FROM INTUITION TO COQ
A CASE STUDY IN VERIFIED RESPONSE-TIME ANALYSIS
OF FIFO SCHEDULING

RTSS’ 22
7 December 2022

Kimaya Bedarkar, Mariam Vardishvili,
Sergey Bozhko, Marco Maida, Björn Brandenburg
PAPER IN A NUTSHELL

CASE STUDY

A formally verified response-time analysis (RTA) for FIFO

→ Formal verification ensures correctness
→ How much effort does it take to formally verify a result?
→ Can RTS researchers with limited Coq know-how do it?

<table>
<thead>
<tr>
<th>Variable</th>
<th>R : duration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis</td>
<td>H_R_max:</td>
</tr>
<tr>
<td></td>
<td>∀ (A : duration),</td>
</tr>
<tr>
<td></td>
<td>is_in_concrete_search_space A →</td>
</tr>
<tr>
<td></td>
<td>∃ (F : nat),</td>
</tr>
<tr>
<td></td>
<td>A + F ≥ \sum_(tsk &lt;- ts) RBF tsk (A + ε)</td>
</tr>
<tr>
<td></td>
<td>∧ F ≤ R.</td>
</tr>
</tbody>
</table>

| Theorem | uniprocessor_response_time_bound_FIFO: |
|         | task_response_time_bound tsk R. |

EMPIRICAL EXPLORATION

Why FIFO?

→ Trivial to implement
→ Low run-time overhead
→ Surprisingly little prior attention
→ Good enough for certain workloads
MOTIVATION
WHY FORMAL VERIFICATION?

The field of real-time systems aims to give strong guarantees

➔Traditionally backed by pen & paper proofs

Pen & paper analyses are not immune to bugs!
WHY FORMAL VERIFICATION?

The field of real-time systems aims to give strong guarantees

➔Traditionally backed by pen & paper proofs

Mechanized proofs protect us from mistakes!
BUT, ISN’T FORMAL VERIFICATION REALLY HARD?

Prior work has used formal verification to prove:

- EDF, FP RTA (Bozhko and Brandenburg, 2020)
- Results in network calculus (Roux et al., 2022)
- etc.

How much **effort** does it take?  
How much **prior knowledge** does it take?

Case study: 
Verification of an RTA
OVERVIEW OF CASE STUDY

System Model

- Modeling Constructs
- Validity Constraints

FIFO RTA

- Policy Description
- IBF
- Search Space

Final Response-Time Bound

Each element corresponds to some Coq code!
**SETUP: SYSTEM MODEL**

System Model

- Ideal uni-processor
- Set of $n$ sporadic, independent real-time tasks
- Arbitrary deadlines
- Worst-case execution time
- Arbitrary arrival curves

- Modeling Constructs
- Validity Constraints
BACKGROUND
BACKGROUND: COQ

Coq is a **proof assistant**

→ You can write **programs/definitions** and then **prove theorems** about them

→ The **proof engine** is not fully automatic!

---

**Theorem** a_simple_theorem:  
\[ \forall x y, \quad x + y = y + x. \]

**Proof.**

move \( \to x y. \)

induction \( x. \)

- by rewrite add0n addn0. (* base *)
- by rewrite addSn IHx addnS. (* step *)

Qed.
BACKGROUND: PROSA

Prosa is a Coq library of definitions and proofs about RTS

→ Basic definitions (jobs, tasks, processor, etc.)
→ Proofs of classic results as well as novel ones

Prosa emphasizes readable specifications

**Higher- and Equal-Priority Interference**

Next, we establish a bound on the interference produced by higher- and equal-priority jobs.

*Section* BoundOnHEPWorkload.

Consider again a job $j$ of the task under analysis $t_{sk}$ with a positive cost.

- **Variable** $j$ : Job.
- **Hypothesis** $H_{job\_of\_task}$ : $job\_of\_task\ tsk\ j$.
- **Hypothesis** $H_{j\_in\_arrivals}$ : $arrives\_in\ arr\_seq\ j$.
- **Hypothesis** $H_{job\_cost\_positive}$ : $job\_cost\_positive\ j$.

https://prosa.mpi-sws.org/
CASE STUDY: FIFO RTA
INTUITIVE VS. FORMAL REASONING

Intuitive definitions and results *usually* have a natural mechanized counterpart.

**Natural Language**

Work conservation: If a job $j$ is backlogged at time $t$, then some other job $j_{other}$ is scheduled at $t$.

**Gallina (Coq)**

Definition `work_conserving` :=
\[
\forall \, j \, t, \\
\text{backlogged} \, j \, t \rightarrow \\
\exists \, j_{\text{other}}, \\
\text{scheduled_at} \, j_{\text{other}} \, t.
\]
OVERVIEW OF CASE STUDY

System Model

Modeling Constructs

Validity Constraints

FIFO RTA

Policy Description

IBF

Search Space

Final Response-Time Bound

Each element corresponds to some Coq code!
OVERVIEW OF CASE STUDY

System Model

- Modeling Constructs
- Validity Constraints

FIFO RTA

- Policy Description
- IBF
- Search Space
- Final Response-Time Bound

Each element corresponds to some Coq code!
SYSTEM MODEL: WORKLOAD

We employ a discrete time model, and let $T = \mathbb{N}$ denote the time domain and $\varepsilon \triangleq 1$ the indivisible least unit of time.

Definition duration := nat.
Definition $\varepsilon := 1.$
We employ a discrete time model, and let $T = \mathbb{N}$ denote the time domain and $\varepsilon \triangleq 1$ the indivisible least unit of time. The workload is a set of $n$ sporadic real-time tasks $\tau \triangleq \{\tau_1, \tau_2, \ldots, \tau_n\}$.

**Definition** $\text{duration} := \text{nat}$.  
**Definition** $\varepsilon := 1$.  
**Context** $\{\text{Task : TaskType}\}$.  
**Variable** $\text{ts} : \text{seq Task}$.  

**SYSTEM MODEL: WORKLOAD**
We employ a discrete time model, and let $\mathbb{T} = \mathbb{N}$ denote the time domain and $\varepsilon \triangleq 1$ the indivisible least unit of time.

The workload is a set of $n$ sporadic real-time tasks $\tau \triangleq \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task $\tau_i \triangleq (C_i, D_i, \alpha_i)$ has a worst-case execution time $C_i$, a relative deadline $D_i$, and an arrival-bound function $\alpha_i(\Delta)$. The role of $\alpha_i(\Delta)$ is to upper-bound the number of activations of $\tau_i$ in any time window of length $\Delta$.

**Definition** duration := nat.
**Definition** $\varepsilon := 1$.

**Context** \{Task : TaskType\}.
**Variable** ts : seq Task.

**Context** \{TaskCost Task\}.
**Context** \{MaxArrivals Task\}.
**Context** \{TaskDeadline Task\}.
SYSTEM MODEL: WORKLOAD

We employ a discrete time model, and let \( T = \mathbb{N} \) denote
the time domain and \( \varepsilon \triangleq 1 \) the indivisible least unit of time.

The workload is a set of \( n \) sporadic real-time tasks \( \tau \triangleq \{\tau_1, \tau_2, \ldots, \tau_n\} \). Each task \( \tau_i \triangleq (C_i, D_i, \alpha_i) \) has a worst-case
execution time \( C_i \), a relative deadline \( D_i \), and an arrival-bound function \( \alpha_i(\Delta) \). The role of \( \alpha_i(\Delta) \) is to upper-bound
the number of activations of \( \tau_i \) in any time window of length \( \Delta \).

\[
\text{Definition} \quad \text{duration} := \text{nat}.
\]
\[
\text{Definition} \quad \varepsilon := 1.
\]

\[
\text{Class} \quad \text{MaxArrivals} (\text{Task} : \text{TaskType}) := \\
\quad \text{max_arrivals} : \text{Task} \rightarrow \text{duration} \rightarrow \text{nat}.
\]
OVERVIEW OF CASE STUDY

System Model

- Modeling Constructs
- Validity Constraints

FIFO RTA

- Policy Description
- IBF
- Search Space
- Final Response-Time Bound

Each element corresponds to some Coq code!
**SYSTEM MODEL: VALIDITY CONSTRAINTS**

We employ a discrete time model, and let $\mathbb{T} = \mathbb{N}$ denote the time domain and $\varepsilon \triangleq 1$ the indivisible least unit of time.

The workload is a set of $n$ sporadic real-time tasks $\tau \triangleq \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task $\tau_i \triangleq (C_i, D_i, \alpha_i)$ has a worst-case execution time $C_i$, a relative deadline $D_i$, and an arrival-bound function $\alpha_i(\Delta)$. The role of $\alpha_i(\Delta)$ is to upper-bound the number of activations of $\tau_i$ in any time window of length $\Delta$.

**Mathematical Language**

\[ J_{i,j} := j^{th} \text{ job of } i^{th} \text{ task} \]

\[ \forall t, \forall \Delta, \left| \{ J_{i,j} \mid t \leq a_{i,j} < t + \Delta \} \right| \leq \alpha_i(\Delta) \]

**Gallina (Coq)**

\[ \text{Definition} \quad \text{respects_max_arrivals} := \]

\[ \forall (t1 \ t2 : \text{instant}) (\text{tsk} : \text{Task}), \]

\[ \text{t1} \leq \text{t2} \rightarrow \]

\[ \#\text{task_arrivals arr_seq tsk t1 t2} \leq \text{max_arrivals tsk (t2 - t1)}. \]
OVERVIEW OF CASE STUDY

System Model
- Modeling Constructs
- Validity Constraints

FIFO RTA
- Policy Description
- IBF
- Search Space
- Final Response-Time Bound

Each element corresponds to some Coq code!
**ANALYSIS: INTERFERENCE BOUND FUNCTION**

Our RTA applies the **busy-window principle**

→ Cumulative interference incurred within the busy window of job \( \leq \) Interference Bound Function (IBF).

\[
RBF_i(\Delta) = C_i \times \alpha_i(\Delta)
\]

**Let**

\[
IBF\ tsk_i\ (A : \text{duration}) := \left(\sum_{\substack{tsk_k \leftarrow ts}} RBF\ tsk_k\ (A + \varepsilon)\right) - \text{task\_cost \ tsk\_i}.
\]

**Mathematical Language**

\[
IBF(A) = \left(\sum_{\tau_k \in \tau} RBF_k(A + \varepsilon)\right) - C_i
\]

**Gallina (Coq)**

\[
IBF(A) = c_{i,j} = C_i
\]
OVERVIEW OF CASE STUDY

System Model

- Modeling Constructs
- Validity Constraints

FIFO RTA

- Policy Description
- IBF
- Search Space

Final Response-Time Bound

Each element corresponds to some Coq code!
The final response-time bound is stated as a fixed point

\[ \forall A \in A, \exists F, A + F = \sum_{\tau_k \in \tau} RBF_k(A + \varepsilon) \land F \leq R. \quad (5) \]

**Theorem 1.** If a finite bound \( L \) on the maximum busy-window length exists, then any job \( J_{i, j} \) of any given task \( \tau_i \in \tau \) will finish execution by time \( a_{i, j} + R \).

**Mathematical Language**

**Variable** \( R : \text{duration} \).

**Hypothesis** \( H_{R \text{ max}}: \)

\[ \forall (A : \text{duration}), \exists (F : \text{duration}),
A + F \geq \sum_{(tsk_k \leftarrow ts)} RBF \ tsk_k (A + \varepsilon) \land F \leq R. \]

**Gallina (Coq)**

\[ \text{Theorem uniprocessor_response_time_bound_FIFO:} \]

\[ \forall j, \text{job_of_task tsk j} \rightarrow \text{completed_by j (job_arrival j + R)}. \]
WHAT DID IT TAKE?

**Proof effort:** \(\approx 3\) months

- One person with limited prior Coq experience
- And limited RTS experience

**Proof artifact**

- Proof artifact has since been modified
- Comments and structure aiding accessibility
- Artifact with proof and profusely commented specs:
  
  https://people.mpi-sws.org/~bbb/papers/details/rtss22/

**Slightly more than 400 lines of code**

- Surprisingly low
- Made possible by building on existing Prosa definitions

<table>
<thead>
<tr>
<th>Total LOC</th>
<th>432</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifications</td>
<td>92</td>
</tr>
<tr>
<td>Proof scripts</td>
<td>132</td>
</tr>
<tr>
<td>Comments</td>
<td>208</td>
</tr>
</tbody>
</table>
EMPIRICAL EXPLORATION
Generated each task $\tau_i$ with:

- Period: non-uniform distribution over the set $\{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ ms
- Cost: Randomly generated using Kramer et al.’s tables
- For each cardinality $2 \leq i \leq 30$, 500 tasks were generated
BASELINE COMPARISON

How does our RTA compare with the baseline?

The Case for FIFO Real-Time Scheduling

Sebastian Altmeyer
University of Luxembourg
sebastian.altmeyer@uni.lu

Sakthivel Manikandan Sundharam
University of Luxembourg
sakthivel.sundharam@uni.lu

Nicolas Navet
University of Luxembourg
nicolas.navet@uni.lu

Abstract—Selecting the right scheduling policy is a crucial issue in the development of an embedded real-time application. Whereas scheduling policies are typically judged according to their ability to schedule task sets at a high percentage utilization, other concerns, such as predictability and simplicity, are often overlooked. In this paper, we argue that FIFO scheduling with offsets is a suitable choice when these concerns play a key role. To this end, we examine the predictability of FIFO, present a schedulability analysis for it, and evaluate both performance and predictability of FIFO scheduling with and without offsets. Our results show that FIFO with offsets exhibits competitive performance for task with regular periods, at an unmatched predictability.

Analysis
- Feasibility
- This paper (Theorem 1)
- Altmeyer et al. (2016)

Feasibility and proposed FIFO RTA curves overlap
CAN FIFO BE A VIABLE POLICY?

For which workloads can FIFO be a suitable choice?

Proposed RTA gives us a tool to test if we can get away with using FIFO

Ratio of response times of tasks in FP and FIFO schedules

These tasks perform better with FIFO than with FP

These tasks do not benefit from FIFO
CONCLUSION
Case study
➔ Similarity of formal and intuitive arguments
➔ Roadmap for formalizing RTS results

Empirical exploration
➔ Proposed RTA works for all feasible workloads
➔ FIFO scheduling beneficial for lower rate-tasks (at the expense of higher-rate tasks)

For a one-to-one mapping of pen and paper results to code, check out the Prosa webpage!

Library prosa.results.fifo.rta
- Response-Time Analysis for FIFO Schedulers
  - A. Defining the System Model
    - Tasks and Jobs
    - The Job Arrival Sequence
    - Absence of Self-Suspensions and WCET Compliance
    - The Task Set
    - The Task Under Analysis
    - The Schedule
  - B. Encoding the Scheduling Policy and Preemption Model
  - C. Classic and Abstract Work Conservation
  - D. Bounding the Maximum Busy-Window Length
  - E. Defining the Interference Bound Function (IBF)
    - Absence of Priority Inversion
    - Higher- and Equal-Priority Interference
    - Correctness of IBF
  - F. Defining the Search Space
  - G. Stating the Response-Time Bound R
  - H. Soundness of the Response-Time Bound

https://prosa.mpi-sws.org/