Formal Correctness Proof for an EDF Scheduler Implementation

Florian Vanhems – florian.vanhems@univ-lille.fr

19th July, 2022
1 Overview

2 Proofs & Hypotheses

3 Models & Implementation
Motivations

○ Critical software of embedded systems
○ EDF is optimal and well understood
○ Experiment with our proof methodology
Aspirations

- Proof of concept - not a production ready scheduler
- Share our conception and understanding of software proofs
Overview
First formally proven implementation of an Earliest Deadline First scheduler for arbitrary sequences of jobs
Overview of the scheduler

Back-end

Election Function

Interface

State
Star of the talk

Election Function

Written directly in Coq
Formally proven properties
Translated word to word to C
Star of the talk

Election Function

Written directly in Coq
Star of the talk

Election Function

Written directly in Coq
Formally proven properties
Star of the talk

Election Function

Written directly in Coq  Formally proven properties  Translated word to word to C
General informations about the scheduler

- Schedules arbitrary sequences of jobs (as opposed to tasks)
- Periodically called
- Online

\(^1\): while the election function schedules online, our scheduler feeds it hard coded jobs for simplicity
Election function of an Earliest Deadline First scheduler

Job $a$

- Release date ($r_a$): $t = 7$
- Deadline ($d_a$): $t = 3$
- Execution period: $t = 6$
- Budget ($c_a$): $t = 3$

Election function:
1. $t = 1$: $\{\}$
2. $t = 2$: $\{a\}$
3. $t = 3$: $\{b\}$
4. $t = 4$: $\{a\}$
5. $t = 5$: $\{b\}$
6. $t = 6$: $\{a\}$
7. $t = 7$: $\{b\}$
Election function of an Earliest Deadline First scheduler

\[ \text{election function}(1) = \emptyset \]

\[ \text{election function}(2) = \text{Job}_a \]

\[ \text{election function}(3) = \text{Job}_b \]

\[ \text{election function}(4) = \text{Job}_a \]

budget \( (c_a) \)
Election function of an Earliest Deadline First scheduler

- Job $a$ with $t = 2$
- Release date $r_a$ at $t = 2$
- Budget $c_a$ at $t = 3$
- Deadline $d_a$ at $t = 6$
- Job's execution period from $t = 3$ to $t = 6$
- Election function $E(1) = \emptyset$
- Election function $E(2) = \{a\}$
- Election function $E(3) = \{b\}$
- Election function $E(4) = \{a\}$

Florian Vanhems – florian.vanhems@univ-lille.fr

Page 10 of 26
Election function of an Earliest Deadline First scheduler

- **Job** \(a\)
  - Release date: \(t = 2\)
  - Deadline: \(t = 7\)
  - Budget: \(c_a\)

- **Job** \(b\)
  - Release date: \(t = 3\)
  - Deadline: \(t = 6\)
  - Budget: \(c_b\)

The election function is:

- \(t = 2\) \(\Rightarrow\) election function (1) = ∅
- \(t = 3\) \(\Rightarrow\) election function (2) = Job \(a\)
- \(t = 4\) \(\Rightarrow\) election function (3) = Job \(b\)
- \(t = 5\) \(\Rightarrow\) election function (4) = Job \(a\)
Election function of an Earliest Deadline First scheduler

- Job $a$'s execution period: $t = 2$ to $t = 7$
- Budget $c_a$: $t = 3$
- Release date $r_a$: $t = 2$
- Deadline $d_a$: $t = 7$
- $t = 1$ is the election function

Florian Vanhems – florian.vanhems@univ-lille.fr
Election function of an Earliest Deadline First scheduler

- Release date: $r_a = 2$
- Budget: $c_a$
- Deadline: $d_a = 7$
- Execution period: $t = 6$

Election function:
- $t = 2$: $\emptyset$
- $t = 3$: $a$
- $t = 4$: $b$
- $t = 5$: $a$
- $t = 6$: $b$
- $t = 7$: $a$
Election function of an Earliest Deadline First scheduler

- **Job** $a$
  - **Release date** $r_a = 2$ a.u.
  - **Deadline** $d_a = 7$ a.u.
  - **Budget** $c_a = 3$ a.u.

- **Job** $b$
  - **Deadline** $d_b = 6$ a.u.
  - **Release date** $r_b = 3$ a.u.

The election function at different times is as follows:

- $t = 2$: $\text{election function} = \emptyset$
- $t = 3$: $\text{election function} = \text{Job } a$
- $t = 4$: $\text{election function} = \text{Job } b$
- $t = 5$: $\text{election function} = \emptyset$
Election function of an Earliest Deadline First scheduler

Job b

$r_a$ $d_a$

time (a.u.)
Election function of an Earliest Deadline First scheduler

- Job \( \alpha \) released at \( t = 2 \) with budget \( c_\alpha \), deadline \( d_\alpha \), and execution period \( t = 6 \)
- Job \( \beta \) released at \( t = 3 \) with budget \( c_\beta \) and deadline \( d_\beta \) before deadline of \( \alpha \)

Election function:
- At \( t = 2 \): \( \{ \alpha \} \)
- At \( t = 3 \): \( \{ \beta \} \)
- At \( t = 4 \): \( \{ \alpha \} \)
- At \( t = 5 \): \( \{ \alpha \} \)
- At \( t = 6 \): \( \{ \} \)
Election function of an Earliest Deadline First scheduler

- Time (a.u.)
- Job $a$ budget ($c_a$)
  - Release date ($r_a$) $t = 2$
  - Deadline ($d_a$) $t = 6$
- Job's execution period $t = 6$
- Job $b$ budget ($c_b$)
  - Release date ($r_b$) $t = 7$
  - Deadline ($d_b$) $t = 3$
- $d_b < d_a$
- Election function $E(1) = \emptyset$
- Election function $E(2) = Job_a$
- Election function $E(3) = Job_b$
- Election function $E(4) = Job_a$
Election function of an Earliest Deadline First scheduler

\[ \text{Job}_a \text{ budget} (c_a) = 2 \]
\[ \text{release date} (r_a) = 7 \]
\[ \text{deadline} (d_a) = 3 \]
\[ \text{Job's execution period} (t) = 6 \]

\[ r_b < d_a! \]

\[ \text{election function} (1) = \emptyset \]
\[ \text{election function} (2) = \text{Job}_b \]
\[ \text{election function} (3) = \text{Job}_a \]
\[ \text{election function} (4) = \text{Job}_b \]
Election function of an Earliest Deadline First scheduler

\[
\text{election function}(1) = \emptyset \\
t = 1
\]
Election function of an Earliest Deadline First scheduler

election function(2) = Job a

t = 2
Election function of an Earliest Deadline First scheduler

$\text{election function}(3) = \text{Job } b$

$t = 3$
Election function of an Earliest Deadline First scheduler

election function(4) = Job a

\[ t = 4 \]
Proofs & Hypotheses
So what did we prove?

Scheduled job sets are scheduled such that no job misses its deadline
Schedulability property

Given any two moments $t, t'$, let $\Gamma_{t,t'}$ be the set of jobs $j$ to schedule in the interval $[t, t']$. If the sum of the budget $c_j$ of the jobs in that set is less than $t' - t$, then the job set is schedulable.

Definition (Schedulability property)

$$\forall t, t'. t < t' \implies \sum_{j \in \Gamma_{t,t'}} c_j \leq t' - t$$
Well-formedness assumptions

For each job:

- \( r_i + c_i \leq d_i \): the deadline comes late enough for the job to complete its execution if executed alone on the processor.
- \( 0 < \delta_j \leq c_j \): the actual duration \( \delta \) of a job is strictly positive and less than its budget \( c \).
- unique identifiers
- released exactly once
Three steps to reach correctness
Earliest Deadline First policy

EDF scheduling policy

For any job $j$ and any time instant $t$, if the job $j$ is running at instant $t$, then for any other job $j'$ that is ready to run at the same instant, it holds that $d_j \leq d_{j'}$.

Applying the policy on a job set (up to a certain time instant $t$) is defined as:

$$\text{EdfPolicyUpTo } t$$

EDF policy correctness property

$$\text{schedulable } \implies \forall j. \forall t. \text{EdfPolicyUpTo } t \implies \neg \text{overdue } j t$$
Correctness of an intermediate election function

Implement an idealised election function that acts like the one that will be executed.

The next step is to prove it implements the EDF policy defined previously.

**Functional election function implements EDF policy**

\[
\forall t, \forall o, \forall s. \text{idealised}\_scheduler(t) = (o, s) \implies \text{EdfPolicyUpTo } t.
\]

From this property follows the correctness of this idealised election function.
Correctness of the election function

Implement the final, translatable to C, election function *that relies on a chosen set of primitives*.

The next step is to prove that it acts the same way as the functional one.

**Actual election function has same effects as functional**

\[
\forall t. \quad \{ \text{env} = E \land s = \text{init} \} \\
(o, s') := \text{scheduler}(t) \quad \{ \text{idealised_scheduler}(t) = (o, s') \}
\]

From this property follows the correctness of the scheduler.
Models & Implementation
Overview of the scheduler

- Back-end
- Election Function
- Interface
- State
The scheduler from the proof’s point of view
The scheduler from the proof’s point of view

State monad

*idealised* mathematical version of the components

Election Function

Interface

State
The scheduler from the proof’s point of view

- Oracles
- Interface
- State
- Election Function

State monad

Idealised mathematical version of the components

Environment monad

Constraints on the behaviour, no direct description

Florian Vanhems – florian.vanhems@univ-lille.fr
The scheduler from the proof’s point of view

Environment monad

*constraints* on the behaviour, no direct description

Oracles

Election Function

Interface

State

Florian Vanhems – florian.vanhems@univ-lille.fr
The actual scheduler

Election Function

Back-end

Interface

State
The actual scheduler

- Back-end
- Election Function
- Interface
- State

**Actual implementation**
Different from the model, but arguably close enough
Most common yet forgotten assumption

The models properly describe the behaviour of components we rely on.
Conclusion
Conclusion

We have shown an EDF scheduler with a proved election function, describing:

○ The role of the election function, the interface and state, and the back-end
○ The correctness of the election function
○ The assumptions
○ The monadic approach
Thank you for your attention!

Sources & directions to run the scheduler can be found on our repository:

https://github.com/2xs/pip_edf_scheduler

and it passed the artifact validation process