, the user can specify initial constraints on the parameters occurring in the model. These constraints play the role of an invariant: they are true in each reachable configuration explored by TREX. The reachability analysis can begin either from the initial state of the model or from a symbolic initial configuration (i.e., control state and symbolic valuation of variables) specified by the user.

TREX can check on-the-fly safety properties on the transitions of the input model. The property should be given as an *observer* written in IF. The observer is an extended automaton sharing

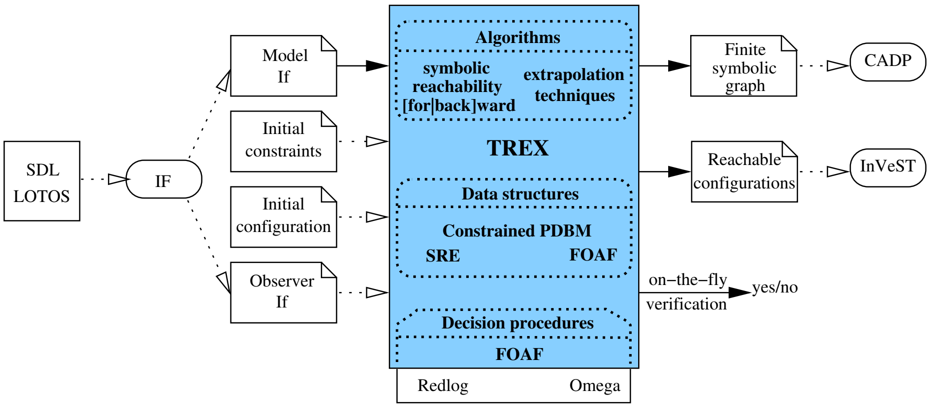

Fig. 1. Overview of the TREX architecture and environment.

transition labels with the input model. TREX does on-the-fly the product of the observer with the input model and synchronizes transitions having the same labels. If the synchronization is not possible, it means that the input model does not satisfy the property. Moreover, the sequence of transitions from the initial state of the input model to the state where the synchronization fails represents a diagnosis (counterexample). Indeed, the symbolic configuration where the synchronization fails gives a necessary (super-approximation) set of constraints ϕ for which the property is not satisfied. This set can be reduced by verifying the same property with an initial constraint including the negation of a subset of ϕ. The minimal set of constraints needed to satisfy the property can be obtained by iterating this procedure. We use intensively this technique for the synthesis of constraints during the verification of safety properties for the root contention protocol.

Also, TREX can generate the (finite) graph of reachable configurations. It is a finite abstraction of the analyzed model, which can be used for finite-state model checking.

The core of TREX are a forward/backward exploration algorithm and the symbolic representation structures for infinite sets of valuations for variables. The exploration algorithm is generic and can be used for any kind of symbolic representation providing a data structure, basic operations ($\cup$, $\cap$, $\subseteq$, ...), symbolic successor/predecessor operations, and an extrapolation procedure.

In the current version, TREX provides three packages for symbolic representation of configurations: SRE (Simple Regular Expressions) [3] package for lossy FIFO channels, Constrained Parametric DBMs [4] for parameterized counter and timed automata, and FOAF (First Order Arithmetical Formulas) for linear and nonlinear constraints on parameters. Since the models used for this case study are parametric timed automata, we present shortly the last two data structures above.

Parametric DBMs (PDBMs) are extension of DBMs [10], which are used (e.g., in KRONOS) in order to represent clock valuation sets (zones). A PDBM is a square matrix, which elements are pairs of the form (comparison operator, arithmetical term on parameters). In Constrained PDBMs, the PDBM is associated with a first order formula on the parameters occurring in the matrix. All operations on DBMs (emptiness, intersection, and inclusion tests) can be adapted for Constrained PDBMs [4]. These operations use a canonical form for Constrained PDBM, which is computed by a *symbolic* Floyd-Warshall algorithm. Since this algorithm is very time consuming, TReX does the hashing of all new computations. In practice, this optimization gives good results, reducing by 2 to 10 times the number of computations of the canonical form.

The FOAF package provides a compact representation (including simplifications according to the arithmetical rules) for terms and formulas used in Constrained PDBMs. It provides methods to give for each term/formula its kind (linear or non-linear). This is particularly useful to know which decision procedure can be applied for the test of satisfiability of formulas. Moreover, FOAF provides the Fourier-Motzkin procedure [11] for elimination of quantifiers over real variables. In order to test the satisfaction of formulas, TReX uses also external decision procedures like: Omega [18] for linear constraints on integers, and Reduce [13] for linear and non-linear constraints on reals.

TReX has been used to analyze several nontrivial protocols in their parametric versions, such as the Bounded Retransmission Protocol (BRP) [8]. This particular example requires the full power of TReX since it is a parametric heterogeneous model involving clocks, counters, and lossy channels. Moreover, the constraints manipulated in this model are nonlinear (contain products between variables). As far as we know, TReX is the only existing tool which allows to deal fully automatically with such a complex model.

3 Root Contention Protocol

The IEEE's Microcomputer Standards Committee started to work in 1986 on the unification of several serial buses such as Vme, Multibus II, and Future Bus. The goal was to provide easy to use, low cost, high speed communications. This effort led to a new serial bus protocol defined in the IEEE 1394 Standard [1], which supports transfer rates of 100, 200, and 400Mps. Although the standard has been finalized, the architecture is still being refined and adapted to provide additional data rates of 800, 1,600, and 3,200Mbps. Part of this ongoing work is reflected in the proposals IEEE 1394a [25] and 1394b.

The IEEE 1394 standard specifies a serial bus that supports peer-to-peer, isochronous and asynchronous data transfers among up to 64 devices. Also, the bus is hot-plug-and-play since its reconfiguration occurs automatically whenever a new device is added. The IEEE 1394 architecture involves *nodes* (addressable entities that run their own part of the protocol) connected by serial *cables* or backplane. On each node, the protocol consists of four layers: the physical layer, the link layer, the transaction layer, and the serial bus management layer. We restrict our attention to the protocol of the physical layer executed on nodes linked by cables.

The physical layer includes electrical signaling, mechanical interface, data transmission, *bus configuration* and arbitration. At this level, each node has several ports which may be connected by cables to other nodes. The configuration task of the physical level consists of tacking the relatively flat physical topology of the bus and turning it into a logical tree structure with a root node. Configuration consists of three phases: (1) the bus initialization, (2) the tree identification, where it is checked whether the bus topology is a tree and, if so, a root is elected, and (3) the self identification where each node selects a unique physical identifier and sends it to other nodes.

The root contention protocol occurs at the end of the second phase. In this second phase, each node tries to assign to its connected ports a parent or a child flag. Leaf nodes (i.e., nodes having only one connected port) start by outputting a `Parent_Notify` signal to their nearest neighbor node (with certain propagation delay). When a node receives a `Parent_Notify` signal on one of its ports, it marks that port as bound to a child and outputs a `Child_Notify` signal on the same port. Upon detecting this state, the leaf node marks this port as a parent port and removes the signaling, thereby confirming that the leaf node has accepted the child designation. A node with several connected ports waits until only one port remains unmarked and then it signals `Parent_Notify` on this port. The port on which it receives a `Child_Notify` is marked as parent. The node with all its connected ports marked child is chosen as the root of the tree. In the final stage of the tree identification phase, it is possible that two neighboring nodes have marked all but one port as being child and signal `Parent_Notify` to each other. In this case, the *root contention protocol* is initiated in order to elect one of the two nodes as the root of the tree.

If a node receives a `Parent_Notify` signal on a port, while sending a `Parent_Notify` signal on that port, it knows, locally, that it is in root contention. Then, it removes the `Parent_Notify` signal and leaves the line in the `Idle` state. At the same time, it starts a timer and picks a random bit. If the random bit is one, the node will wait for a long time (`Root_Cont_Slow`), else it will wait for a short time (`Root_Cont_Fast`).

When the waiting timer expires, the node samples its contention port. If it sees `Idle`, it starts sending `Parent_Notify` anew and looks to see a `Child_Notify` signal in return for acknowledgement. When the timer expires and the node sees `Parent_Notify` on its port, it sends a `Child_Notify` signal back as acknowledgement. Now, it knows that it has to take the root role. However, the root contention may occur again if the nodes pick the same random bits (at the same moment) or they pick different random bits within a time equal with the difference between `Root_Cont_Slow` and `Root_Cont_Fast`. In this case, both nodes detect renewed root contention and the whole process is repeated until one of them becomes root.

[1] specifies that $1590\ ns \leq$ `Root_Cont_Slow` $\leq 1670\ ns$ and $760\ ns \leq$ `Root_Cont_Fast` $\leq 850\ ns$ for a cable length of maximum 4.5 m (so the communication delay is also a bounded parameter). Since the ongoing work on the standard converges to changes of the transfer speed and cable length, it is interesting to have the relation between the parameters of the protocol such that its properties are satisfied.

The following three properties are required for this protocol:

- *Safety1*: A unique root is elected.
- *Safety2*: In a round of the protocol, a root is elected only on the request made during this round. The rounds are delimited by `Idle` signals.
- *Liveness*: The protocol terminates under the assumption that the probability to pick different bits is equal to one.

The first and third property are natural. The second property is stated in informal notes to the IEEE 1394a Working Group (see [23] for more details). It says that the the decision of a node to become root have to be made on the basis of the `Parent_Notify` signal sent after the detection of the contention, and not on an old signal. This means that the `Idle` state signaled by a node after the detection of the root contention have to arrive to the node's pair before the expiration of its waiting interval.

In the remainder of the paper, we present the models we consider as input of TReX and the results obtained for the verification of the properties above.

4 Models and Experimental Results

The concrete architecture of all models considered consists in two (symmetrical) automata modeling the contention nodes and two automata implementing the cable, one wire for each direction (see Figure 2).

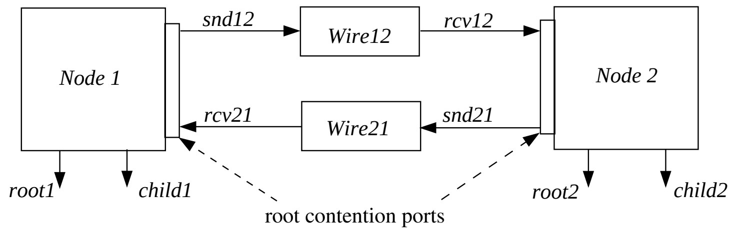

Fig. 2. General architecture of the implementation models..

Depending on the semantics chosen for the communication between the nodes, the models proposed for the root contention protocol fall into two classes. The first class [26] considers that the communication is done by message passing, the messages being `Parent_Notify` and `Child_Notify`. Also, the messages can be overwritten and lost. It is pointed out in [23] that this abstraction is inappropriate, since IEEE 1394 standard specifies that, at this phase, the communication is done by signaling states, not by transmitting packets. Indeed, the whole tree identification process occurs in a matter of microseconds. The second class of models [23] implements the communications by signals persisting at the input port of the receiving node, until the sending node changes its output port signal. So, besides the `Parent_Notify` and `Child_Notify` signals, the wire can be left undriven (`Idle` signal). We take this last model for communications.

All models are parametrized by five parameters: the maximum delay for communication and signal processing (*delay*), the minimum and maximum values for the long waiting interval (*rc_slow_min* resp. *rc_slow_max*), and the minimum and maximum values for the short waiting interval (*rc_fast_min* resp. *rc_fast_max*). The initial constraint on these parameters, ϕ_0, says that all the timers, including the communication delay cannot be 0, and that the values of the faster interval of waiting are less than the values of the slower interval:

$$\phi_0 \equiv \mathit{delay} > 0 \wedge 0 < \mathit{rc\_fast\_min} \leq \mathit{rc\_fast\_max} < \mathit{rc\_slow\_min} \leq \mathit{rc\_slow\_max}$$

In the remainder of this paper, we use the following shorter notations for the signals : `req` for `Parent_Notify` and `ack` for `Child_Notify`. All experiments have been performed on a Pentium III at 597MHz and 256MB of memory.

4.1 Model I1: synchronous detection of the initial root contention

This model is inspired by the one proposed in [23] (see Figure 3). There are four timed, input/output automata with urgent states (marked by U). We suppose that the reader is familiar with the model of timed automata. Input/output automata have transitions labelled by a tuple build from: the name of the port (e.g., `snd` and `rcv`), the operation on this port (`?` for input, `!` for output), and the signals sent (e.g., `req`, `idle` or empty). The output operation is not blocking, but the input operation is blocking until the input is available. We reduce the number of control states of the automata proposed in [23] by using enumerated variables in order to stock signals received. Also, we use guarded input operations in the generic wire automaton with the following semantics: the input is taken only if the guard (boolean expression between square brackets) is satisfied.

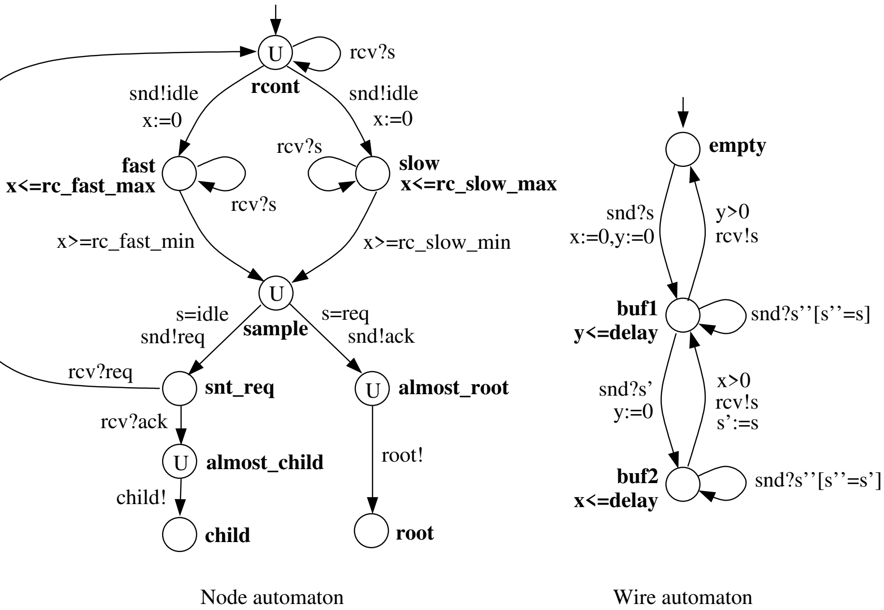

Fig. 3. Generic node and wire automata for Model I1.

The generic automata of Figure 3 are substituted with the signals specific to each node. The whole system is then described by the following CCS-like expression:

Node1 [*root1* /root,*child1* /child,*snd12* /snd,*rcv21* /rcv]
|| *Wire12* [*snd12* /snd,*rcv12* /rcv] || *Wire21* [*snd21* /snd,*rcv21* /rcv] ||
Node2 [*root2* /root,*child2* /child,*snd21* /snd,*rcv12* /rcv]

The model supposes that initially, the nodes detect simultaneously the root contention. However, this is not true for the subsequent rounds. The modeling of wires as two place buffers is correct if we suppose that the wires do not get more inputs than the nodes can handles, i.e., than the receiving part can distinguish. So, until a new signal arrives, the old one continue to be driven across the wire. From experimental results, [23] shows that the modeling of wires as two place buffers is correct if $delay < rc\_fast\_min$, which is the case in IEEE 1394.

Since the input model of TReX allows only communications by shared variables and lossy FIFO channels, we modeled the input/output communications by shared variables. This gives us a model having four processes, five parameters, six clocks, and six enumerated variables. The same model in HyTech input format have four processes, five parameters, and six clocks.

Verification of Safety1: The observer used to verify the property *Safety1* is given on Figure 4. First, we apply TReX on the full parameterized version of the model, with ϕ_0 as initial constraint.

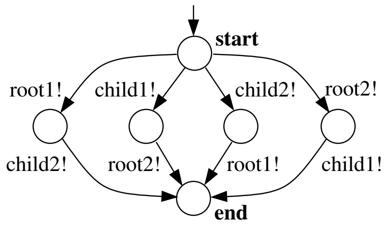

Fig. 4. Observer for property *Safety1*.

We obtain that the property is not satisfied. The performances are reported in Table 1. Note that HyTech fails to give this diagnosis by forward reachability because of lack of memory. This is due to the fact that, in forward analysis, HyTech does not check the property on-the-fly, but tries to build the reachability graph. Using backward reachability, HyTech succeeds to give a diagnosis, but not to synthesize a symbolic constraint like TReX.

The symbolic configuration where synchronization fails include the following set of constraints between parameters:

$$\begin{array}{l}
\phi_0 \;\wedge\; \overbrace{\mathit{rc_fast_min} < \mathit{delay}}^{\tilde{\phi_1}} \wedge \overbrace{\mathit{rc_fast_min} < 2 * \mathit{delay}}^{\tilde{\phi'_1}} \wedge \overbrace{\mathit{rc_slow_min} - \mathit{rc_fast_max} < 2 * \mathit{delay}}^{\tilde{\phi_2}} \;\wedge \\
\mathit{rc_slow_min} - \mathit{rc_fast_max} + 2 * \mathit{rc_fast_min} < 2 * \mathit{delay} \;\wedge \\
\mathit{rc_slow_max} - \mathit{rc_fast_max} < \mathit{delay} \;\wedge\; 2 * \mathit{rc_fast_min} < \mathit{delay} \;\wedge \\
\mathit{rc_slow_min} - \mathit{rc_fast_max} < \mathit{delay} \;\wedge\; \mathit{rc_slow_min} - \mathit{rc_fast_max} + \mathit{rc_fast_min} < \mathit{delay} \;\wedge \\
\mathit{delay} < \mathit{rc_slow_max} \;\wedge\; \mathit{delay} < \mathit{rc_fast_max} \;\wedge\; \mathit{delay} < \mathit{rc_slow_min} - \mathit{rc_fast_min} \;\wedge \\
2 * \mathit{delay} < \mathit{rc_slow_min} \;\wedge\; 2 * \mathit{delay} < \mathit{rc_fast_max} \;\wedge\; \mathit{rc_fast_max} + \mathit{rc_fast_min} < \mathit{rc_slow_min}
\end{array}$$

The constraints $\tilde{\phi_1}$, $\tilde{\phi'_1}$, and $\tilde{\phi_2}$, are interesting since their strict negation has been found by experiments in [23] to be sufficient conditions to satisfy properties *Safety1* and *Safety2*.

Then, we introduce the strict part of the negation of each constraint above in the initial constraint and we model-check the safety property. It results that only the constraint $\phi_1 \equiv \mathit{delay} < \mathit{rc\_fast\_min}$ is needed to satisfy this property. This result has been obtained only with HyTech (see Table 1). The partial graph obtained with TReX satisfies the property.

Since the parameterized extension of Uppaal [15] is not yet available, we only put in Table 1 the results reported in [15] (and obtained on a similar configuration). For this, we checked the property for some instantiations of parameters. The case we consider is where each guard or invariant for *delay* is strict, both *rc_fast_max* and *rc_slow_max* are infinite, and both *delay* and *rc_slow_min* are equal to *rc_fast_min*. We have so a model with only one parameter. The safety property is verified. The results we obtain are compared with those reported by [15] in Table 1. The difference of performances in this case is in our disfavor. This can be explained by the more complex model we considered (synchronizations done by shared variables) and the use of the heavy decision procedures for non-linear constraints of REDUCE.

Table 1. Experimental results for Model I1 and property *Safety1*.

model	*property*	*no. param.*	*constr.*	*result*	TReX		HyTech		*P*Uppaal	
I1	*Safety1*	5	ϕ_0	false	3:09:53	45MB	0:27:46	out of mem.	not given	
		5	$\phi_0 \wedge \phi_1$	true	> 67h	> 192MB	0:00:06	5MB	not given	
		1	$rc\_fast\_min > 0$	true	0:03:09	15MB	0:00:01	5MB	0:00:08	11MB

Verification of Safety2: This property cannot be easily verified on this model, since for the initial round no request is sent. In order to verify it, we will introduce the Model I2 (see section 4.2).

Verification of Liveness1: This cannot be done with this model, since we cannot model the probability to choose the random bits. This property will be considered for the Model I3 (see section 4.3).

4.2 Model I2: asynchronous detection of the initial root contention

We relax the hypothesis of Model I1 concerning the initial detection of the root contention, since it cannot be satisfied in a distributed system. In fact, before detection of contention, each node sends initially a *req* signal on the (unique) port which is not yet marked as child port, and then waits for an acknowledgement or a request. Suppose that *Node1* (resp. *Node2*) sends *req* at the (absolute) time t_1 (resp. t_2). Then, $-delta \leq t_1 - t_2 \leq delta$. Indeed, if $t_1 - t_2 > delta$, *Node1* cannot send a *req* because it has already received a *req*, the delay of communications being bound by *delta*.

Using this observation, we modify the generic automaton of nodes for Model I1 as follows. The **rcont** state is no more initial, and we add a new initial state named **init**. A node may wait in the **init** state no more than *delta* (i.e., the invariant of state **init** is $\mathbf{x} \leq \mathbf{delta}$), then it sends a request signal and moves to **snt_req** state. This model is particularly useful for the verification of the property *Safety2*.

Verification of Safety2: The observer used to verify the property *Safety2* for the Node 2 is given on Figure 5. Like for the first property, we apply TReX on the full parameterized version of

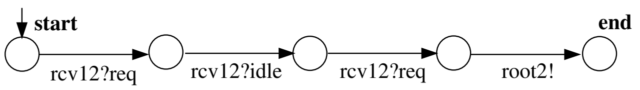

Fig. 5. Observer for property *Safety2* and the Node 2.

the model, with ϕ_0 as initial constraint. We obtain that the property is not satisfied. The symbolic configuration where synchronization fails include the following set of constraints between parameters:

$$\phi_0 \ \wedge \ \overbrace{rc\_fast\_min < delay}^{\tilde{\phi_1}} \ \wedge \ delay < rc\_fast\_max$$

This corresponds to the constraint synthesized for the property *Safety1*. If we add to ϕ_0 the strict negation of the constraint $\tilde{\phi_1}$, ϕ_1, we obtain that the property is false, and the following constraint is synthesized:

$$\phi_0 \ \wedge \ \overbrace{delay < rc\_fast\_min}^{\phi_1} \wedge \overbrace{rc\_fast\_min < 2 * delay}^{\tilde{\phi_1'}} \ \wedge \ 2 * delay < rc\_fast\_max$$

The strict negation of $\tilde{\phi_1'}$ has been found manually in [23] to be a sufficient condition to satisfy property *Safety2*. Then, we introduce the strict part of the negation of this constraint in the initial constraint and we model-check the safety property.

It follows that the property is not satisfied, and we obtain the constraint $\tilde{\phi_2} \equiv rc\_slow\_min - rc\_fast\_max < 2 * delay$ and a diagnosis showing that the observer fails because the renewal of the root contention. This is a liveness problem, and we show for Model I3 that $\phi_2 \equiv 2 * delay < rc\_slow\_min - rc\_fast\_max$ is the condition to solve in one pass the root contention if different bits are picked.

The experimental results are in Table 2. For PUppaal, no results are reported in [15], for this particular case.

Table 2. Experimental results for Model I2 and property Safety2.

model	*property*	*no. param.*	*constr.*	*result*	TReX		HyTech	
I2	*Safety2*	5	ϕ_0	false	0:00:50	14MB	0:00:50	29MB
		5	$\phi_0 \wedge \phi_1$	false	0:05:57	9MB	0:00:08	5MB
		5	$\phi_0 \wedge \phi_1'$	false	0:01:41	17MB	0:00:08	8MB
		5	$\phi_0 \wedge \phi_1' \wedge \phi_2$	false	0:03:13	20MB	0:00:07	4MB

4.3 Model I3: centralized, boolean fairness for Model I1

The models above do not include the hypothesis that the probability to pick different bits is equal to one. Since the input model of TReX does not include probabilistic aspects, we simplify this hypothesis by adding to the Model I1 a centralized generator of bits which always gives different values for any two successive calls. Therefore, it is essential that root contention is resolved within the same round if both nodes pick different random bits. Since the probability to pick different random bits is strictly greater than zero in each pass, the nodes will eventually pick different bits and thus elect a root, with probability one.

We add to the Model I1 a boolean variable shared by the two nodes, which implement the fair generator. Also, we add labels `retry1!` resp. `retry2!` to the transitions from the state `snt_req` to the state `rcont` in order to observe the renewal of the protocol. We use the same observer as for the property *Safety1*, but the set of observed labels is now updated with the `retry1!` and `retry2!`. This means that the observer does not accept synchronization with any retry transition.

By applying TReX on the full parameterized version of the model, with ϕ_0 as initial constraint, we obtain that the property is not satisfied. The set of constraints synthesized is:

$$\begin{array}{l}\phi_0 \ \wedge \ \overbrace{rc\_fast\_min < delay}^{\tilde{\phi_1}} \wedge \overbrace{rc\_fast\_min < 2 * delay}^{\tilde{\phi_1'}} \ \wedge \\ rc\_slow\_max < 2 * delay \ \wedge \ rc\_slow\_max - rc\_fast\_min < delay \ \wedge \ delay < rc\_fast\_max\end{array}$$

The constraints synthesized are those needed for the satisfaction of the safety properties. So, we repeat the experiment by tacking $\phi_0 \wedge \phi_1$ resp. $\phi_0 \wedge \phi_1'$ as initial constraints. In both cases, we obtain that the property is not satisfied. The following constraint appears in the fail state of both counterexamples:

$$\tilde{\phi_2} \equiv rc\_slow\_min - rc\_fast\_max < 2 * delay$$

By tacking now the strict negation of this constraint, we obtain that the property is satisfied. This has been also proved by experiments in [23]. The performances are given in Table 3.

Table 3. Experimental results for Model I3.

model	*property*	*no. param.*	*constr.*	*result*	TReX		HyTech	
I3	*Liveness*	5	ϕ_0	false	1:05:58	28MB	0:00:04	3MB
		5	$\phi_0 \wedge \phi_1$	false	0:02:54	9MB	0:00:03	2MB
		5	$\phi_0 \wedge \phi_1'$	false	0:02:51	9MB	0:00:03	2MB
		5	$\phi_0 \wedge \phi_1' \wedge \phi_2$	true	0:02:09	8MB	0:00:01	4MB

4.4 Model A1: abstraction of communications

The timed, input/output automaton A1 of Figure 6 taken from [26, 23] abstracts the communications between the nodes and records the status of each of the two nodes. They prove that if the parameters satisfy the additional constraint:

$$\overbrace{2 * delay < rc\_fast\_min}^{\phi_1'} \ \wedge \ \overbrace{2 * delay < rc\_slow\_min - rc\_fast\_max}^{\phi_2}$$

this abstraction is correct, in the sense that all execution paths of I1 are included in A1. For this model, we compute the symbolic graph using TReX and HyTech. The two symbolic graphs obtained are equivalent. The performances are given in Table 4. Then, we consider A1 as being an observer for the Model I1 in the presence of the initial constraint $\phi_0 \wedge \phi_1' \wedge \phi_2$. This property is verified, so we are able to verify mechanically the results given in [26, 23].

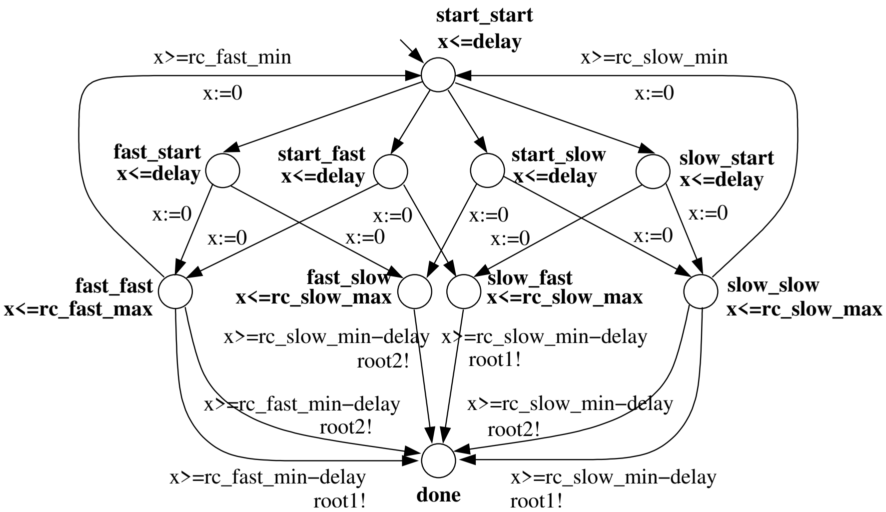

Fig. 6. Model A1: abstraction of communications.

Table 4. Experimental results for Model A1.

model	*property*	*no. param.*	*constr.*	*result*	TReX		HyTech	
A1	true	5	ϕ_0	false	0:00:02	0.1MB	0:00:01	0.1MB
I1	A1	5	$\phi_0 \wedge \phi_1' \wedge \phi_2$	false	7:24:30	190MB	0:31:17	out of mem.

5 Conclusion

This paper reports about the fully automatic verification of the IEEE 1394 root contention protocol using TReX and HyTech tools. This is an industrial protocol in which timing parameters play an essential role.

We consider the models proposed in [23] and improve them in order to verify easily the safety and liveness properties of the protocol. This verification allows us to *synthesize automatically* the constraints derived by experimental results in [23]. Although in some cases the performances of TReX are not so good as those obtained with the prototype extension of Uppaal [15], we are able to deal with more complex properties and to directly synthesize the constraints. Compared with HyTech, TReX is slower, but it provides symbolic diagnosis while a safety property is checked. This is very useful for constraint synthesis.

The relatively bad timing performances obtained motivate us to connect TReX to a tool for linear constraints solving and to improve its input model with communication by synchronization on actions.

References

1. *IEEE Standard for a High Performance Serial Bus.* Number Std 1394-1995. IEEE, August 1996.

2. P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proceedings of 5th TACAS*, volume 1579 of *LNCS*. Springer Verlag, 1999.
3. P.A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy, FIFO channels. In *Proceedings of the 10th CAV*, volume 1427 of *LNCS*, pages 305–317. Springer Verlag, 1998.
4. A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855 of *LNCS*, pages 419–434. Springer Verlag, July 2000.
5. G. Bandini, R. Lutje Spelberg, R. de Rooij, and W. Toetenel. Parametric verification of the ieee-1394a root contention protocol using lpmc. Submitted for publication. http://tvs.twi.tudelft.nl/, July 2000.
6. A. Bouajjani, A. Collomb-Annichini, and M. Sighireanu. Trex: A tool for reachability analysis of complex systems. In *Proceedings of CAV'01*, LNCS. Springer Verlag, 2001.
7. M. Bozga, J.-C. Fernandez, L. Girvu, S. Graf, J.-P. Krimm, and L. Mounier. If: A validation environment for times asynchronous systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855 of *LNCS*, pages 543–547. Springer Verlag, July 2000.
8. P.R. D'Argenio, J.-P. Katoen, T.C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In *Proceedings of 3rd Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1217 of *LNCS*, pages 416–432. Springer Verlag, 1997.
9. M. Devilliers, W. Griffioen, J. Romijn, and F. Vaandrager. Verification of a leader election protocol — formal methods applied to IEEE 1394. Technical Report CSI-R9728, Computing Science Institute, University of Nijmegen, December 1997.
10. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proceedings of the 1st CAV*, volume 407 of *LNCS*, pages 197–212. Springer Verlag, 1989.
11. B.C. Eaves and U.G. Rothblum. Dines-fourier-motzkin quantifier elimination and an application of corresponding transfer principles over ordered fields. *Mathematical Programming*, 53:307–321, 1992.
12. J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. Cadp (cæsar/aldebaran development package): A protocol validation and verification toolbox. In R. Alur and T.A. Henzinger, editors, *Proceedings of the 8th CAV*, volume 1102 of *LNCS*, pages 437–440. Springer Verlag, August 1996.
13. A.C. Hearn. *REDUCE — User's and Contributed Packages Manual*. Codemist Ltd., February 1999. version 3.7.
14. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):110–122, 1997.
15. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. inear parametric model checking of timed automata. In *Proceedings of TACAS'01*, 2001.
16. R.F. Lutje Spelberg, W.J. Toetenel, and M. Ammerlaan. Partion refinement in real-time model checking. In A.P. Ravn and H. Rischel, editors, *Proceedings of 5th FTRTFT*, volume 1486 of *LNCS*, pages 143–157. Springer Verlag, 1998.
17. S. Luttik. Description and formal specification of the link layer of ieee-p1394. In *Proceedings of FICS'97*, pages 43–56, 1997. Also available as Report SEN-R9706, CWI, Amsterdam.
18. Omega Team. *The Omega Library*, November 1996. version 1.1.0.
19. P. Pettersson and K.G. Larsen. UPPAAL2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, February 2000.
20. J. Romijn. A timed verification of the ieee 1394 leader election protocol. In *Proceedings of FMICS'99*, pages 2–29, May 1999. Appeared in Formal Methods in System Design 2000.
21. C. Shankland and M. van der Zwaag. The tree identify protocol of IEEE 1394 in μ-CRL. Technical Report SEN-R9831, CWI, Amsterdam, November 1998.
22. M. Sighireanu and R. Mateescu. Verification of the link layer protocol of the ieee-1394 serial bus ("firewire"): an experiment with e-lotos. *Software Tools for Technology Transfer*, 2(1):68–88, 1998.

23. D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the ieee 1394a root contention protocol using uppaal2k. Technical Report CSI-R0009, Computing Science Institute Nijmegen, May 2000. Conditionally accepted to STTT.
24. Normed simulations. W. griffioen and f. vaandrager. In *Proceedings of CAV'98*, volume 1427 of *LNCS*, pages 332–334. Springer Verlag, 1998.
25. IEEE Computer Society. Draft standard for a high performance serial bus (supplement). Technical Report P1394a Draft 2.0, IEEE, March 1998.
26. M. Stoelinga and F. Vaandrager. Root contention in IEEE 1394. In *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems, ARTS'99*, volume 1601. Springer Verlag, 1999.
27. S. Yovine. Kronos: A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer*, 1(1/2), October 1997.