CertiCAN: A Tool for the Coq Certification of CAN Analysis Results

Pascal Fradet, Xiaojie Guo, Jean-François Monin, Sophie Quinton

CertiCAN: A Tool for the Coq Certification of CAN Analysis Results

Pascal Fradet[1], Xiaojie Guo[1,2], Jean-François Monin[2] and Sophie Quinton[1]

[1]*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble France*
[2]*Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, F-38000 Grenoble France*

***Abstract*—This paper introduces CertiCAN, a tool produced using the Coq proof assistant for the formal certification of CAN analysis results. Result certification is a process that is light-weight and flexible compared to tool certification, which makes it a practical choice for industrial purposes.**

The analysis underlying CertiCAN, which is based on a combined use of two well-known CAN analysis techniques, is computationally efficient. Experiments demonstrate that CertiCAN is faster than the corresponding certified combined analysis. More importantly, it is able to certify the results of RTaW-Pegase, an industrial CAN analysis tool, even for large systems. This result paves the way for a broader acceptance of formal tools for the certification of real-time systems analysis results.

I. Introduction

A. Motivation

There is a general trend toward certified[1] proofs for real-time systems analysis. One reason for this is the need to increase our confidence in the analysis techniques developed by the research community. A recent series of mistakes in the analysis of self-suspending tasks [9] underlines the limitations of pen-and-paper proofs for such complex problems. This issue is not new, as illustrated by the flaw in the original Response Time Analysis (RTA) of CAN messages proposed by Tindell *et al.* [26], [28], [27], which was found and fixed only many years later [12]. This motivated the development of Prosa [11], an open-source library of definitions and proofs for real-time systems analysis based on the Coq [4] proof assistant. Computer assisted proofs provide the additional advantage that they make it easier to build on top of existing results and to precisely identify the hypotheses required for a result to hold.

A second reason behind the need to certify real-time systems analysis results comes from industry. Standards such as ISO 26262 for automotive or DO-178C for avionics advocate the use of formal methods for the development and validation of safety critical systems. In particular, proof assistants have now reached a level of maturity where they are used for industrial applications – see, for example, the use of the CompCert C compiler [3], [18] based on Coq or the Sel4 microkernel [8] based on Isabelle/HOL [6]. It is only natural that such a general trend towards formal proofs also affects real-time aspects. For all these reasons, we aim at providing certified real-time guarantees for industrial systems.

As a first example of a (not so) simple analysis with industrial relevance and following our Work-in-Progress paper [16], we focus in this work on the CAN analysis cited earlier. The underlying analysis is a RTA for task sets with offsets under Fixed Priority Non Preemptive (FPNP) scheduling, with a notion of transaction, i.e., messages sent from the same electronic control unit (ECU). The CAN [10] protocol is widely used in automotive applications and there exist several commercial tools performing CAN analysis. Among these, we focus on RTaW-Pegase [7], for which we obtained an academic license.

B. Certifying the results of a CAN analysis tool

Rather than certifying RTaW-Pegase, that is, formally proving that the CAN analysis implemented in RTaW-Pegase is correct, we choose to build a tool based on the Coq proof assistant that can certify the results of the CAN analysis performed by RTaW-Pegase. In other words, our tool, called CertiCAN, can be called every time a result obtained with RTaW-Pegase [2] must be certified. This choice is motivated by the fact that result certification is a process that is light-weight and flexible compared to tool certification, which makes it a practical choice for industrial purposes. Indeed, RTaW-Pegase is a complex tool for which we do not have the source code. It is likely to be highly optimized and subject to regular changes. All this would make it difficult to certify the tool directly and this correctness proof would need to be updated regularly.

Our problem is then: Can we certify efficiently enough the analysis results computed by RTaW-Pegase? Compared to a traditional RTA, can we use the fact that a result certifier is given as input the bound it is expected to certify?

Our solution is based on the following idea: We use a combination of two existing analysis techniques, one precise but with high computational complexity and another that is much faster

[1]Throughout this paper, the term *certified* means *formally verified* using a proof assistant, in our case Coq.

[2]Note that CertiCAN does not depend on the internals of the RTA tool considered. It is interoperable with any other tool analyzing the same CAN system models as RTaW-Pegase.

but approximate (it may compute pessimistic upper bounds on response times). These two analyses were introduced by Tindell for the RTA of tasks with offsets dispatched according to the Fixed Priority Preemptive policy [24], [25]. The precise analysis was adapted to CAN by Meumeu Yomsi *et al.* [29].

Our tool first tries to certify the result provided as input using the approximate analysis only, then resorts to the exact analysis for cases which cannot be certified using the approximate analysis. Experiments demonstrate that CertiCAN is faster than the corresponding certified combined analysis tool. More importantly, it is able to certify the results of RTaW-Pegase even for fairly large systems.

C. Contribution

The main contribution of this paper is CertiCAN, the first formally proven tool able to certify the results of commercial CAN analysis tools. This is however not the only contribution of the paper. More specifically, we propose:

1) A new RTA for CAN that combines two well-known analyses, one precise and another approximate;
2) The correctness proof in Coq of the three analyses;
3) Three Coq-certified tools in OCaml extracted from the proofs, one for each analysis;
4) Based on the same principle as the new RTA, a method and its corresponding tool formally verified in Coq; The tool, called CertiCAN, is entitled to certify the results of non certified tools such as RTaW-Pegase.
5) Experiments that show the differences in efficiency between the various techniques, and which demonstrate the usability of CertiCAN for industrial practice.

Beyond CertiCAN, we believe that the results presented in this paper are significant in that they demonstrate the advantage of result certification over tool certification for the RTA of CAN buses. In addition, the underlying technique can be reused for any other system model for which there exist RTAs with different levels of precision.

All the Coq specifications and proofs are available online [2].

D. Paper structure

The rest of this paper is laid out as follows. Section II introduces the system model and some notations and definitions used later on. We present in Section III the two existing CAN analyses on which our certifier is based. Sections IV and V describe the main contributions of the paper: first, an optimized RTA combining the previous CAN analyses, then CertiCAN, a tool based on Coq for certifying CAN analysis results, and finally their experimental evaluation. Additional details about the proofs and the generality of the approach are provided in Section VI. Related work is presented in Section VII and we conclude in Section VIII.

II. Context

The analyses of CAN networks proposed by RTaW-Pegase are based on a precise RTA of periodic tasks with offsets dispatched according to the FPNP scheduling policy [29]. In addition to the precise analysis, RTaW-Pegase proposes an approximate but faster version. The implementation of these analyses uses several undocumented optimizations.

In this section, we present the system model considered in these analyses as well as notations and definitions used throughout this article.

A. System model

The system model considered consists of a set of transactions representing ECU nodes

$$S := \{Tr_1, Tr_2, \ldots, Tr_N\}$$

where each transaction Tr_i is a set of periodic tasks (representing messages):

$$Tr_i := \{\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,M}\}$$

Each task $\tau_{i,k}$ has a fixed and unique priority k (a smaller number means a higher priority) and is characterized by a 4-tuple

$$(C_{i,k}, D_{i,k}, T_{i,k}, O_{i,k})$$

where

- $C_{i,k}$ denotes its worst-case execution time (WCET),
- $D_{i,k}$ its relative deadline,
- $T_{i,k}$ its activation period, and
- $O_{i,k}$ its *offset*, i.e., the delay between the first activation of the task and the first release of its transaction Tr_i. In this paper, *constrained* offsets are assumed, formally $O_{i,k} < T_{i,k}$.

Tasks within the same transaction share the same clock. All tasks of Tr_i being periodic, their offsets define a precise timing relation between them.

Task $\tau_{i,k}$ activates periodically its jobs at $O_{i,k}+m.T_{i,k}$ with $m \geq 0$. A job $\jmath$ of a task $\tau_{i,k}$ is characterized by

- its activation time $act_{i,k}(\jmath)$,
- its finishing time $end_{i,k}(\jmath)$ and
- its computation time $c_{i,k}(\jmath)$ ($c_{i,k}(\jmath) \leq C_{i,k}$)

Its response time $RT_{i,k}(\jmath)$ is defined as $end_{i,k}(\jmath) - act_{i,k}(\jmath)$. The worst-case response time (WCRT) of task $\tau_{i,k}$, denoted $wcrt_{i,k}$, is the largest possible response time among all jobs of task $\tau_{i,k}$.

The model does not suppose any global synchronization between transactions. Any possible time shift between any two transactions is assumed to be possible and must be considered by the analysis.

B. Notations and definitions

We note $hep(k)$, $hp(k)$, $lp(k)$ the sets of tasks of the system under study whose priorities are higher than or equal to, higher than or lower than k, respectively.

The RTAs considered here rely on the well-known concept of busy window which we define now.

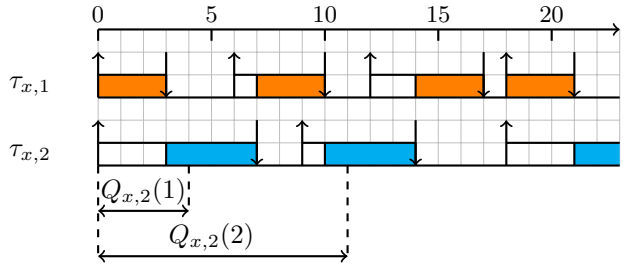

Figure 1: Example of queueing prefixes in a busy window.

Definition 1 (Level-k quiet time)**.** *An instant t is said to be a* level-k quiet time *if all jobs of priority higher than or equal to k released strictly before t have completed at t.*

Definition 2 (Level-k busy window)**.** *A time interval $[t_1, t_2[$ is said to be a* level-k busy window *if:*

1) *t_1 and t_2 are level-k quiet times;*
2) *there is no level-k quiet time in $]t_1, t_2[$; and*
3) *at least one job with a priority higher than or equal to k is released in $[t_1, t_2[$.*

Clearly, a job with a priority higher than or equal to k has completed by the end of its level-k busy window. In other words, its response time can be bounded by the length of the corresponding busy window. Such a bound is however quite coarse. In particular, there may be several jobs of the same task activated in the same busy window. We thus consider the response time of each level-k job in a level-k busy window. To this aim, we use the notions of phase and queueing prefix as defined in [15].

Definition 3 (Queueing prefix)**.** *The* q-th queueing prefix *of task $\tau_{i,k}$ in a level-k busy window $[t_1, t_2[$ is the time interval $[t_1, t_q]$ where t_q is the instant at which the q-th job of task $\tau_{i,k}$ receives its first service (*i.e., *is scheduled for the first time).*

Fig. 1 shows the first and second queueing prefixes of a task $\tau_{x,2}$ in a level-2 busy window, $Q_{x,2}(1)$ and $Q_{x,2}(2)$ respectively.

Definition 4 (Phase)**.** *The* phase *of the q-th job $\jmath$ of task $\tau_{i,k}$ in a level-k busy window $[t_1, t_2[$ is the duration $act_{i,k}(\jmath) - t_1$.*

Due to the FPNP scheduling policy, a lower priority job than k can be executed at the beginning of a level-k busy window. This is referred to as the *blocking factor*. It is easy to prove that the blocking factor of a level-k busy window is bounded by:

$$B_k = \max_{\tau_{i,x} \in lp(k)} (C_{i,x} - 1) \tag{1}$$

Indeed, the worst case is when the lower priority task with the largest worst-case execution time activates a job with such an execution time just one time unit before the start of the level-k busy window.

Another key notion for RTA is the workload of a task, which quantifies its request for resources.

Definition 5. *The workload bound function of task $\tau_{i,k}$ for a given interval $[t_1, t_1 + \Delta[$ is defined as:*

$$wl^+_{\tau_{i,k}}(t_1, \Delta) = (\underbrace{\lceil \frac{\Delta - \theta_{i,k}(t_1)}{T_{i,k}} \rceil}_{n_a}) C_{i,k} \tag{2}$$

where $\theta_{i,k}(t_1) = (T_{i,k} + O_{i,k} - (t_1 \bmod T_{i,k})) \bmod T_{i,k}$ is the time duration between t_1 and the first activation of $\tau_{i,k}$ after t_1.

When the tasks of the same transaction have distinct periods, the RTA presented in the next section makes use of the so-called hyper-period of transactions.

Definition 6. *(Hyper-period). The hyper-period T_i^+ of a transaction $Tr_i := \{\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,M}\}$ is the least common multiple of the periods of all its tasks. Formally,*

$$T_i^+ = lcm\{T_{i,1}, T_{i,2}, \ldots, T_{i,M}\} \tag{3}$$

III. Certified RTAs for CAN

In this section, we describe the two RTAs for CAN that we use to certify the results of the RTaW-Pegase tool. The correctness of these RTAs has been proved using the Coq proof assistant [4] on top of the Prosa library [1]. In the following, we consider a task $\tau_{i,k}$ and describe how the two RTAs compute an upper bound on its worst-case response time. The presentation follows the Coq specification. We omit proofs of lemmas and theorems and refer the interested reader to the Coq source [2].

The two RTAs follow the same procedure:

1) Let $\jmath$ be a job of task $\tau_{i,k}$.
2) For all possible scenarios (precise or approximate) compute an upper bound $RT_{BW_k}(\jmath)$ of that job $\jmath$ by examining all level-k queueing prefixes in its corresponding level-k busy window BW_k.
3) The upper bound on the response time of $\jmath$, denoted by $RT_{BW_k}(\jmath)$, is the maximum of the results for all scenarios.

Both analyses rely heavily on the analysis of busy windows and queueing prefixes introduced in the previous section.

A. Busy window analysis

We start by describing how to analyze the response time of of a task $\tau_{i,k}$ in a concrete busy window. Assume that there exists a level-k busy window $[t_1, t_2[$ in which a job $\jmath$ of $\tau_{i,k}$ is released. It can be shown that if the utilization is below 100%, it is always possible to compute that busy window. Assume that this job is the q-th job of task $\tau_{i,k}$ arrived in the busy window $[t_1, t_2[$. Let $Q_{i,k}(q)$ denote the q-th queueing prefix of task $\tau_{i,k}$ (see Def. 3) in that busy window, then $\jmath$ finishes at the latest at

$$t_1 + Q_{i,k}(q) + c_{i,k}(\jmath) - 1$$

The response time of this instance is bounded by:

$$RT_{i,k}(\jmath) \leq t_1 + Q_{i,k}(q) + c_{i,k}(\jmath) - 1 - act_{i,k}(\jmath)$$

The phase of $\jmath$ (see Def. 4) can also been defined as

$$act_{i,k}(\jmath) - t_1 = \theta_{i,k}(t_1) + (q-1) * T_{i,k}$$

As a result, the bound of the response time of job $\jmath$ can be rewritten as

$$RT_{i,k}(\jmath) \leq Q_{i,k}(q) - \underbrace{(\theta_{i,k}(t_1) + (q-1) * T_{i,k})}_{\text{phase}} + c_{i,k}(\jmath) - 1$$

Let us write $BW_k = t_2 - t_1$ to denote the size of the busy window. We know that there are at most

$$q^+_{\tau_{i,k},BW_k} = \left\lceil \frac{BW_k}{T_{i,k}} \right\rceil$$

jobs of task $\tau_{i,k}$ in that busy window. Therefore, within busy window $[t_1, t_2[$, the WCRT of task $\tau_{i,k}$ can be locally bounded by $RT_{BW_k}(\jmath)$ defined as

$$\max_{q \leq q^+_{\tau_{i,k},BW_k}} (Q_{i,k}(q) - \underbrace{(\theta_{i,k}(t_1) + (q-1) * T_{i,k})}_{\text{phase}} + c_{i,k}(\jmath_q) - 1)$$

where $\jmath_q$ represents the q-th job of task $\tau_{i,k}$ released in the busy window.

To find the WCRT of task $\tau_{i,k}$, we must find, for any possible scenario,

1) an upper bound on BW_k;
2) an upper bound on $Q_{i,k}(q)$ for any $q \leq q^+_{\tau_{i,k},BW_k}$;
3) a lower bound on $\theta_{i,k}(t_1)$.

BW_k and $Q_{i,k}(q)$ are defined as fixed points of two workload functions. In order to define these functions, we must first define the notion of workload. Note that this definition is different from the workload bound function defined in Sec. II.

Definition 7 (workload)**.** *The* workload $wl\tau_{j,l}(t_1, \Delta)$ *of task* $\tau_{j,l}$ *in a time interval* $[t_1, t_1 + \Delta[$ *is the cumulative cost (*i.e., *required service time) of its jobs released in that interval.*

BW_k can be found by computing the least fixed point of the following equation:

$$BW_k = f_B(BW_k)$$

where

$$f_B(\Delta) = b_k(t_1, \Delta) + \sum_{\substack{\tau_{j,l} \in Tr_j \\ Tr_j \in S \\ l \leq k}} wl_{\tau_{j,l}}(t_1, \Delta)$$

and $b_k(t_1, BW_k)$ is the blocking factor, that is the time duration at the beginning of the busy window when a lower priority task may execute.

Similarly, $Q_{i,k}(q)$ can be found by computing the least fixed point of the following equation:

$$Q_{i,k}(q) = f_Q(q, Q_{i,k}(q))$$

where

$$\begin{aligned} f_Q(q, Q_{i,k}(q)) \quad &= b_k(t_1, \Delta) + \textstyle\sum_{\substack{\tau_{j,l} \in Tr_j \\ Tr_j \in S \\ l < k}} wl_{\tau_{j,l}}(t_1, \Delta) \\ &+ wl_{\tau_{i,k}}(t_1, \theta_{i,k}(t_1) + (q-1) * T_{i,k}) + 1 \end{aligned}$$

A static analysis will have to find upper bounds of BW_k and $Q_{i,k}(q)$. To this aim, it is sufficient to find two functions that bound $f_B(\Delta)$ and $f_Q(q, \Delta)$ and to compute their fixed points. The correctness of this approach is expressed by the following lemma.

Lemma 1. *Let* $f, g : \mathbb{N} \to \mathbb{N}$ *be two functions and* Δ_1 *and* Δ_2 *be fixed points of the equations* $\Delta = f(\Delta)$ *and* $\Delta = g(\Delta)$ *then, if for all* $x : \mathbb{N},\ f(x) \leq g(x)$ *and, for all* $x : \mathbb{N}^+,\ x < \Delta_1$*, we have* $x < f(x)$*, then* $\Delta_1 \leq \Delta_2$.

Finally, computing an upper bound on the WCRT of task $\tau_{i,k}$ amounts to finding a finite set of scenarios such that $f_B(\Delta)$ and $f_Q(q, \Delta)$ for any busy window are bounded by the corresponding functions of a scenario in that set. The WCRT of the task $\tau_{i,k}$ is found by taking the maximum WCRT found for all these scenarios.

B. Precise analysis

The precise analysis considers the finite set of scenarios corresponding to the cases where

1) all jobs in the busy window take their worst-case execution time to complete; and
2) t_1 is aligned with an activation in each transaction.

The set of possible alignments corresponds to the set of scenarios.

We must show that, for any concrete busy window starting at t_1, there is a scenario belonging to the set described above that maximizes the functions $f_B(\Delta)$ and $f_Q(q, \Delta)$ and minimizes $\theta_{i,k}(t_1)$.

First, we show a lower bound of $\theta_{i,k}(t_1)$,

Lemma 2 (Alignment-θ)**.** *For any task* $\tau_{j,l}$ *in the system, and for any time duration* Δ*,*

$$\theta_{i,k}(O_j^{t_1}) \leq \theta_{i,k}(t_1) \tag{4}$$

where $O_j^{t_1}$ *is the duration between the beginning of the busy window and the first activation with a priority higher than or equal to* k *in* Tr_j *that occurs after* t_1.

Next, let us compute upper bounds of functions $f_B(\Delta)$ and $f_Q(q, \Delta)$. To this aim we bound the blocking time and the workload.

It is easy to prove that the actual blocking time in a level-k busy window is bounded by B_k (see Equation. 1):

Lemma 3. $b_k(t_1, BW_k) \leq B_k$

For any task $\tau_{j,l}$ in the system, and for any time instant t_1 and time duration Δ, the workload of task $\tau_{j,l}$ is maximized when all its jobs take their WCET (see Def. 5). Formally,

Lemma 4 (WCET)**.** $wl_{\tau_{j,l}}(t_1, \Delta) \leq wl^+_{\tau_{j,l}}(t_1, \Delta)$

For any task $\tau_{j,l}$ in the system, and for any time instant t_1 and time duration Δ, the workload of task $\tau_{j,l}$ is maximized when we right shift the interval $[t_1, t_1 + \Delta[$ to align it with $O_j^{t_1}$.

Lemma 5 (Alignment-wl)**.** $wl^+_{\tau_{j,l}}(t_1, \Delta) \leq wl^+_{\tau_{j,l}}(O_j^{t_1}, \Delta)$

With the three above lemmas, we can provide upper bounds to the functions $f_B(\Delta)$ and $f_Q(q, \Delta)$.

Lemma 6 (Bound-$f_B(\Delta)$)**.** *For any time duration* Δ*,*

$$f_B(\Delta) \leq f_B^+(\Delta) \tag{5}$$

where

$$f_B^+(\Delta) := B_k + \sum_{Tr_j \in S} wl^+_{Tr_j}(O_j^{t_1}, \Delta) \tag{6}$$

and

$$wl^+_{Tr_j}(O_j^{t_1}, \Delta) = \sum_{\substack{\tau_{j,l} \in Tr_j \\ l \leq k}} wl^+_{\tau_{j,l}}(O_j^{t_1}, \Delta) \tag{7}$$

Lemma 7 (Bound-$f_Q(q, \Delta)$)**.** *In any level-k busy window* BW_k *and for any time duration* Δ

$$f_Q(q, \Delta) \leq f_Q^+(q, \Delta) \tag{8}$$

where

$$\begin{aligned} f_Q^+(q, \Delta) := B_k \\ &+ \sum_{\substack{Tr_j \in S \\ j \neq i}} wl^+_{Tr_j}(O_j^{t_1}, \Delta) \\ &+ \sum_{\substack{\tau_{i,h} \in Tr_i \\ h < k}} wl^+_{\tau_{i,h}}(O_i^{t_1}, \Delta) \\ &+ wl^+_{\tau_{i,k}}\left(O_i^{t_1}, \theta_{i,k}(O_i^{t_1}) + (q-1) * T_{i,k}\right) + 1 \end{aligned} \tag{9}$$

Let LO^{t_1} be a list of alignments which consists of one $O_j^{t_1}$ for each transaction $Tr_j \in S$ and let BW_k^+ be the least fixed point of the following equation:

$$BW^+_{LO^{t_1}} = f_B^+(BW^+_{LO^{t_1}}) \tag{10}$$

For any $q \leq q^+_{BW^+_{LO^{t_1}}}$, we compute the least fixed point of equation:

$$Q^+_{LO^{t_1}} = f_Q^+(q, Q^+_{LO^{t_1}}) \tag{11}$$

Then, the response times of jobs of task $\tau_{i,k}$ released in the busy window $[t_1, t_2[$ are upper bounded by $RT^+_{LO^{t_1}}(\tau_{i,j})$ defined as:

$$\max_{q \leq q^+_{BW^+_{LO^{t_1}}}} \left\{ Q^+_{LO^{t_1}}(q) - \underbrace{(\theta_{i,k}(O_i^{t_1}) + (q-1) * T_{i,k})}_{\text{phase}} + C_{i,k} - 1 \right\} \tag{12}$$

To upper bound the WCRT $wcrt_{i,k}$ of the task $\tau_{i,k}$, we need to test all possible such $O_j^{t_1}$ for each transaction Tr_j. The list LO_j of candidates for $O_j^{t_1}$ for each transaction Tr_j is composed of all possible activations of jobs with a higher priority than k ($\in hep(k)$) in the transaction Tr_j within its hyper-period T_j^+:

$$LO_j = \bigcup_{\tau_{j,l} \in Tr_j \cap hep(k)} \{o \mid o = O_{j,l} + x * T_{j,l}, o < T_j^+, x \in \mathbb{N}\}$$

This list represents all possible alignments of the busy window with an activation. The list of all scenarios is made of all combinations of alignments over all transactions. The WCRT $wcrt_{i,k}$ of task $\tau_{i,k}$ is bounded by the maximal WCRT of all scenarios. Formally,

Theorem 1. *Let* $\times$ *denote the cartesian product, then*

$$wcrt_{i,k} \leq \max_{o \in LO_1 \times \cdots \times LO_N} RT_o^+(\tau_{i,k})$$

C. Approximate analysis

For large systems, the number of precise scenarios explodes and the precise analysis quickly becomes intractable. In this subsection, we present a more efficient but approximate analysis. It follows the same approach as presented in [24]. Its principle is to maximize the workload of each transaction.

First, we define the approximate workload bound function of a transaction.

Definition 8. *The approximate workload bound function of a transaction* Tr_j *for the duration* Δ *is defined as the maximum workload among all possible alignments represented by* LO_j*:*

$$wl^*_{Tr_j}(\Delta) = \max_{o \in LO_j} \left\{ \sum_{\substack{\tau_{j,l} \in Tr_j \\ l \leq k}} wl^+_{\tau_{j,l}}(o, \Delta) \right\}$$

Functions $f_B^+(\Delta)$ and $f_Q^+(q, \Delta)$ are upper bounded by using $wl^*_{Tr_j}(\Delta)$ for each transaction Tr_j. However, in order to obtain a tighter bound, we compute the precise workload of transaction Tr_i of task $\tau_{i,k}$ (*i.e.,* the task we analyze).

Lemma 8 (Bound-$f_B^+(\Delta)$)**.** *For any time duration* Δ *and any* $o \in LO_i$

$$f_B^+(\Delta) \leq f_B^*(\Delta)$$

where

$$f_B^*(\Delta) := B_k + \sum_{\substack{Tr_j \in S \\ j \neq i}} wl^*_{Tr_j}(\Delta) + \sum_{\substack{\tau_{i,l} \in Tr_i \\ l \leq k}} wl^+_{\tau_{i,l}}(o, \Delta)$$

Lemma 9 (Bound-$f_Q^+(q, \Delta)$)**.** *For any time duration* Δ

$$f_Q^+(q, \Delta) \leq f_Q^*(q, \Delta)$$

where

$$\begin{aligned} f_Q^*(q, \Delta) := B_k \\ &+ \sum_{\substack{Tr_j \in S \\ j \neq i}} wl^*_{Tr_j}(\Delta) \\ &+ \sum_{\substack{\tau_{i,h} \in Tr_i \\ h < k}} wl^+_{\tau_{i,h}}(O_i^{t_1}, \Delta) \\ &+ wl^+_{\tau_{i,k}}\left(O_i^{t_1}, \theta_{i,k}(O_i^{t_1}) + (q-1) * T_{i,k}\right) + 1 \end{aligned} \tag{13}$$

We compute $BW^*_{O_i^{t_1}}$ the least fixed point of equation

$$BW^*_{O_i^{t_1}} = f_B^*(BW^*_{O_i^{t_1}})$$

and, for each $q \leq q^{+}_{BW^*_{O_i^{t_1}}}$, the least fixed point of equation:

$$Q^*_{O_i^{t_1}} = f^*_Q(q, Q^*_{O_i^{t_1}})$$

then, the response time of jobs of task $\tau_{i,k}$ released in the busy window $[t_1, t_2[$ is upper bounded by $RT^*_{O_i^{t_1}}(\tau_{i,k})$ defined as:

$$\max_{q \leq q^{+}_{BW^*_{O_i^{t_1}}}} \left\{ Q^*_{O_i^{t_1}}(q) - \underbrace{(\theta_{i,k}(O_i^{t_1}) + (q-1) * T_{i,k})}_{\text{phase}} + C_{i,k} - 1 \right\}$$

Then, the WCRT $wcrt_{i,k}$ of task $\tau_{i,k}$ is the maximum of these values for all possible alignments represented by LO_i.

Theorem 2.

$$wcrt_{i,k} \leq \max_{o \in LO_i} RT^*_o(\tau_{i,k})$$

Compared to the precise analysis, we do not consider all possible combinations (the cartesian product) of all alignments of all transactions. The approximate scenarios consist in the alignments of the considered transaction Tr_i only; the other transactions are represented by the approximate workload bound function.

IV. Combined RTA and Result Certifier

In the previous section, we presented two analyses for CAN with offsets: a precise analysis and an approximate analysis. The precise analysis computes a tight bound of the WCRT but is potentially very costly. The approximate analysis is much more efficient but yields pessimistic bounds.

In this section, we present

1) a RTA that combines both analyses to compute a tight bound of WCRT more efficiently; and
2) CertiCAN, a certifier combining two certified precise and approximate analyses, which can certify the results of industry analyzers for large systems.

A. Combined RTA

The combined analysis is based on a precise and approximate analyses. Its main features are:

- It uses the approximate analysis to avoid unnecessary computations and thus increase performance;
- It nevertheless computes the same results as the precise analysis.

The precise analysis, R_p, considers a list of precise scenarios S_p and computes a worst case response time for each. Its result is the maximum of all these computations that is

$$\max_{s_p \in S_p}(R_p\ s_p)$$

The approximate analysis, R_a, does the same on a list of approximate scenarios S_a. The two key properties of a valid approximate analysis are that:

- Each approximate scenario s_a dominates a list of precise scenarios written Ds_a *i.e.,*

$$max_{s_p \in (Ds_a)}(R_p\ s_p) \leq R_a s_a \tag{14}$$

- The list S_a of approximate scenarios dominates all precise scenarios of S_p.

Therefore, we know that the result of the approximate analysis is an over approximation of the precise result, that is

$$max_{s_p \in S_p}(R_p\ s_p) \leq max_{s_a \in S_a}(R_a s_a) \tag{15}$$

The combined analysis is based on the following observation: If the WCRT obtained for an approximate scenario s_a is smaller than the WCRT found so far on the set of precise scenarios visited, then there is no need to analyze the precise scenarios dominated by s_a.

Algorithm 1 Combined Response Time Analysis

```
procedure CRTA(n, l)
    match l with
    | nil => return n                    ▷ l empty: returns the WCRT
    |(r, s) :: l' =>                                  ▷ otherwise
    if n ≥ r then CRTA(n, l')
    else  m ← max_{s_p ∈ (D s)}(R_p s_p);   ▷ local precise result
          CRTA(max(n, m), l')
    end if
end procedure
     CRTA(0,sort (map (λs.(R_a s, s)) S_a))
```

The structure of the combined analysis is shown in Algorithm 1. First, the approximate analysis is applied to each approximate scenario. These results paired with the corresponding scenario are sorted in descending order ($sort(map(\lambda s.(R_a s, s))\ S_a)$). Sorting the list in that order leads to considering the largest approximate WCRTs first. This heuristic relies on the intuition that the largest precise WCRT (which is the value to be found) is more likely to be dominated by a large approximate WCRT and therefore, will be found earlier with that ordering.

The CRTA function is called with this list and 0 as the initial WCRT to be found. CRTA considers each approximate WCRT of the argument list in turn. If the approximate WCRT is less than the current WCRT then CRTA proceeds with the next element of the list. Otherwise (and that is always the case in the first call of CRTA), the function computes the maximal response time on all precise scenarios dominated by the current approximate scenario ($\max_{s_p \in (D\ s)}(R_p\ s_p)$). This response time replaces the current WCRT if it is greater than the latter and CRTA proceeds with the next element in the list. When all the elements of the list have been considered, the computed WCRT, which can be proved to be the same as the precise WCRT, is returned.

Example 1. *Consider an example with 2 approximate scenarios, each one dominating 3 precise scenarios. In Fig. 2, vertices represent scenarios, edges represent the domination relation and labels next to vertices are their corresponding worst case response times. For this example, the combined analysis*

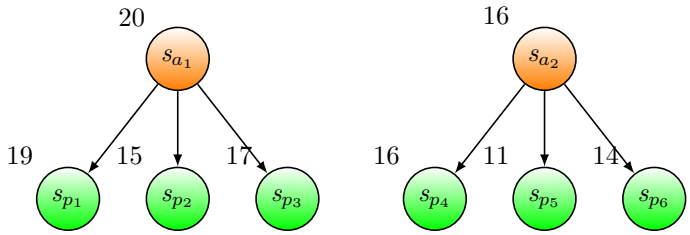

Figure 2: Scenario dominations.

CRTA which is called with 0 *and the list* $[(20, s_{a_1}); (16, s_{a_2})]$ *and proceeds as follows:*

- *the current approximate response time (*20*) is greater than the current WCRT (*0*) then the maximum response time of the three dominated precise scenarios is computed (*19*) and becomes the current WCRT;*
- *because* 19 *is greater than the next approximate response in the list (*16*), the response times of the corresponding dominated scenarios do not need to be computed (they are necessarily smaller);*
- *the analysis returns* 19 *which is the precise WCRT.*

The combined RTA returns the same WCRT as the precise analysis. Formally,

Theorem 3. *Let* R_p*,* R_a*,* S_p *and* S_a *verifying the domination equations 14 and 15 then*

$$\mathit{CRTA}(0, sort(map(\lambda s.(R_a s, s))\ S_a)) = \max_{s \in S_p}(R_p\ s)$$

B. Result certifier

The procedure of the result certifier is very similar to the combined RTA and they share the same time complexity in theory. In practice, since it starts with the precise WCRT to be checked, it is more efficient. Our experiments show that it has a reasonable scalability and can certify results of quite large systems.

To check that R_0 is equal to, or larger than the precise WCRT, CertiCAN considers each approximate WCRT of the argument list in turn. If the approximate WCRT is equal to, or less than R_0 then the certification is complete (and returns *True*) since all remaining approximate WCRT of the ordered list are also less than R_0. Otherwise, the local precise WCRT is computed on all precise scenarios dominated by the current approximate scenario. If it is equal to, or less than R_0, CertiCAN proceeds with the remaining list. Otherwise, it means that there is a precise WCRT greater than R_0 which is incorrect and CertiCAN returns *False*.

If the certifier returns true for a value R_0, it means that R_0 is greater than or equal to the precise WCRT. Formally,

Theorem 4. *Let* R_p*,* R_a*,* S_p *and* S_a *verifying the domination equations 14 and 15 then*

$$\mathit{CertiCAN}(sort(map(\lambda s.(R_a\ s, s))\ S_a)) \Rightarrow R_0 \geq \max_{s \in S_p}(R_p\ s)$$

Algorithm 2 The CertiCAN result Certifier

The variable R_0 contains the WCRT to certify

procedure CERTICAN(l)
  match l with
  | nil => **return** $True$ ▷ l empty: returns $True$
  |$(r, s) :: l'$ => ▷ otherwise ...
  if $R_0 \geq r$ **then return** $True$
  else $m \leftarrow \max_{s_p \in (D\ s)}(R_p\ s_p)$ ▷ local precise result
    if $R_0 \geq m$ **then** CERTICAN(l')
    else return $False$
    end if
  end if
end procedure
CertiCAN(sort (map $(\lambda s.(R_a s, s))\ S_a$))

V. Experimental Evaluation

Having completed the Coq formalization of our analyses, we used the Coq extraction feature to obtain four certified tools: a precise analyzer, an approximate analyzer and a combined analyzer as well as CertiCAN, the result certifier based on the combined analysis.

In this section, we evaluate these certified tools in terms of performance and scalability. We use synthetic test task sets that are generated by a simple homemade generator. It starts from an initial configuration based on the configuration for CAN analysis presented in [29] (see Table I). Then, the generator sets several parameters: the number of transactions, the number of tasks in each transaction, and the period of each transaction.

Table I: Initial configuration for the simple generator.

Payload	1 - 8 bytes
Period	$\{20, 50, 100, 200, 500\}$
Offset	$\{10, 20, \ldots, 500\}$
Priority	unique, arbitrary distribution
Transmission speed	500 kbits/s

In all figures, all results are obtained from an Intel Core i7@2.6GHz, 8Gb, 64bits laptop and reported only for schedulable systems, and passed the artifact evaluation [2].

A. Evaluation of extracted analyzers

First, we compare the three analyzers (precise, approximate and combined).

Fig. 3 shows that the precise analyzer has a very high time complexity compared to the approximate and combined analyzers. Even if it returns precise results, the performance of the combined analyzer efficiency remains close enough to the approximate analyzer.

Note that a figure showing the number of precise scenarios evaluated rather the runtime of the analysis would look very similar to Figure 3. The runtime of the analyses depends directly on the number of precise scenarios that are evaluated.

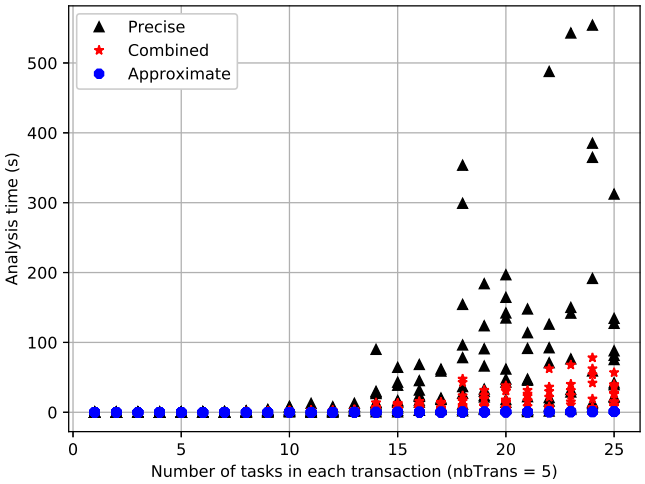

Figure 3: Comparison of the three certified analyzers

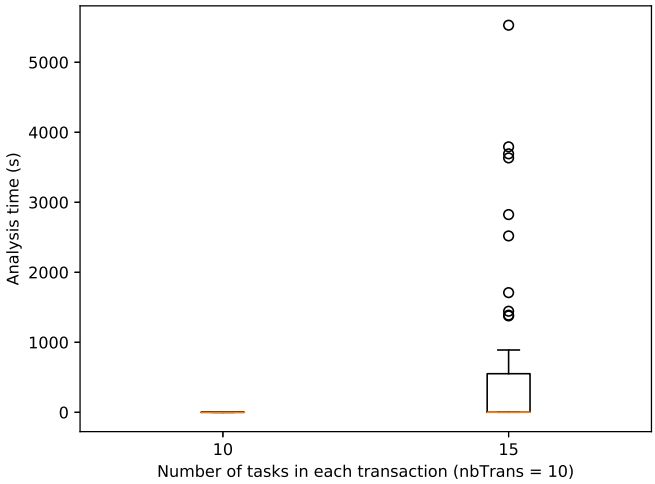

Figure 5: Certifying the results of RTaW-Pegase (nbTrans=10)

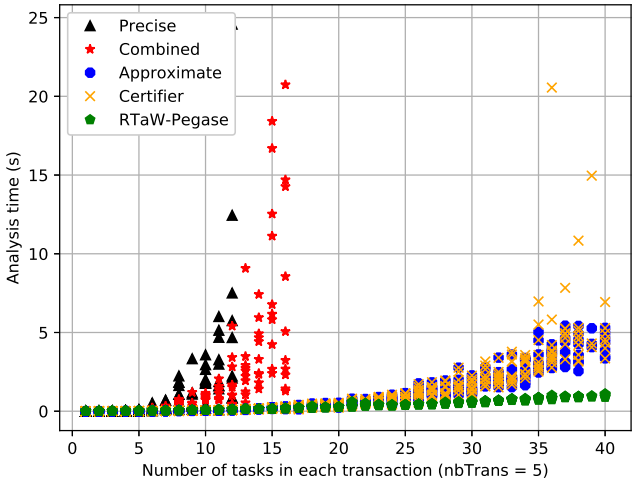

Figure 4: Certifying the results of RTaW-Pegase (nbTrans=5)

B. *Evaluation of the CertiCAN certifier*

Next, we evaluate CertiCAN by verifying the results produced by the industrial tool RTaW-Pegase. The CAN analyzer of RTaW-Pegase has a very good scalability due to some optimization techniques. On the other hand, CertiCAN is based on two unoptimized (but formally proved) analyses.

In Fig. 4, we consider models with five transactions and a number of tasks up to 40 (*i.e.,* systems with up to 200 tasks). After evaluating about 400 task sets (systems), we observe that

- the RTaW-Pegase analysis is very efficient;
- CertiCAN is much slower but still tractable. It is able to certify results of large systems at least with a reasonable number of transactions;
- Up to 35 tasks per transaction, the runtime of CertiCAN is close to that of the certified approximate analyzer.

In Fig. 5, we consider models with ten transactions and evaluate two different sizes of systems: one with 10 tasks in each transaction and the other with 15 tasks in each transactions. The utilization of generated systems lies between 0.2 and 0.9 and 80% of systems are analyzed in less than 1000 seconds. It takes up to 1.5 hours to certify results of the most complex systems (*e.g.,* 10 transactions and 15 tasks per transaction). The number of transactions is the main factor of complexity growth as it quickly increases the number of scenarios to analyze. The number of tasks and the utilization have a much lower impact.

These results show again that CertiCAN is able to certify the result of relatively large systems. As far as we know, even in modern cars, no more than 15-20 ECUs are connected to a single CAN bus [20]. Furthermore, not all tasks connected to a CAN bus are periodic [21] and among them sporadic tasks would not increase the number of scenarios to analyze. Note however that all systems analyzed above are such that tasks in the same transaction share the same period.

According to all our experiments, we found that CertiCAN is as efficient as our certified approximate analyzer, which is good sign even though it remains much slower than RTaW-Pegase. The main reason is that the implementation of CertiCAN has been kept quite simple to facilitate correctness proofs. We believe that it can be optimized by proving some additional lemmas. In particular, when transactions have tasks with harmonic periods all analyses are an order of magnitude slower. Indeed, the analyses must consider the hyper period and the number of scenario increases considerably. Optimizations should be considered to mitigate this effect; this is a topic for future work.

Even as it is, CertiCAN can certify the results of relatively large systems in less than a few hours. This can be considered

time consuming but it can also be argued that it is a small price to pay to obtain formal guarantees before putting a system into production.

VI. Discussion

A. Experience with the Coq proof assistant

We have formally proven in Coq the correctness of the precise and approximate RTAs presented in Section III [5]. Our proofs build upon the Prosa library [1] and use the basic definitions that it provides (task, job, arrival sequence, schedule, busy window, etc.). The combined analyzer and the result certifier (see Sec. IV) have also been specified and proved correct in Coq. Note that if proofs are machine-checked, this cannot be the case for specifications and theorem declarations. Besides using some basic definitions from Prosa, we also had to define the FPNP scheduling policy and the task model. Those specifications are quite small and simple compared to the proofs and can be scrutinized by the interested reader [2].

Here are a few figures to illustrate the complexity of the proof effort. The specifications and proofs related to task arrival and workload represent around 2500 LOC, the busy window analysis about 3000 LOC, the RTA analyses approximately 700 LOC and CertiCAN around 800 LOC.

Formalizing these developments in Coq requires more time and effort than on paper, but it also brings important benefits:

- It gives formal guarantees about the soundness of the specification and the absence of flaws in the proofs;
- It provides a better understanding of the role of each assumption, which helps to generalize proofs;
- The Coq extraction technique permits to produce formally verified tools (such as analyzers and certifiers) in the form of OCaml programs.

B. Possible extensions of the approach

One of the most interesting by-products is that formalization often leads to more general and reusable proofs. For instance, our proof of busy window analysis does not rely on a specific task model but on abstract functions. It can be reused for other task models as long as we have the corresponding abstract functions.

Also, we defined two RTAs by instantiating the abstract functions with two different workload functions. Actually, the approach could be applied to get other RTAs with different levels of approximation. The combined analysis and certifier depend of generic properties (domination relations) that do not rely on the specific real-time model under study. The correctness proofs for the combined algorithm would apply to many other kinds of analyses possibly disconnected to real-time theories. For the time being, we have considered only two different levels of approximation but the principle should hold for an arbitrary number of levels as long as they have the relevant domination properties.

C. On result certification versus tool certification

Proving the correctness of highly optimized analyzers is usually very costly and complex. Furthermore, each new version of the analyzer requires additional, potential large, proof effort. This makes the certification of results a very appealing alternative. The results of complex, optimized and/or changing analyzers can be checked by the same certifier. Of course, the certifier itself should be proved correct. In our context, the efficiently of a certifier is not too critical: the analyzer can be used to experiment and tune the system and only its final results have to be certified once. Still, without the combination of precise and approximate analyses, CertiCAN could not have been used to certify the results of large systems. CertiCAN has been kept as simple as possible and its code is a naive implementation of the algorithm 2. It is likely that its efficiency could be improved without demanding too much additional proof work.

VII. Related work

To the best of our knowledge, CertiCAN is the first tool for certifying real-time systems analysis results. It does, however, build on top of existing results in different areas.

A. Formal proofs for real-time systems analysis

CertiCAN uses definitions and lemmas from the Prosa library [1]. Prosa already contains RTA proofs, from which one can directly extract certified RTA tools. These proofs, however, do not cover task sets with offsets and the bounds they could provide would be unnecessarily pessimistic.

Prosa is the largest effort to date regarding the certification of real-time systems analyses. It is however not the first one. Previous publications in the area include [13], [14] and [17] based on the PVS proof assistant and which use state machines as the underlying formalism. The first two papers focus on the priority ceiling protocol and the latter on the scheduler of the Deos real-time operation system. While related to our work in a broader sense, these contributions do not tackle the problem of certifying RTA results.

Closer to us, a recent attempt [19] aims at certifying the results of Network Calculus computations using the interactive proof assistant Isabelle/HOL. In particular, [19] makes a case for result certification. The presented results are however preliminary and appear to have been discontinued. Our paper can thus be seen as the concrete realization, with a different proof assistant and another underlying analysis technique, of the idea proposed in [19].

B. Abstraction refinement

Our combined RTA follows a principle that is similar to the abstraction refinement method used in [22] and [23]. In particular, these two papers already use two different abstraction levels to compute precise bounds with increased efficiency. The main difference is that [22] deals with the analysis of digraph tasks with constrained deadlines, which does not fit

the CAN context. Also, this approach proves to be particularly well suited for result certification: Having the response time bound to certify given as input makes the combined analysis all the more efficient.

VIII. Conclusion

In this paper, we have presented CertiCAN, a tool extracted from Coq proofs for the certification of CAN analysis results.

The analysis underlying CertiCAN is based on a combined use of two well-known CAN analysis techniques, one precise and the other approximate. The resulting analysis is as tight as the precise analysis, but much faster. All three analyses have been proven correct in Coq on top of the Prosa library.

We have shown that the CertiCAN approach, which provides result certification rather than tool certification, is a realistic solution for industrial practice. The reason for this is twofold: First, it is flexible and light-weight in the sense that it does not depend on the internal structure of the analysis tool that it complements. Second, it is efficient enough in terms of computation time. In particular, it is able to certify results computed by RTaW-Pegase, an industrial CAN analysis tool, even for large systems.

We believe that this work represents a significant step toward a formal certification of real-time systems analysis results in general. In particular, the underlying technique can be reused for any other system model for which there exist RTAs with different levels of precision. Future work includes the extension of the approach to networks of CAN buses, and to other communication protocols. Also, we plan to use the results presented in this paper, which have been obtained using synthetic test cases, to convince industrial partners to provide us with real case studies.

Acknowledgment

This work has been partially supported by the LabEx PERSYVAL-Lab (grant ANR-11-LABX-0025-01) through the CASERM project and the French national research organization ANR (grants ANR-15-CE25-0008 and ANR-17-CE25-0016) through the VOCAL and RT-PROOFS projects. We would like to thank RealTime-at-Work for granting us an academic license and Jiajie Wang for providing an XML parser.

References

[1] A Library for formally proven schedulability analysis. http://prosa.mpi-sws.org/.

[2] A Tool for the Coq Certification of CAN Analysis Results. https://team.inria.fr/spades/certican/.

[3] The CompCert C compiler. https://www.absint.com/compcert/.

[4] The Coq proof assistant. http://coq.inria.fr.

[5] Coq sources and Ocaml code of CertiCAN. Available from the track chair upon request.

[6] The Isabelle proof assistant. https://isabelle.in.tum.de/.

[7] RTaW-Pegase: a Tool for Modeling, Simulation and automated Configuration of communication networks. http://www.realtimeatwork.com/software/rtaw-pegase/.

[8] The seL4 microkernel. https://sel4.systems/.

[9] K. Bletsas, N. C. Audsley, W. Huang, J. Chen, and G. Nelissen. Errata for three papers (2004-05) on fixed-priority scheduling with self-suspensions. *LITES*, 5(1):02:1–02:20, 2018.

[10] Bosch. CAN specification version 2.0. Technical report, Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart, 1991.

[11] F. Cerqueira, F. Stutz, and B. B. Brandenburg. Prosa: A case for readable mechanized schedulability analysis. In *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*, pages 273–284. IEEE, 2016.

[12] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[13] B. Dutertre. Formal analysis of the priority ceiling protocol. In *21st IEEE Real-Time Systems Symposium (RTSS)*, pages 151–160, 2000.

[14] B. Dutertre and V. Stavridou. Formal analysis for real-time scheduling. In *19th Digital Avionics Systems Conference (DASC)*, 2000.

[15] P. Fradet, X. Guo, J.-F. Monin, and S. Quinton. A generalized digraph model for expressing dependencies. In *RTNS'18-26th International Conference on Real-Time Networks and Systems*, pages 1–11, 2018.

[16] X. Guo, S. Quinton, P. Fradet, and J.-F. Monin. Work-in-progress: Toward a coq-certified tool for the schedulability analysis of tasks with offsets. In *Real-Time Systems Symposium (RTSS), 2017 IEEE*, pages 387–389. IEEE, 2017.

[17] V. Ha, M. Rangarajan, D. Cofer, H. Rues, and B. Dutertre. Feature-based decomposition of inductive proofs applied to real-time avionics software: An experience report. In *26th International Conference on Software Engineering (ICSE)*, pages 304–313, 2004.

[18] X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[19] E. Mabille, M. Boyer, L. Fejoz, and S. Merz. Towards certifying network calculus. In *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, pages 484–489, 2013.

[20] A. Monot, N. Navet, B. Bavoux, and C. Maxim. Fine-grained simulation in the design of automotive communication systems. In *ERTSS-Embedded Real Time Software and Systems-2012*, 2012.

[21] S. Quinton, T. T. Bone, J. Hennig, M. Neukirchner, M. Negrean, and R. Ernst. Typical worst case response-time analysis and its use in automotive network design. In *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*, pages 44:1–44:6, 2014.

[22] M. Stigge, N. Guan, and W. Yi. Refinement-based exact response-time analysis. In *26th Euromicro Conference on Real-Time Systems, ECRTS 2014, Madrid, Spain, July 8-11, 2014*, pages 143–152, 2014.

[23] M. Stigge and W. Yi. Combinatorial abstraction refinement for feasibility analysis of static priorities. *Real-Time Systems*, 51(6):639–674, 2015.

[24] K. Tindell. *Using offset information to analyse static priority pre-emptively scheduled task sets*. Technical report YCS 182. University of York, Department of Computer Science, 1992.

[25] K. Tindell. *Adding time-offsets to schedulability analysis*. University of York, Department of Computer Science, 1994.

[26] K. Tindell and A. Burns. Guaranteeing message latencies on controller area network (CAN). In *Proceedings of 1st international CAN conference*, pages 1–11, 1994.

[27] K. Tindell, A. Burns, and A. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[28] K. Tindell, H. Hanssmon, and A. J. Wellings. Analysing real-time communications: Controller area network (CAN). In *Proceedings of the 15th IEEE Real-Time Systems Symposium (RTSS '94), San Juan, Puerto Rico, December 7-9, 1994*, pages 259–263, 1994.

[29] P. M. Yomsi, D. Bertrand, N. Navet, and R. I. Davis. Controller area network (can): Response time analysis with offsets. In *2012 9th IEEE International Workshop on Factory Communication Systems*, pages 43–52, May 2012.