# Work In Progress: Toward a Coq-certified Tool for the Schedulability Analysis of Tasks with Offsets

Xiaojie Guo[1,2], Sophie Quinton[1], Pascal Fradet[1] and Jean-François Monin[2]
[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 presents the first steps toward a formally proven tool for schedulability analysis of tasks with offsets. We formalize and verify the seminal response time analysis of Tindell by extending the *Prosa* proof library, which is based on the Coq proof assistant. Thanks to Coq's extraction capabilities, this will allow us to easily obtain a certified analyzer. Additionally, we want to build a Coq certifier that can verify the correctness of results obtained using related (but uncertified), already existing analyzers. Our objective is to investigate the advantages and drawbacks of both approaches, namely the certified analysis and the certifier. The work described in this paper as well as its continuation is intended to enrich the *Prosa* library.

## 1. INTRODUCTION

For hard real-time systems used in application domains such as avionics, missing a deadline may have catastrophic consequences. Schedulability analysis, which aims to guarantee the absence of any deadline miss, is therefore of the utmost importance in the verification of such critical systems. Schedulability analysis has been the subject of intensive research over the past decades and many different techniques have been implemented into tools, including several commercial analyzers such as NETCAR-Analyzer [2], SymTA/S [3], etc. There is, however, no guarantee that the results provided by these tools are correct, for two reasons:

1. The theory that they build upon may be flawed, as in the original schedulability analysis for CAN messages [5];
2. They may contain undetected bugs and thus return erroneous results.

Our general objective is to provide schedulability analysis tools with formal guarantees using the Coq [4] proof assistant. We aim to investigate two options:

1. Write the entire analyzer and its correctness proof in Coq and extract from it a certified[1] Caml implementation;
2. Write in Coq a program that can guarantee that a given result is correct, in contrast to computing that result — for example verifying that a given value is indeed a fixed point rather than computing such a fixed point. The extracted Caml tool would then have to be used together with an existing schedulability analysis tool.

Which option will prove the best compromise in terms of computational complexity, implementation effort, acceptance from industry is still an open question.

As a first step, we focus on the schedulability analysis of uniprocessor systems of tasks with offsets dispatched according to the Fixed Priority Preemptive (FPP) scheduling policy. Schedulability analysis in presence of offsets is a non-trivial problem with a high computational complexity. In contrast to the traditional (offset oblivious) analysis, many scenarios must be tested and compared to identify which one represents the worst-case scenario.

In this paper, we formalize and prove in Coq an analysis technique presented in [6] and [7] to bound the response time of tasks with offsets, and thus decide on the schedulability of a system. In addition to paving the way toward a Coq-certified tool for schedulability analysis of task sets with offsets, our formalization effort underlines implicit assumptions made by the author in [6] and eases the generalization of the verified analysis. Our formalization is based on Prosa [1], a library of definitions and proofs for real-time schedulability analysis.

The rest of this paper is laid out as follows: Section 2 presents the system model; in Section 3, we present our formalization of the existing analysis. Section 4 discusses the Coq proof; finally, we present future work in Section 5.

## 2. SYSTEM MODEL

The system model considered in this paper is the one presented by Tindell in [6]. It consists of a set of periodic tasks running on a uniprocessor using FPP scheduling. A system $S$ is a set of transactions

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

where each transaction is a set of tasks $Tr_i := \{\ldots, \tau_{i,k}, \ldots\}$ sharing the same period $T_i$. A task is described as a tuple $\tau_{i,k} := \{C_{i,k}, O_{i,k}, D_{i,k}, T_i\}$ where $k$ denotes its priority (a greater $k$ means a higher priority), $C_{i,k}$ its worst-case execution time and $D_{i,k}$ its deadline. There is no global synchronization between transactions so there is an indefinite time

[*]This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

[1]Certified in this context means formally verified in Coq.

shift between any two transactions. Within a transaction $Tr_i$, however, there is a fixed and known relation between task activations: an instance of a task $\tau_{i,k}$ is activated $O_{i,k}$ time after each release of $Tr_i$, where $O_{i,k}$ is called an *offset.*

In hard real-time systems, the duration between the activation and the completion of any instance of a task must be less than its deadline. The $j$-th instance (or job) of a task $\tau_{i,k}$ is characterized by its activation time $act_{i,k}(j)$, its finishing time $end_{i,k}(j)$ and its computation time $c_{i,k}(j)$ — with $c_{i,k}(j) \leq C_{i,k}$. Its response time $RT_{i,k}(j)$ is defined as $end_{i,k}(j) - act_{i,k}(j)$. The worst-case response time of task $\tau_{i,k}$, denoted $wcrt_{i,k}$, is the largest possible response time among all instances of task $\tau_{i,k}$.

Figure 1 shows an example with two transactions and four tasks. Black upward arrows indicate task activations, black downward arrows show task finishing times while red upward arrows represent the periodic release of transactions. Grey boxes indicate time intervals during which a task is executing and white boxes intervals during which a task is delayed by a higher priority task. Note that task $\tau_{1,8}$, for example, is always activated one time unit after task $\tau_{1,10}$, but the alignment between activations of $\tau_{1,8}$ and $\tau_{i,9}$ is unknown and may vary.

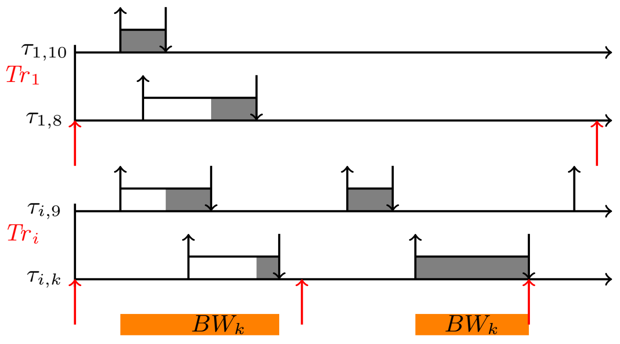

Figure 1: A system with 2 transactions and 4 tasks ($k < 8$).

The analysis considered in this paper relies on the following hypotheses (we discuss this further in the conclusion):

1. Tasks in the same transaction have the same period. Therefore, a task activation can be associated with a unique transaction release, and conversely.
2. (Restricted offsets) The offset of a task is strictly less than its period: $\forall \tau_{i,k}, O_{i,k} < T_i$.
3. (Implicit deadlines) The deadline of a task is equal to its period: $\forall \tau_{i,k}, D_{i,k} = T_i$.

The analysis relies heavily on the concept of *busy window* which we explain now. An instant $t$ is said to be a *level-$k$ quiet time* if all tasks of priority higher than or equal to $k$ released strictly before $t$ have completed at $t$. A *level-$k$ busy window* is a time interval $[t_1, t_2[$ in which:

1. a task with a priority higher than or equal to $k$ is activated at $t_1$;
2. $t_1$ and $t_2$ are level-$k$ quiet times;
3. there is no other level-$k$ quiet time between $t_1$ and $t_2$.

In Figure 1, two level-$k$ busy windows are shown using orange boxes marked with $BW_k$.

## 3. RESPONSE TIME ANALYSIS (RTA)

In this section, we reformulate the approximate RTA technique presented in [6]. We discuss in Section 4 how it relates to the original proof. From now on, we focus on a task $\tau_{i,k}$ and analyze its worst-case response time.

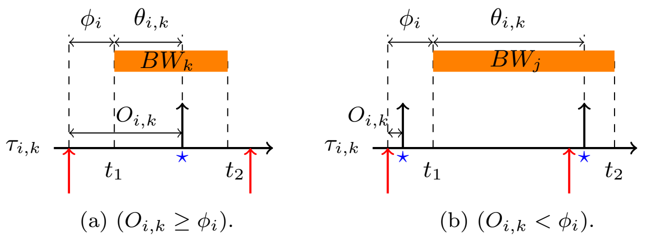

Figure 2: Relation between $O_{i,k}$ and $\phi_i$ (with $\star = act_{i,k}$).

Assume that there exists a level-$k$ busy window $[t_1, t_2[$ in which the $j$-th instance of $\tau_{i,k}$ is released[2]. This job finishes at the latest at $t_2$, so the response time of this instance can be bounded by:

$$RT_{i,k}(j) \leq t_2 - act_{i,k}(j) \tag{1}$$

which, with the notation $BW_k := t_2 - t_1$, is the same as

$$RT_{i,k}(j) \leq BW_k + t_1 - act_{i,k}(j) \tag{2}$$

The key to the analysis is to show that the response time is maximized when: (1) all jobs in the busy window take their worst-case execution time to complete; and (2) $t_1$ is aligned with an activation of each transaction.

The definition of a busy window implies that $BW_k$ is the least fixed point of $wl_{t_1}$:

$$BW_k = wl_{t_1}(BW_k) \tag{3}$$

where $wl_{t_1}(\Delta)$ is the *workload* of priority higher than or equal to $k$ that arrives between $t_1$ and $t_1+\Delta$. This workload is computed by adding up the execution times of all jobs of tasks with a priority higher than or equal to $k$ which are activated between $t_1$ and $t_1 + \Delta$.

In order to bound $RT_{i,k}(j)$, we first focus on upper bounding $wl_{t_1}$ and then reason about fixed points to bound $BW_k$. Let $\phi_i$ denote the duration between $t_1$ and the latest release of $Tr_i$ that precedes it:

$$\phi_i = t_1 \bmod T_i \tag{4}$$

Then, the duration $\theta_{i,k}$ between $t_1$ and the first activation of $\tau_{i,k}$ in $BW_k$ is:

$$\theta_{i,k} = (T_i + O_{i,k} - \phi_i) \bmod T_i \tag{5}$$

Note that this definition holds in the two cases shown in Figure 2.

LEMMA 1.

$$wl_{t_1} \leq \sum_{Tr_j \in S} wl^+_{j,t_1} \tag{6}$$

*where*

$$wl^+_{j,t_1}(\Delta) := \sum_{\substack{\tau_{j,l} \in Tr_j \\ k \leq l}} \left\lceil \frac{\Delta - \theta_{j,l}}{T_j} \right\rceil \times C_{j,l} \tag{7}$$

*with* $\theta_{j,l} = (T_j + O_{j,l} - \phi_j) \bmod T_j$ *with* $\phi_j = t_1 \bmod T_j$.

*Proof sketch.* $wl^+_{j,t_1}(\Delta)$ upper bounds the actual workload $wl_{j,t_1}(\Delta)$ induced by tasks of priority higher or equal to $k$ in $Tr_j$ between $t_1$ and $t_1 + \Delta$ by assuming all jobs take their

[2]Note that if the utilization is below 100%, it is always possible to compute that busy window.

worst-case execution time to finish. $\lceil\frac{\Delta-\theta_{j,l}}{T_j}\rceil$ is the number of activations of $\tau_{j,l}$ in $[t_1, t_1+\Delta[$, knowing that the first activation is at $t_1+\theta_{j,l}$. □

The principle of the approximate analysis is to maximize the workload induced by each transaction in isolation. This may introduce an overapproximation in the obtained bound but it greatly reduces the computational complexity of the analysis compared to the exact solution.

LEMMA 2. *For $Tr_j \in S$,*

$$wl^+_{j,t_1} \leq wl^*_j \tag{8}$$

*where*

$$wl^*_j := \max_{O\in\mathcal{O}_j}\{wl^+_{j,O}\}$$

$$\mathcal{O}_j := \{O_{j,l} \mid \tau_{j,l} \in Tr_j \wedge k \leq l\}$$

$$wl^+_{j,O}(\Delta) := \sum_{\substack{\tau_{j,l}\in Tr_j\\ k\leq l}} \lceil\frac{\Delta-\theta^O_{j,l}}{T_j}\rceil \times C_{j,l}$$

*with $\theta^O_{j,l} = (T_j + O_{j,l} - O) \mod T_j$.*

*Proof sketch.* Let $O^{t_1} = (O^{t_1}_1, \ldots, O^{t_1}_m)$ be such that $O^{t_1}_j$ is the offset of the first task of priority higher than or equal to $k$ in $Tr_j$ that is activated after $t_1$. It is fairly easy to prove that:

$$wl^+_{j,t_1}(\Delta) \leq wl^+_{j,O^{t_1}_j}(\Delta)$$ □

We now differentiate $Tr_i$ from the others and obtain:

$$wl_{t_1}(\Delta) \leq wl^+_{i,O^{t_1}_i}(\Delta) + \sum_{\substack{Tr_j\in S\\ j\neq i}} wl^*_j(\Delta) \tag{9}$$

Let $BW^*_{O^{t_1}_i}$ be the least fixed point of the above function upper bounding $wl_{t_1}$. Using properties of fixed point iteration, we can prove that $BW_k \leq BW^*_{O^{t_1}_i}$ and thus:

$$RT_{i,k}(j) \leq BW^*_{O^{t_1}_i} + t_1 - act_{i,k}(j) \tag{10}$$

We then prove that $act_{i,k}(j) - t_1 \geq (T_i + O_{i,k} - O^{t_1}_i) \bmod T_i$, so:

$$RT_{i,k}(j) \leq BW^*_{O^{t_1}_i} - (T_i + O_{i,k} - O^{t_1}_i) \bmod T_i \tag{11}$$

The main theorem follows:

THEOREM 1.

$$wcrt_{i,k} \leq \max_{O\in\mathcal{O}_i}\{BW^*_O - (T_i + O_{i,k} - O) \mod T_i\} \tag{12}$$

## 4. FORMAL PROOFS IN COQ

We have formalized in Coq the system model described in Section 2 and proved the correctness of the analysis presented in Section 3. Coq is acknowledged as a very reliable (and rich) interactive environment to develop and verify proofs. To be convinced, an external reader only has to understand the problem specification and the main correctness theorem. Coq also allows the extraction of certified tools. Our development made use of the Prosa library where FPP, busy windows, workload and associated properties are mostly already defined (but Equation (3), for example, was not available). However, periodic tasks with offsets and related analyses are not considered. Our work will be used to enrich Prosa.

The interested reader can access the proofs sources at https://team.inria.fr/spades/coq-proofs-offsets. The Coq code follows the content of the previous sections.

Compared to Tindell's informal proof, we had to prove in Lemma 1 that the worst-case response time is achieved when all execution times are equal to their worst-case execution time — this property is not even mentioned in [6]. Also, Tindell does not prove that the worst case happens within a busy window starting with an activation of each transaction. This led us to prove several nontrivial auxiliary properties about fixed point iterations.

The proof, as it is presented in Section 3 and formalized in Coq, looks quite different from the original paper. We used $\phi$, $BW$ and $\theta$ instead of $W$, $w$ and $\hat{O}$ respectively (these notations are used in other offset-related papers). The main result, however, is of course very similar.

## 5. CONCLUSION AND FUTURE WORK

In this short paper, we presented the formalization of a response time analysis for tasks with offsets and its correctness proof using Coq. We defined and proved generic lemmas that can be reused in other proofs. Some hypotheses can be removed to generalize our results. For example, the implicit deadline assumption is not used, while the restricted offsets assumption does not seem to be critical in our proofs. We will investigate some extensions and expect their correctness proof to be very lightweight, as we can build on top of the current proof.

Clearly, this work is in progress. The next steps are:

1. Use the Coq extraction feature to generate a certified schedulability analysis tool (in OCaml). This step should not involve much work.
2. Design a Coq certifier for a standard tool (e.g., pyCPA). The certifier should ensure the correctness of results of an uncertified analyzer. This task is more involved and could require further research (e.g., the analyzer might need to be instrumented to produce additional information for the certifier).
3. Compare the runtimes and effort required by the certified analyzer and the certifier. Checking a fixed point is faster than computing it: the respective efficiency of both tools should depend on how much time is spent in fixed point computations.

## 6. REFERENCES

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

[2] NETCAR-Analyzer: the RTaW-Sim plugin for worst-case timing analysis on Controller Area Network. http://www.realtimeatwork.com/software/netcar-analyzer/.

[3] SymTA/S: Model-based timing analysis and optimization. https://auto.luxoft.com/uth/timing-analysis-tools/.

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

[5] 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.

[6] 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.

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