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

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS
European Research Council
Established by the European Commission
DFG Deutsche Forschungsgemeinschaft
German Research Foundation
MAX-PLANCK-GESELLSCHAFT

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?

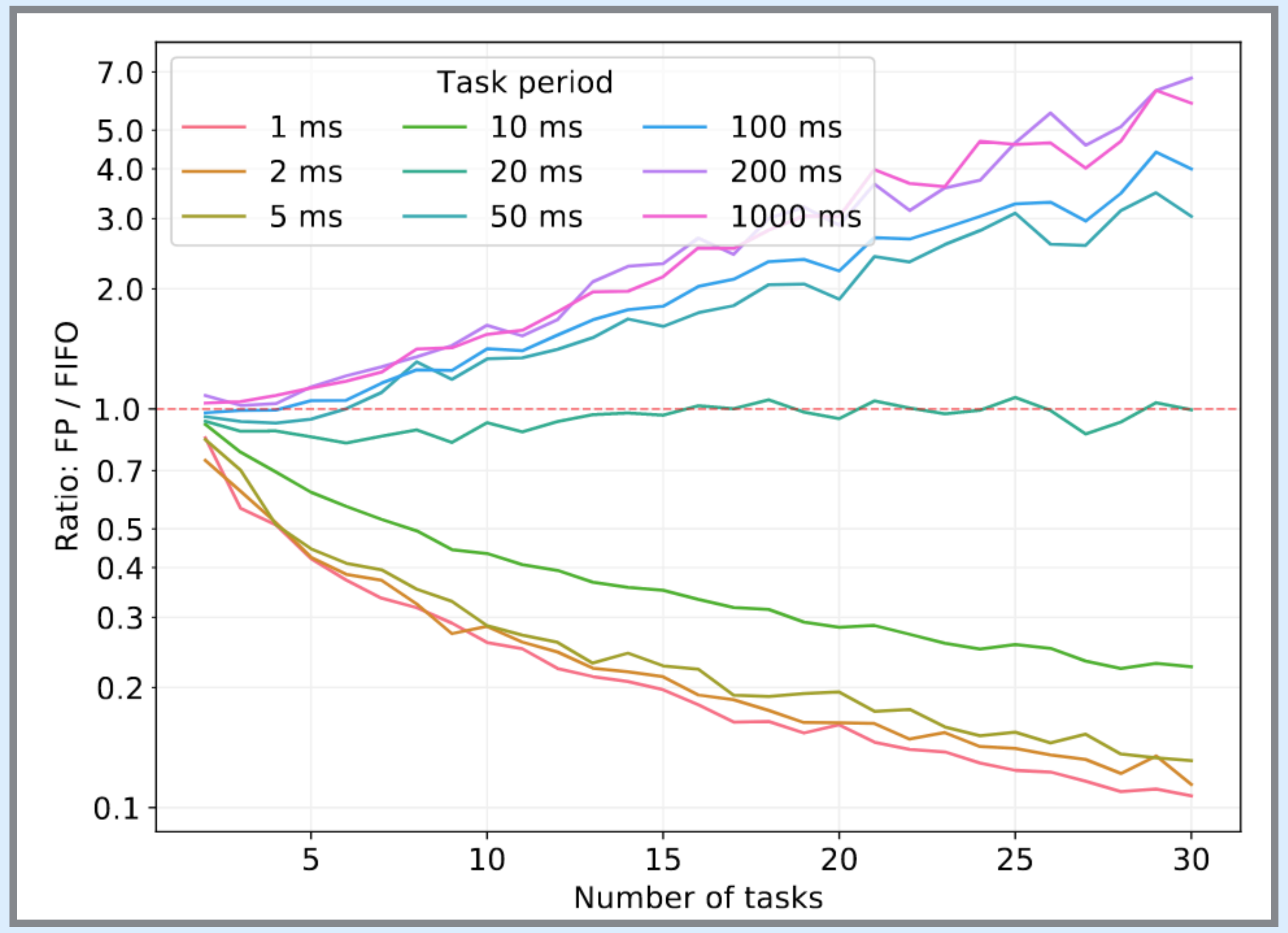
```
Variable R : duration.
Hypothesis H_R_max:
  ∀ (A : duration),
    is_in_concrete_search_space A →
      ∃ (F : nat),
        A + F ≥ \sum_(tsk <- ts) RBF tsk (A + ε)
        ∧ F ≤ R.

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

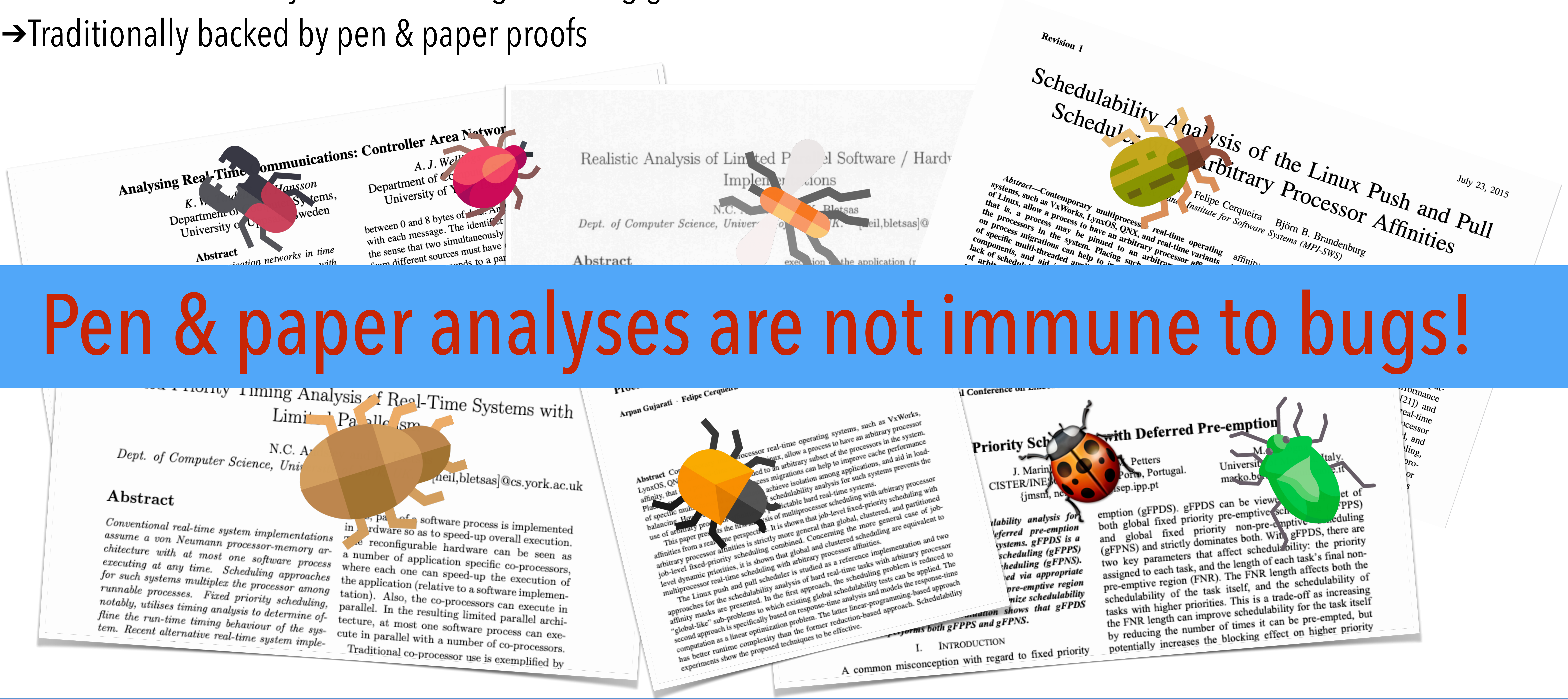


MOTIVATION

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:

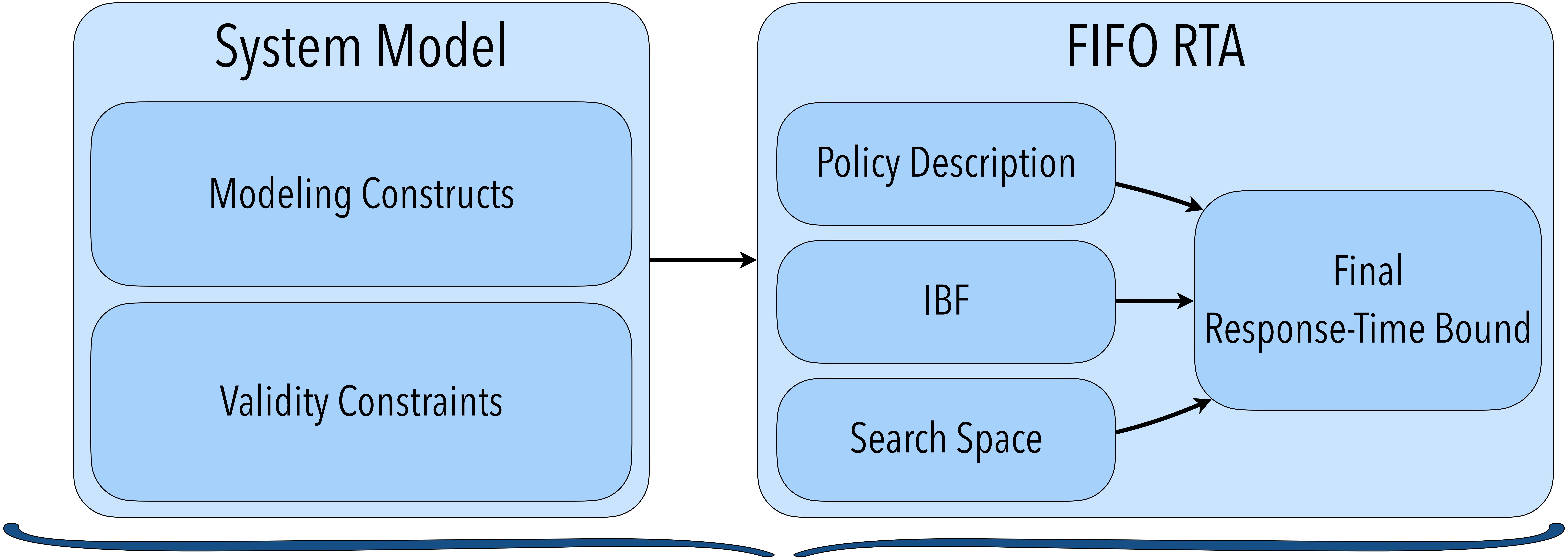
- EDF, FP RTA (Bozhko and Brandenburg, 2020)
- Results in network calculus (Roux *et al.*, 2022)
- *etc.*

How much **effort** does it take?

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

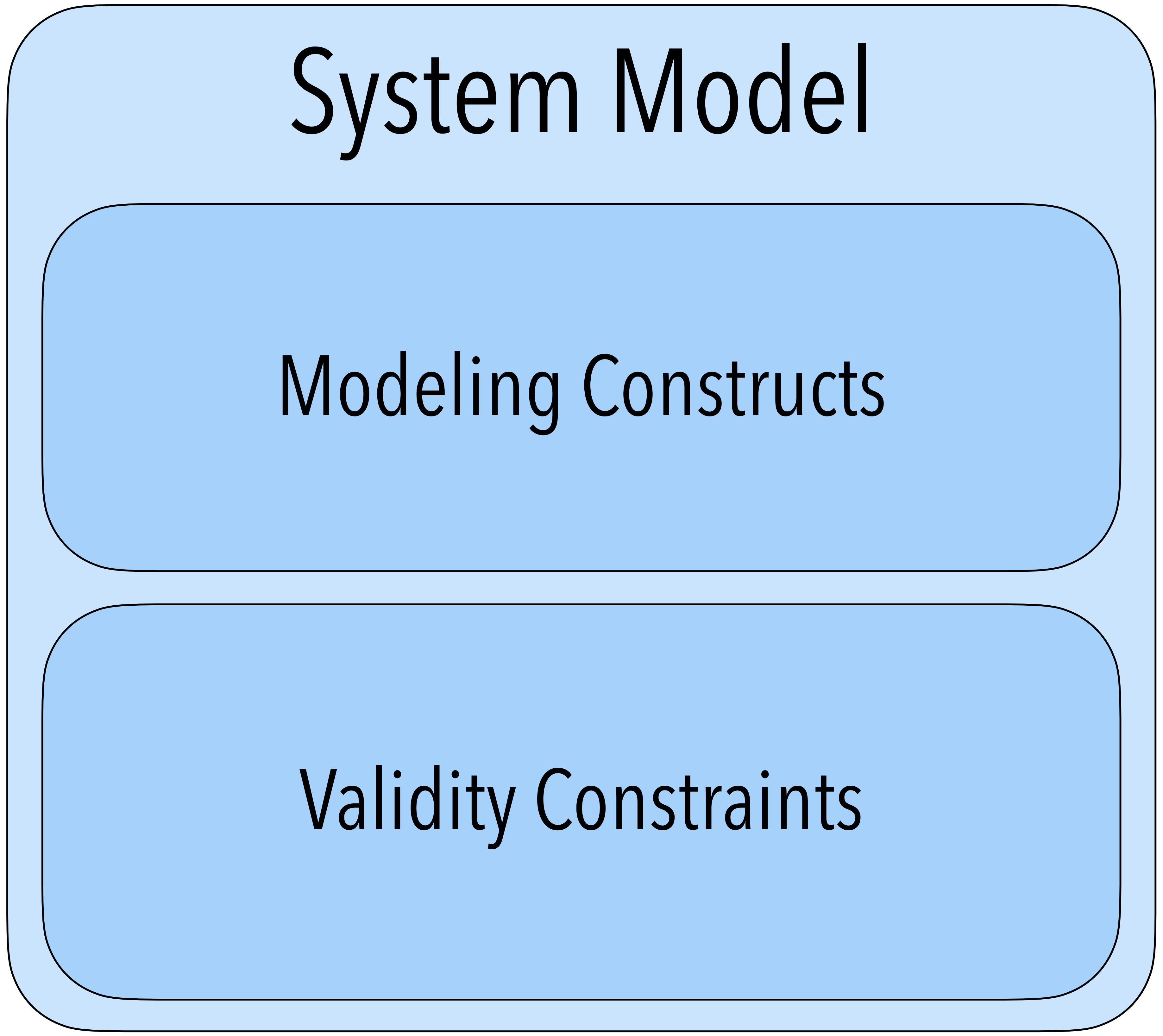
Case study:
Verification of an RTA

OVERVIEW OF CASE STUDY



Each element corresponds to some Coq code!

SETUP: SYSTEM MODEL



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

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:
  ∀ x y,
    x + y = y + x.
Proof.
  move → 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

formally proven schedulability analysis | PROSA

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

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 `tsk` 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.
```

CASE STUDY: FIFO RTA

INTUITIVE VS. FORMAL REASONING

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

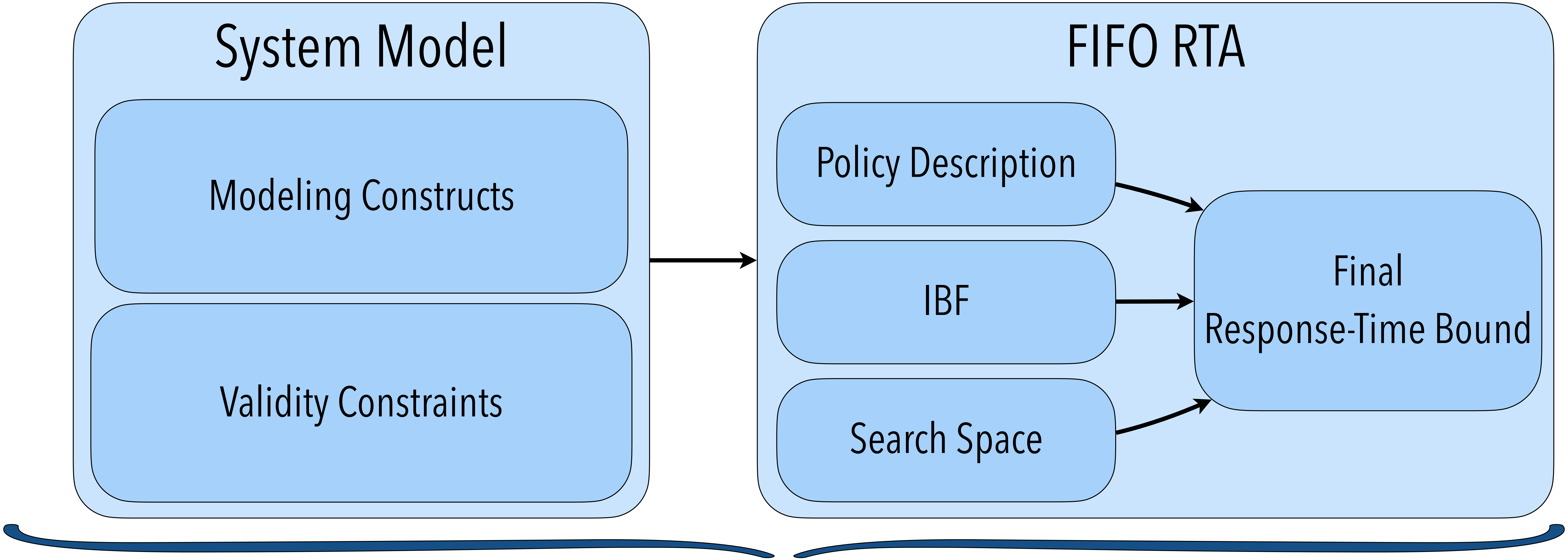
Work conservation: If a job j is backlogged at time t, then some other job j_{other} is scheduled at t.

Natural Language

```
Definition work_conserving :=
  ∀ j t,
    backlogged j t →
    ∃ j_other,
      scheduled_at j_other t.
```

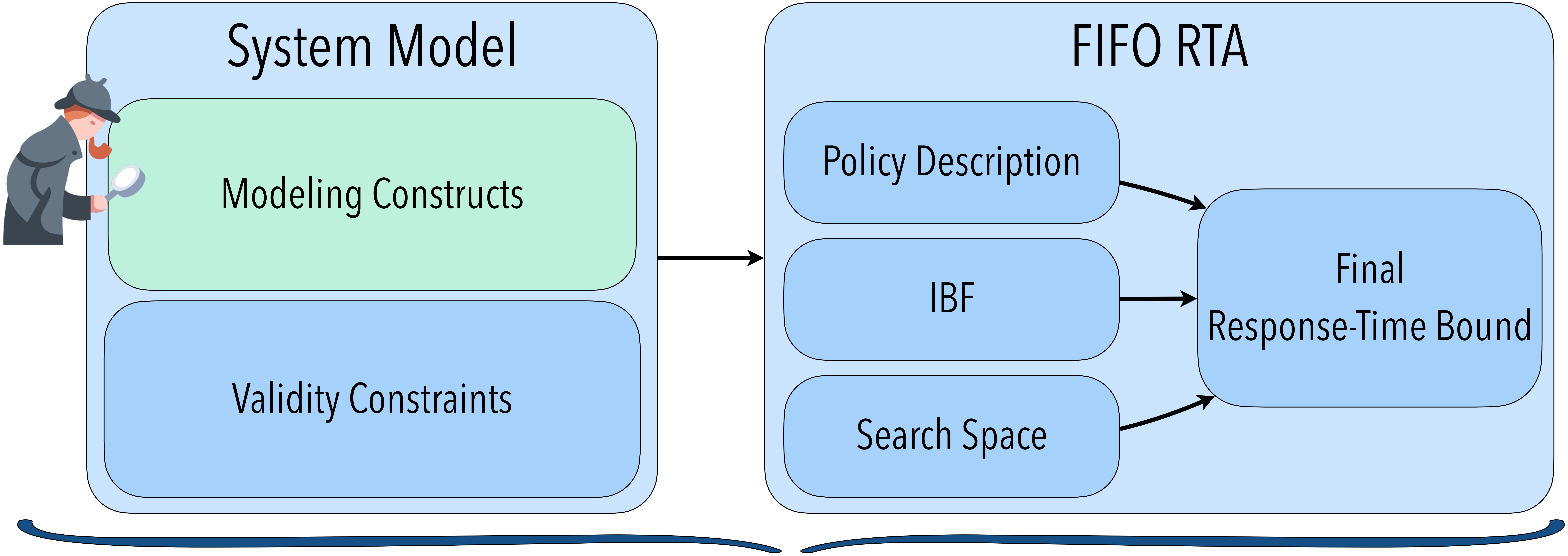
Gallina (Coq)

OVERVIEW OF CASE STUDY



Each element corresponds to some Coq code!

OVERVIEW OF CASE STUDY



Each element corresponds to some Coq code!

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.

```
Definition duration := nat.
Definition ε := 1.
```

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 $\tau \triangleq \{\tau_1, \tau_2, ..., \tau_n\}$.

```
Definition duration := nat.
Definition ε := 1.
```

```
Context {Task : TaskType}.
Variable ts : 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 τ_i in any time window of length Δ.

```
Definition duration := nat.
Definition ε := 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 $\mathbb{T} = \mathbb{N}$ 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 $\tau \triangleq \{\tau_1, \tau_2, ..., \tau_n\}$. 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 τ_i in any time window of length Δ.

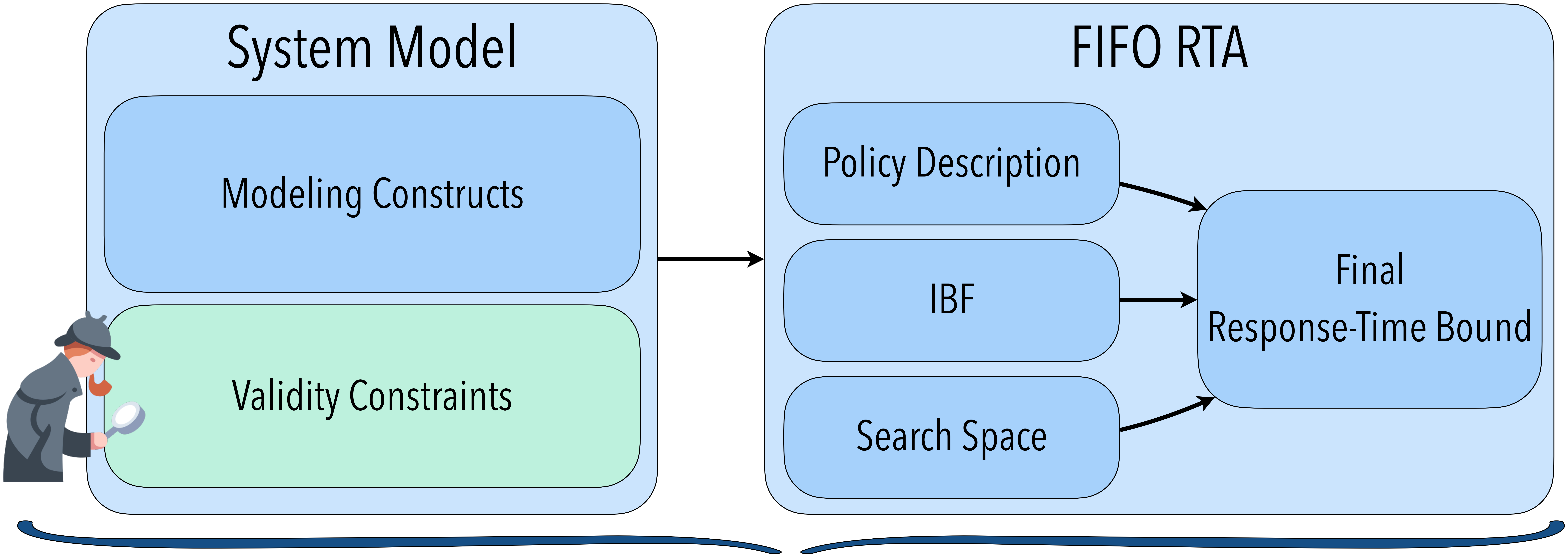
```
Definition duration := nat.
Definition ε := 1.
```

```
Context {Task : TaskType}.
Variable ts : seq Task.
```

```
Context `{TaskCost Task}.
Context `{MaxArrivals Task}.
Context `{TaskDeadline Task}.
```

```
Class MaxArrivals (Task : TaskType) :=
  max_arrivals : Task → duration → nat.
```

OVERVIEW OF CASE STUDY



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 $\tau \triangleq \{\tau_1, \tau_2, ..., \tau_n\}$. 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 τ_i in any time window of length Δ.

$J_{i,j}$:= j^{th} job of i^{th} task

arrival time of $J_{i,j}$

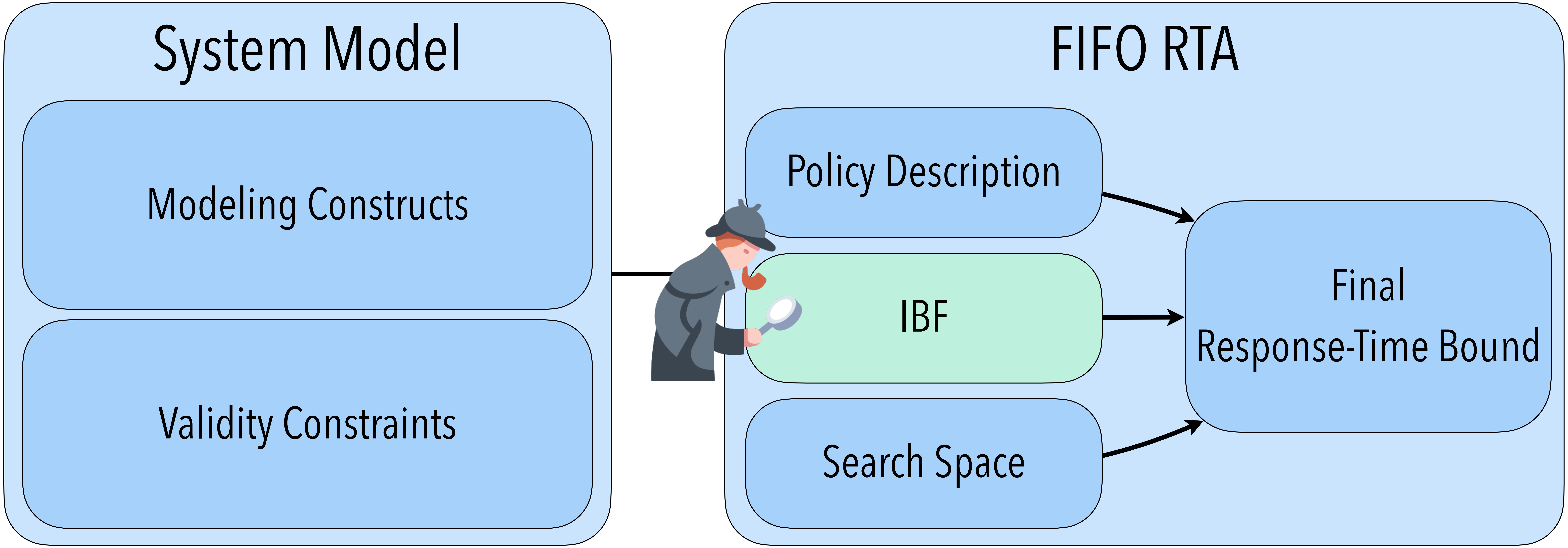
$$\forall t, \forall \Delta, \ |\{J_{i,j} \mid t \leq a_{i,j} < t + \Delta\}| \leq \alpha_i(\Delta)$$

Mathematical Language

```
Definition respects_max_arrivals :=
    ∀ (t1 t2 : instant) (tsk : Task),
      t1 ≤ t2 →
        #|task_arrivals arr_seq tsk t1 t2|
          ≤ max_arrivals tsk (t2 - t1).
```

Gallina (Coq)

OVERVIEW OF CASE STUDY



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 ≤ Interference Bound Function (IBF).

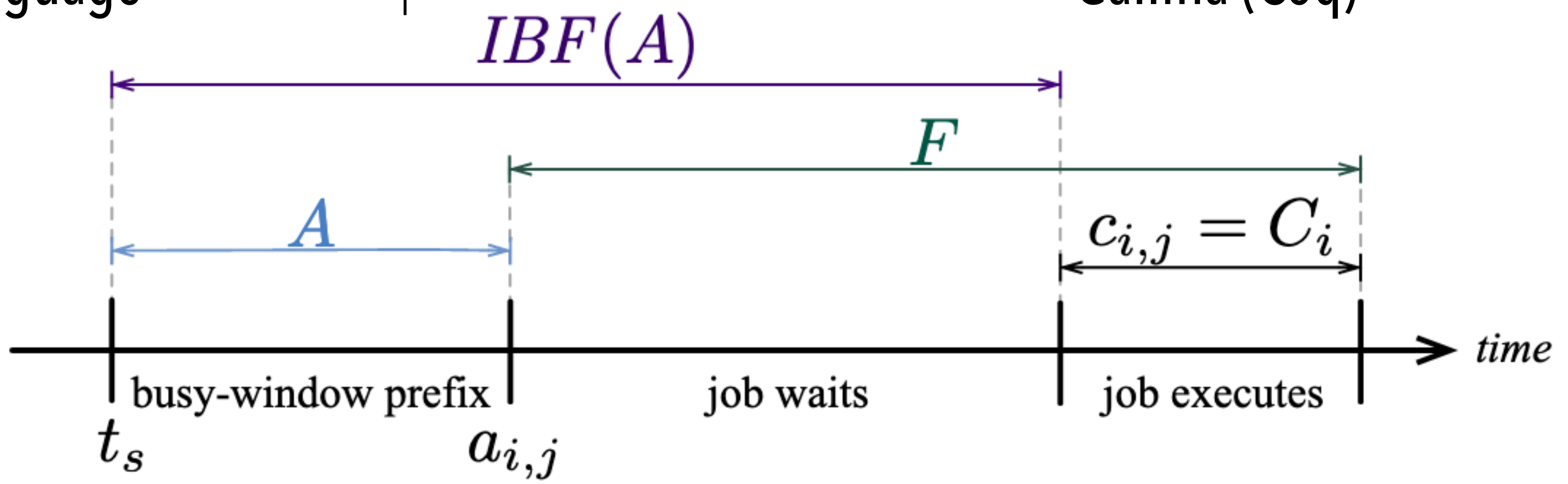
$$RBF_i(\Delta) = C_i \times \alpha_i(\Delta)$$

$$IBF(A) = \Big(\sum_{\tau_k \in \tau} RBF_k(A + \varepsilon)\Big) - C_i$$

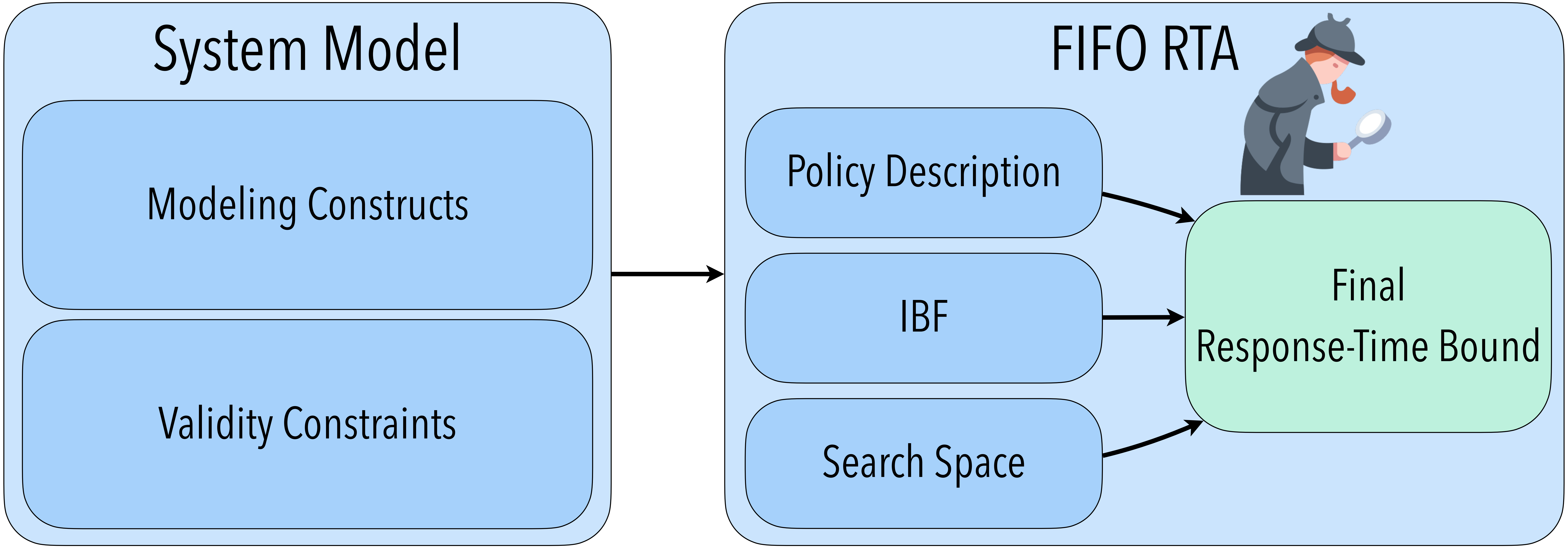
Mathematical Language

```
Let IBF tsk_i (A : duration) :=
  (\sum_(tsk_k <- ts) RBF tsk_k (A + ε))
    - task_cost tsk_i.
```

Gallina (Coq)



OVERVIEW OF CASE STUDY



Each element corresponds to some Coq code!

ANALYSIS: FINAL RESPONSE-TIME BOUND

The final response-time bound is stated as a fixed point

Let R denote the search result, *i.e.*, the least positive value s.th.

$$\forall A \in \mathcal{A}, \; \exists F, \; A + F = \sum_{\tau_k \in \tau} RBF_k(A + \varepsilon) \; \wedge \; F \leq R. \quad (5)$$

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 : duration.
Hypothesis H_R_max:
  ∀ (A : duration),
    is_in_concrete_search_space A →
    ∃ (F : duration),
      A + F ≥ \sum_(tsk_k <- ts) RBF tsk_k (A + ε)
      ∧ F ≤ R.

Theorem uniprocessor_response_time_bound_FIFO:
  ∀ j, job_of_task tsk j →
    completed_by j (job_arrival j + R).
```

Gallina (Coq)

WHAT DID IT TAKE?

Proof effort: ≈ 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

Total LOC	432
Specifications	92
Proof scripts	132
Comments	208

EMPIRICAL EXPLORATION

SETUP

Real World Automotive Benchmarks For Free

Simon Kramer, Dirk Ziegenbein, Arne Hamann
Corporate Research
Robert Bosch GmbH
Renningen, Germany
{simon.kramer2|dirk.ziegenbein|arne.hamann}@de.bosch.com

The progress and comparability of real-time analysis methods that are applicable to real-world is slowed by the absence of realistic benchmarks, mainly due to intellectual property (IP) concerns. We propose a method that supports the generation of realistic but IP free benchmark sets. Further, we provide the application characteristics of a specific real-world automotive software system.

***Keywords*—benchmarks, timing analysis, automotive software**

unsatisfactory given the potential for front loading with formal analysis techniques.

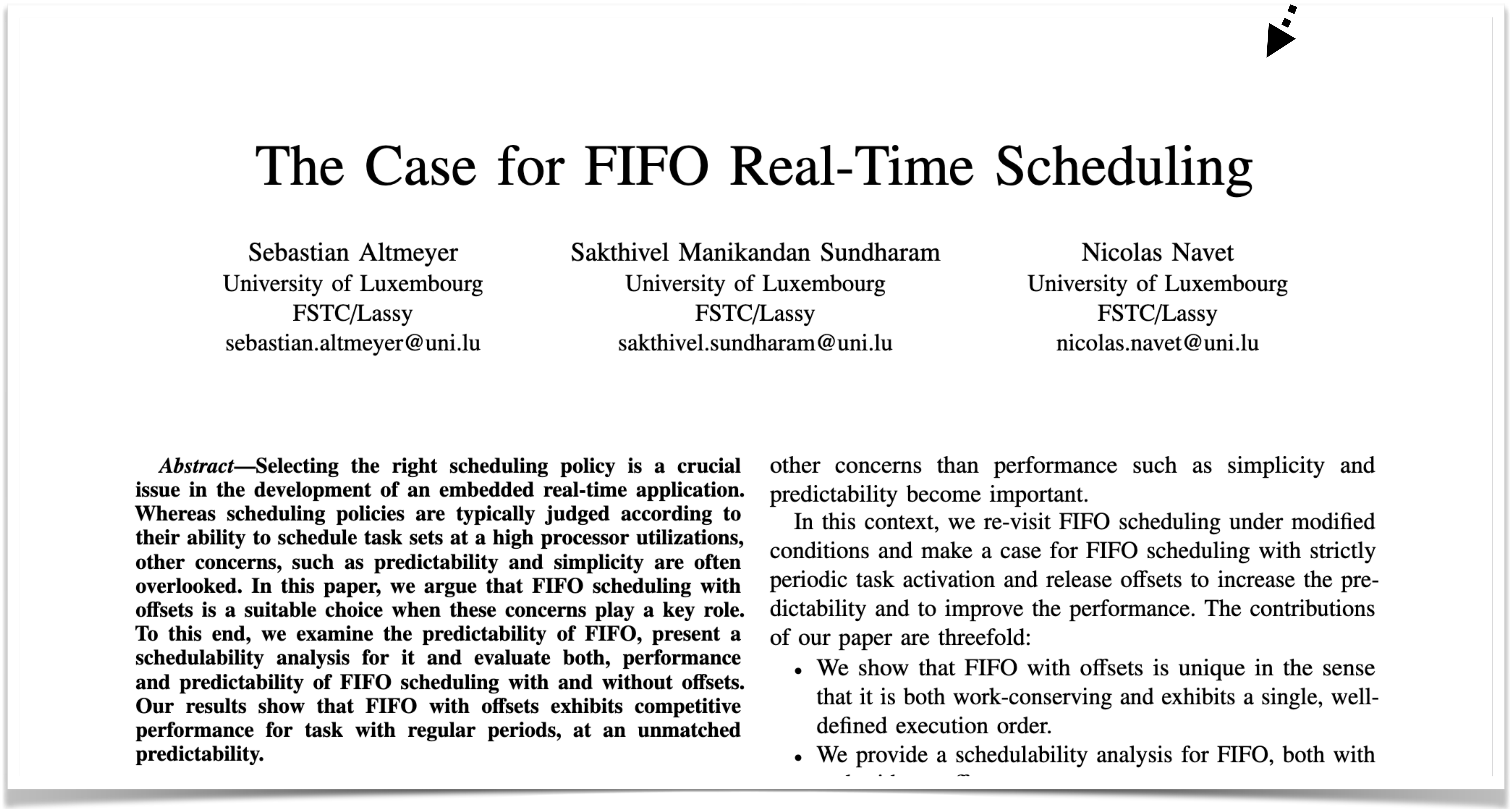
Due to the introduction of multi-core execution platforms, the risk of divergence between academic research and industrial practice is currently increasing. The reason is the strongly increased problem space for timing analysis induced by multi-core systems.

Extending existing approaches is very challenging since

- Generated each task τ_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 $\in \{2,3,\ldots,30\}$, 500 tasks were generated

BASELINE COMPARISON

How does our RTA compare with the <u>baseline</u>?



The Case for FIFO Real-Time Scheduling

Sebastian Altmeyer
University of Luxembourg
FSTC/Lassy
sebastian.altmeyer@uni.lu

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

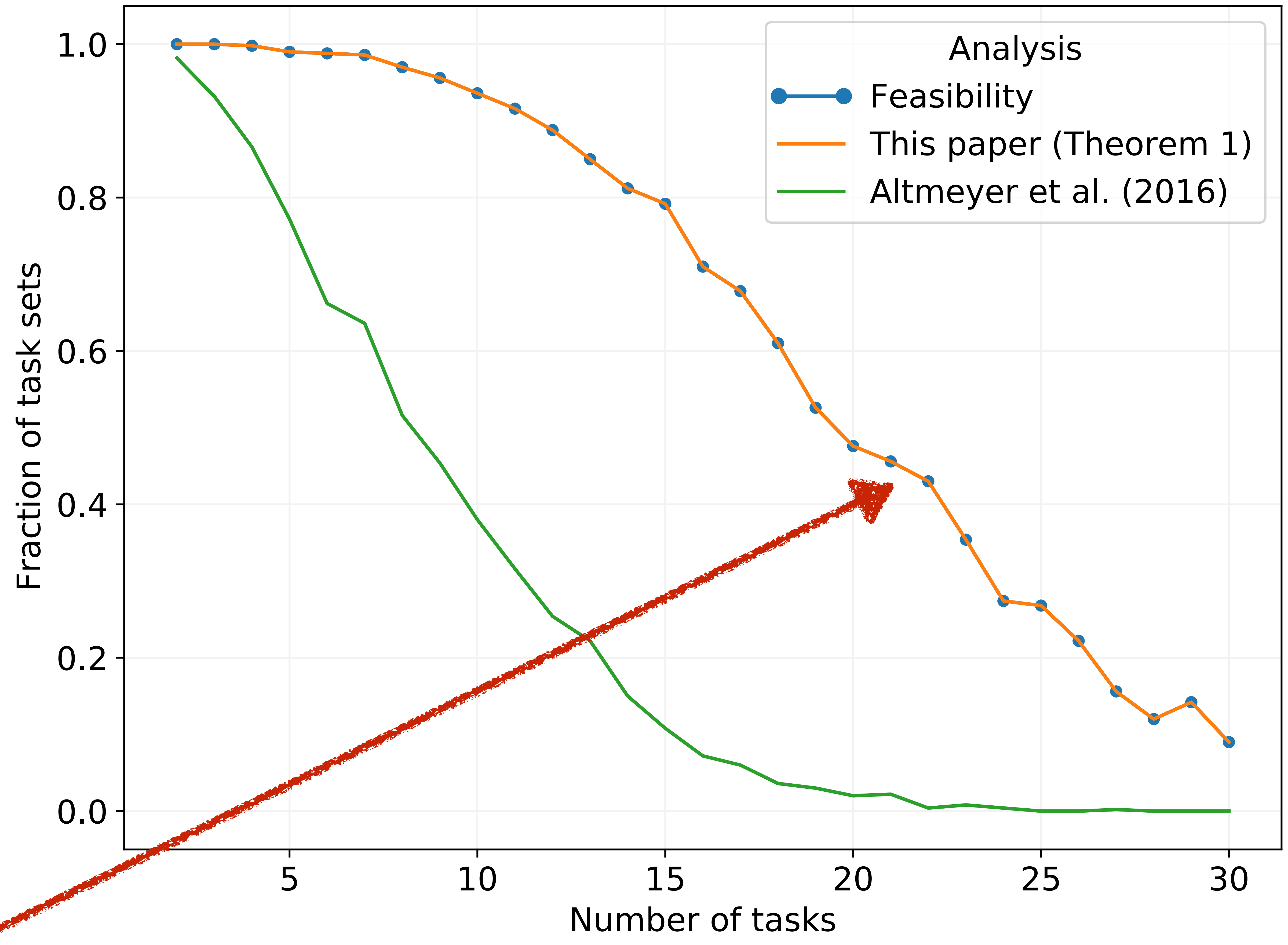
Nicolas Navet
University of Luxembourg
FSTC/Lassy
nicolas.navet@uni.lu

***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 processor utilizations, 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.**

other concerns than performance such as simplicity and predictability become important.

In this context, we re-visit FIFO scheduling under modified conditions and make a case for FIFO scheduling with strictly periodic task activation and release offsets to increase the predictability and to improve the performance. The contributions of our paper are threefold:

- We show that FIFO with offsets is unique in the sense that it is both work-conserving and exhibits a single, well-defined execution order.
- We provide a schedulability analysis for FIFO, both with



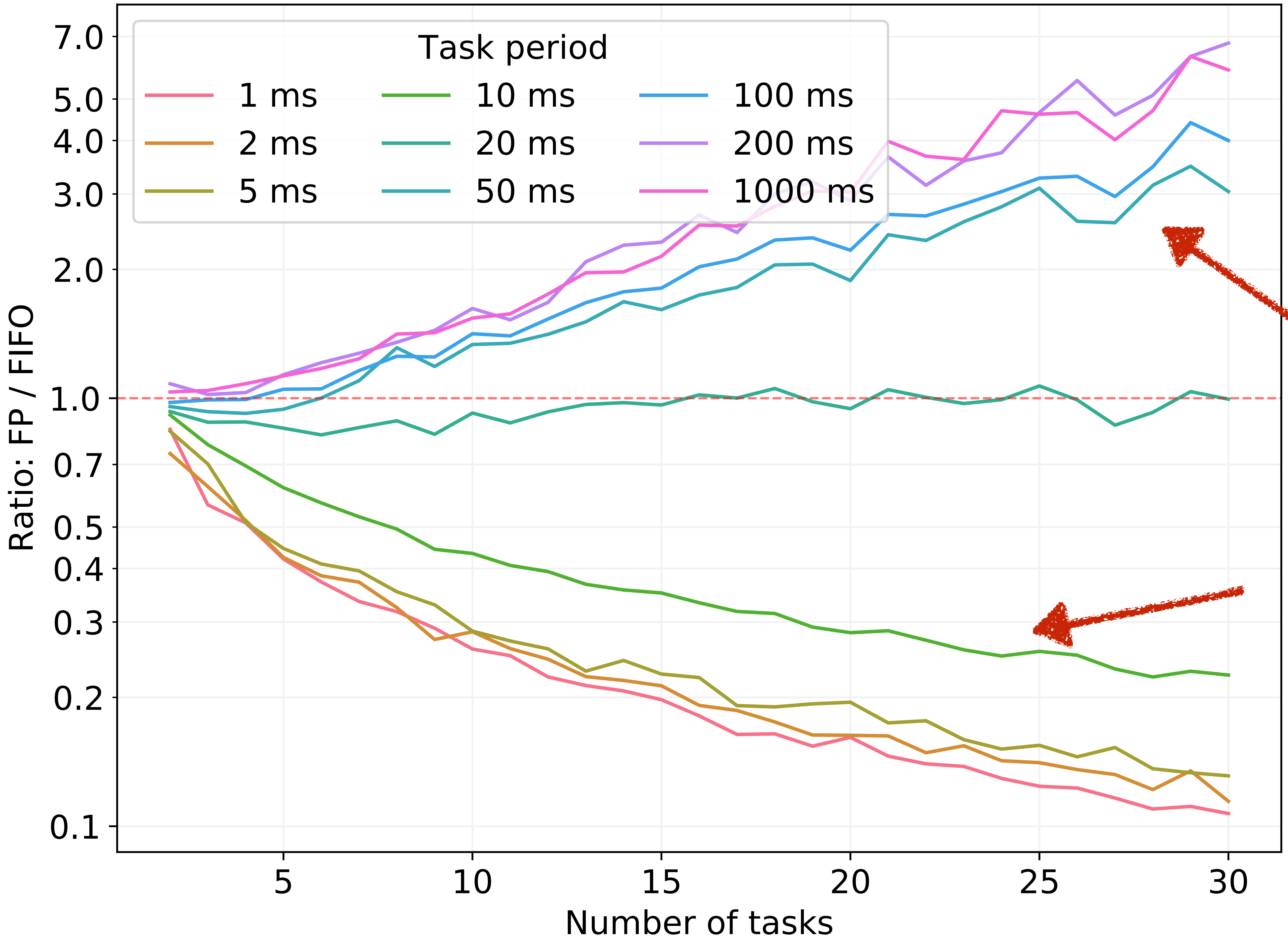
Feasibility and proposed FIFO RTA curves overlap

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

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)

https://people.mpi-sws.org/~bbb/papers/details/rtss22/

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

formally proven schedulability analysis | PROSA

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

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