Verifying min-plus Computations with Coq (extended version with appendix)

Lucien Rakotomalala, Pierre Roux, Marc Boyer

Verifying min-plus Computations with Coq (extended version with appendix)*

Lucien Rakotomalala[1], Pierre Roux[1], and Marc Boyer[1]

ONERA / DTIS, Université de Toulouse, FRANCE

Abstract. *Network-calculus* is a theory that bounds delays in embedded networks such as AFDX networks used in modern airplanes. Effective computations rely on operators from the *min-plus* algebra on real functions. Algorithms on specific subsets can be found in the literature. Such algorithms and related implementations are however complicated. Instead of redeveloping a provably correct implementation, we take an existing implementation as an oracle and propose a *Coq* based verifier.

Keywords: network-calculus · min-plus computations · Coq · functions on real numbers.

1 Problem Statement

Network Calculus is a static analysis method used to bound worst case traversal times of networks. It has noticeably been used since a few decades to certify embedded networks, called Avionics Full DupleX (AFDX), on modern civil aircrafts [9]. Basically, given bounds on emission rates of each end node of the network and hypotheses on the scheduling policy implemented in each switch, *Network Calculus* computes sound bounds on the time taken by any packet to travel between any two nodes.

Network Calculus is based on tropical algebra, more precisely the min-plus dioid of functions on real numbers (used to represent both time and amounts of data). Thus, as an intermediate step in the analysis, the method produces algebraic formulas in this dioid, whose computation eventually gives actual numerical bounds. Soundness of the bounds then crucially relies on both the soundness of the *Network Calculus* theory and of those computations. The soundness of *Network Calculus* theory is outside the scope of this paper [17], we will focus here on verification of computations of algebraic operators in the min-plus dioid of functions.

Efficient algorithms are known for these computations and a few effective implementations do exist [6,7,8]. However, these algorithms are rather tricky, hence the interest in formal proofs to greatly increase the level of confidence in their results. We use the proof assistant Coq [13] to provide formal proofs of correctness of such results. To avoid a costly entire reimplementation of the algorithms, we adopt a skeptical approach, using existing implementations as

* Supported by the ANR/DFG Project RT-proofs (ANR-17-CE25-0016).

untrusted oracles and only providing verified implementations of verifiers for each algebraic operation.

Sections 2 and 3 introduce a few notations and give an overview of the objects and operations manipulated throughout the paper. Then, Section 4 recalls the state of the art. Sections 5 and 6 detail the formalization of these objects, while Sections 7 and 8 prove some of their fundamental properties. Finally Section 9 prove the core soundness arguments of the expected verifiers, Section 10 discuss the implementation and Section 11 concludes.

2 Notations

Let $\mathbb{R}$ denote real numbers, $\mathbb{R}_+ \triangleq \mathbb{R} \cap [0; +\infty[$ and $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{-\infty, +\infty\}$. Let $\mathbb{Q}$ denote rational numbers, $\mathbb{Q}_+ \triangleq \mathbb{Q} \cap [0; +\infty[$ and $\mathbb{Q}_+^\star \triangleq \mathbb{Q}_+ \setminus \{0\}$. Let $\mathbb{N}$ denote rational numbers, $\mathbb{N}^\star \triangleq \mathbb{N} \setminus \{0\}$ and $\mathcal{F}$ denote functions from $\mathbb{R}_+$ to $\overline{\mathbb{R}}$. Let $\vee$ denote the logical *or* and $\wedge$ the logical *and*. For any finite set S, let $\#S \in \mathbb{N}$ denote its cardinal and for any sequence s, $last(s)$ denote its last element.

In Coq code appearing in this paper, `nat` will stand for $\mathbb{N}$, `R` for $\mathbb{R}$, `Rbar` for $\overline{\mathbb{R}}$, `R+` for $\mathbb{R}_+$, `Q` for $\mathbb{Q}$, `Q+` for $\mathbb{Q}_+$, `Q+*` for $\mathbb{Q}_+^\star$ and && for logical conjunction $\wedge$.

We also use some list manipulating functions of Coq: `nth`, `head` and `last`. `nth x0 l i` returns the element of index `i` (starting at 0) of the list `l` or `x0` if `l` contains less than `i` elements. `n.+1` and `n.-1` are the successor and the predecessor of any natural number `n` (the predecessor of 0 is 0). The notation %/ is used for euclidean division. To ease readability of the Coq code, we omit scope annotations in the paper. For each result, we give the name of its Coq implementation: for instance `F_UPP` for Definition 1 below. The code is available at `https://www.onera.fr/sites/default/files/447/NCCoq.tar`.

3 *(min, plus)* Operators on Functions

Network Calculus handles functions in $\mathcal{F}$ and uses *(min, plus)* operations over this set: addition, minimum, convolution and deconvolution. We assume that $+\infty + -\infty = +\infty$. We first present these operators. Then, we introduce subclasses of $\mathcal{F}$ stable for these operators and amenable for effective computations.

3.1 *(min, plus)* Operators

The addition $f+g$ and the minimum $\min(f,g)$ of two functions f and g of $\mathcal{F}$ are pointwise extensions of the corresponding operators on $\overline{\mathbb{R}}$, that is $f+g \triangleq t \mapsto f(t)+g(t)$ and $\min(f,g) \triangleq t \mapsto \min(f(t), g(t))$. We also use two operators, the convolution $f * g$ and the deconvolution $f \oslash g$ that are not pointwise operators, defined as:

$$f * g \triangleq t \mapsto \inf_{\substack{u,v \geqslant 0 \\ u+v=t}} (f(u) + g(v)), \qquad f \oslash g \triangleq \inf\{h | f \leqslant h * g\}. \tag{1}$$

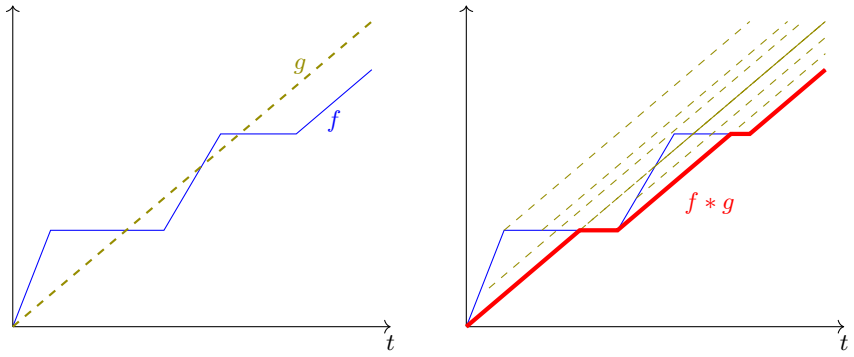


Fig. 1: Two functions f, g (on the left) and their convolution $f * g$ (on the right). Intuitively, the convolution of two functions can be obtained by sliding one function along the other and taking the minimum hull.

where inf on a set $S \subseteq \mathcal{F}$ is $\inf\{S\} \triangleq t \mapsto \inf\{f(t) | f \in S\}$. On Figure 1, we plot an example of convolution. Details can be found in chapter 2 of [5], dedicated to *(min, plus)* theory.

3.2 Sub-classes of Functions for Effective Computation

Network Calculus tools do not manipulates the complete F class but only sub-classes with good stability properties and effective computations [7].

In *Network Calculus*, it is quite common to have periodic behaviors. To describe them, we use functions that are ultimately pseudo-periodic (UPP), denoted $\mathcal{F}_{\text{UPP}}$. A function f belongs to the set $\mathcal{F}_{\text{UPP}}$ if, given a point T (an initial segment), a period d and an increasing element c, it holds, for all t greater than T that $f(t+d) = f(t)+c$. To have a description of these functions, it is sufficient to have the values of T, d and c and the description of the function on the initial segment plus one period.

We consider the sub-class of $\mathcal{F}$ made of the Piecewise Affine (PA) functions, denoted $\mathcal{F}_{\text{PA}}$. For these functions, it is sufficient to give, for each piece, the point of discontinuity, the slope and the offset. These parameters can be recorded in a list although this list can be infinite.

We define $\mathcal{F}_{\text{UPP-PA}} \triangleq \mathcal{F}_{\text{UPP}} \cap \mathcal{F}_{\text{PA}}$. Its elements can be finitely represented by giving T, d and c from $\mathcal{F}_{\text{UPP}}$ and the initial segment of the list from $\mathcal{F}_{\text{PA}}$ representing the function on $[0; T + d)$.

The contributions of this paper are:

- a formalization in Coq of $\mathcal{F}_{\text{UPP-PA}}$ in Sections 5 and 6 and stability properties under *(min, plus)* operations in Sections 7 and 8.
- a check for correctness of addition, minimum and convolution in Section 9.

Intuitively, with the addition, if some tool provides three functions f, g and h and claims that $f + g = h$, we want to check this relation with a finite number of

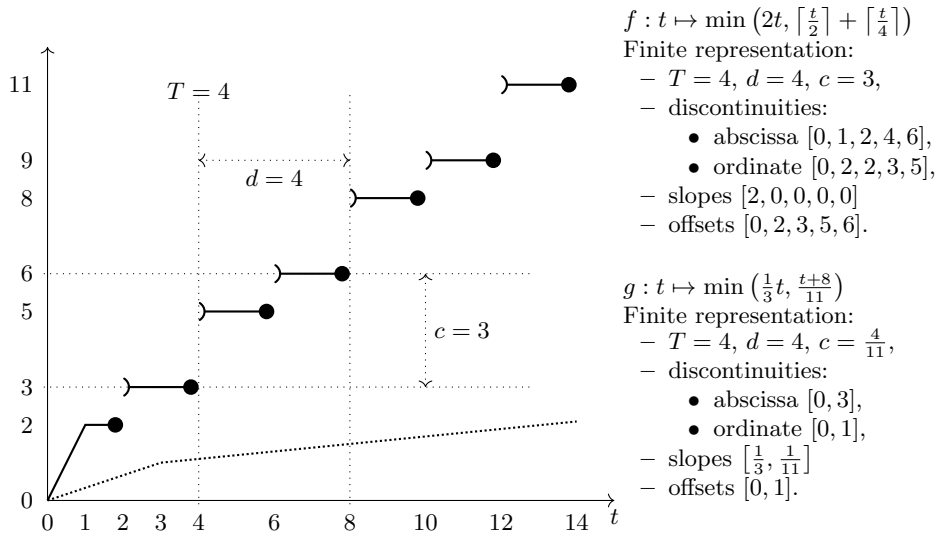


Fig. 2: f (solid) and g (dotted) are UPP-PA functions. Given a UPP-PA function h with compatible parameters, to prove the equality $f + g = h$, it is enough to check $f(t_i) + g(t_i) = h(t_i)$ for a list of t_i: [0; 0.1; 0.9; 1; 1.1; 1.9; 2; 2.1; 2.9; 3; 3.1; 3.9; 4; 4.1; 5.9; 6; 6.1; 7.9].

tests. To this end, we will prove that checking the equality $f(t_i) + g(t_i) = h(t_i)$ on a set of points $t_1, ..., t_n$, plus some compatibility tests on initial segments, periods and increments, is enough to ensure the equality on $\mathbb{R}_+$. We illustrate this on Figure 2.

The minimum and convolution can be checked using similar arguments. Regarding the deconvolution, in practice *Network Calculus* only requires, given two functions f and g, a function h such that $h \geqslant f \oslash g$. It is then enough to check that $h * g \geqslant f$, that is $\min(f, h * g) = f$ which only involve checking a minimum and a convolution.

4 State of the Art

There exist two main classes of curves used in network calculus: the set of concave or convex piecewise linear functions, C[x]PL [19], and the, strictly larger, set of ultimately pseudo-periodic piecewise linear functions UPP-PA, commonly known as UPP [7].

The class of the CPL linear functions has nice mathematical properties: it is stable under the addition and the minimum, and moreover, the convolution can be implemented as a minimum plus a constant. The data structure and related algorithms are so simple that they, to our knowledge, have never been

published. The class of convex piecewise linear functions has very similar properties, replacing minimum by maximum, and its (min,plus) convolution can also be implemented very efficiently [5, Sect. 4.2]. Nevertheless, they cannot accurately model packetized traffic, whereas the UPP-PA class gives better results at the expense of higher computation times [8].

An open implementation of the operators on the C[x]PL class can be found in the DISCO network calculus tool [4].

The algorithms of the operators on the UPP-PA class are given in [7]. An open implementation has been developed but is no longer maintained [6] to our knowledge. An industrial implementation exists, which is the core of the network calculus tool PEGASE [10]. The UPP-PA implementation can be accessed through an on-line console [1].

The Real-Time Calculus toolbox (RTC) does performance analysis of distributed real-time systems [22,23]. Its kernel implements minimum, sum, and convolution on Variability Characterization Curves (VCC's), a class very close to UPP-PA, but no explicit comparison of those two classes has been done up to now.

None of these implementations were formally proved correct.

The first works on the formal verification of network calculus computation were presented in [15]. The aim was to verify that a tool was correctly using the network calculus theory. An Isabelle/HOL library was developed, providing the main objects of network calculus (flows and servers, arrival and service curves) and the statement of the main theorems, but not their proofs. They were assumed to be correct, since they have been established in the literature for long. Then, the tool was extended to provide not only a result, but also a proof on how that network calculus has been used to produce this result. Then, Isabelle/HOL was in charge of checking the correctness of this proof.

Another piece of work, presented in [17], consists in proving, in Coq, the network calculus results themselves: building the min-plus dioid of functions, the main objects of network calculus and the main theorems (statements and proofs).

The PROSA library also provides proofs of correctness for the response time of real-time systems, but focuses on scheduling tasks for processors [11].

5 Ultimately Pseudo Periodic Functions

We now present the formal definition of the set of UPP functions.

Definition 1 (Ultimately Pseudo Periodic Functions, `F_UPP`). *$\mathcal{F}_{UPP}$ is the set of functions $f \in \mathcal{F}$ such that there exists $T \in \mathbb{Q}_+$, $d \in \mathbb{Q}_+^\star$ and $c \in \mathbb{Q}$ for which*

$$\forall t \in \mathbb{R}_+, t \geqslant T \implies f(t+d) = f(t) + c. \tag{2}$$

Remark 1. The values of T, d and c could have been in $\mathbb{R}$. However, we know from [7] that $\mathcal{F}_{\text{UPP}}$ is stable over more operators if T, d and c are rationals. It is not a practical restriction since $\mathbb{Q}$ is the set used in computation.

We represent $\mathcal{F}_{\text{UPP}}$ in Coq as follows.

```
1 Record F_UPP := {
2   F_UPP_val :> R+ → Rbar;
3   F_UPP_T : Q+;  F_UPP_d : Q+*;  F_UPP_c : Q;
4   _ : ∀ t : R+, F_UPP_T ⩽ t →
5       F_UPP_val (t + F_UPP_d) = F_UPP_val t + toR F_UPP_c }.
```

This code means that a value of type `F_UPP` is:

line 2 a function `F_UPP_val` from `nnR` to `Rbar`. The notation :> is a Coq notation for coercion: Coq introduces automatically `F_UPP_val` whenever we give a value of type `F_UPP` when a function from `R+` to `Rbar` is expected.
line 3 `F_UPP_T`, `F_UPP_d` and `F_UPP_c`, the three parameters T, d and c of (2).
lines 4 and 5 the property (2). We use `toR` to cast a rational as a real.

The command `Record` creates a constructor of `F_UPP` named `Build_F_UPP`. To declare a value in `F_UPP`, Coq will require a function, three parameters and a proof of (2).

6 UPP and Piecewise Affine Functions

We briefly presented in section 3 the set $\mathcal{F}_{\text{UPP-PA}}$ of functions that are both UPP and PA. We give in this section a formal definition.

In [7], this set was introduced as the intersection of two sets of functions: $\mathcal{F}_{\text{UPP}}$ and $\mathcal{F}_{\text{PA}}$, the set of PA functions. In this paper, we rather choose to formalize the subset of functions in $\mathcal{F}_{\text{UPP}}$ that are PA, as this greatly simplifies the formalization.

To define PA functions, we need to record points of discontinuities and change of slopes: *jump sequences*.

Definition 2 (Jump Sequence, `JS`). *For any $n \in \mathbb{N}^*$, we call* Jump Sequence *(JS) a tuple $a \in \mathbb{Q}_+^n$ such that $a_0 = 0$ and: $\forall i \in \{0, \dots, n-2\}, a_i < a_{i+1}$. We call n the* size *of the JS and the set of JS of size n is denoted JS_n.*

We represent jump sequences in Coq as follows.

```
Record JS := {
  JS_list :> seq Q+;
  _ : (JS_list != [::])  && (head 0 JS_list == 0) && sorted < JS_list }.
```

A `JS` is a list: `JS_list` of $\mathbb{Q}_+$ that is not an empty list (denoted by [::]), whose initial element is 0 and which is sorted by the usual strict order $<$. The function `head` is a total function: it returns the first element of a list or a default value when empty, here 0.

Each piece is linear on an interval with a slope and an offset.

Definition 3 ((ρ, σ)-*affine* on, `r_s_affine_on`). *Given $\rho, \sigma \in \mathbb{Q}$ and $x, y \in \mathbb{Q}_+$, a function $f \in \mathcal{F}$ is called (ρ, σ)-affine on $]x; y[$ when, for all $t \in]x; y[$:*

$$f(t) = \rho(t - x) + \sigma. \tag{3}$$

We state this definition in Coq as follows.

```
Definition r_s_affine_on (f : F) (rho sigma : Q) (x y : Q+) :=
  ∀ t : R+, x < t < y → f t = toR rho * (t − x) + toR sigma.
```

We want to define a subset of $\mathcal{F} = \mathbb{R}_+ \to \overline{\mathbb{R}}$. So, our functions can return infinite values. The next definition formalizes this point.

Definition 4 (Affine on, `affine_on`). *A function $f \in \mathcal{F}$ is* affine on $]x; y[$ *if*

$$(\forall t \in]x; y[\,, f(t) = +\infty) \tag{4}$$
$$\vee\, (\forall t \in]x; y[\,, f(t) = -\infty) \tag{5}$$
$$\vee\, (\exists \rho, \sigma \in \mathbb{Q}, f \textit{ is } (\rho, \sigma)\textit{-affine on }]x; y[). \tag{6}$$

We state this definition in Coq as follows.

```
Variant affine_on (f : F) (x y : Q+) :=
  | affine_on_p_infty of ∀ t : R+, x < t < y → f t = +∞
  | affine_on_m_infty of ∀ t : R+, x < t < y → f t = −∞
  | affine_on_finite rho sigma of r_s_affine_on f rho sigma x y.
```

We use `Variant` that is a disjunctive version of `Record`.

PA are then functions that are *affine on* all intervals of a JS.

Definition 5 (JS of a Function, `JS_of`). *Let $n \in \mathbb{N}^\star$, $a \in JS_n$ and $f \in \mathcal{F}$. We say that a is a JS of f, denoted $a \in JS(f)$, when for all $i < n - 1$, f is* affine on $]a_i; a_{i+1}[$.

We state this definition in Coq as follows.

```
Definition JS_of a (f : F) :=
  ∀ i, (i.+1 < size a) → r_s_affine_on f (nth 0 a i) (nth 0 a i.+1).
```

So, according to the previous definition, each PA function is associated to a JS but it is not unique. We illustrate this in Figure 3. Also notice that a function $f \in \mathcal{F}$ with $a \in JS(f)$ is a PA function at least up to the last point of a.

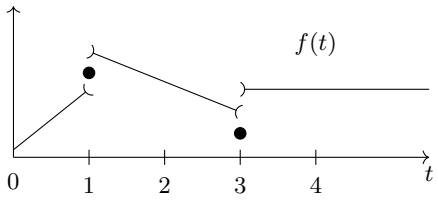


Fig. 3: The function f is piecewise affine. $a \triangleq \{0, 1, 3\}$ and $b \triangleq \{0, 1, 2, 3\}$ are JS of this function: $a \in JS(f)$ and $b \in JS(f)$. We notice that $c \triangleq \{0, 2, 4\} \in JS$ but $c \notin JS(f)$.

Definition 6 (UPP-PA Functions, `F_UPP_PA`). *The set $\mathcal{F}_{UPP\text{-}PA}$ of UPP-PA functions is the set of functions $f \in \mathcal{F}_{UPP}$ with T for initial segment and d for period, such that there exists $a \in JS(f)$ and $last(a) = T + d$.*

We represent $\mathcal{F}_{\text{UPP-PA}}$ in Coq as follows.

```
Record F_UPP_PA := {
  F_UPP_PA_UPP :> F_UPP;
  F_UPP_PA_JS : JS;
  _ : JS_of F_UPP_PA_JS F_UPP_PA_UPP;
  _ : last 0 F_UPP_PA_JS = F_UPP_T F_UPP_PA_UPP + F_UPP_d F_UPP_PA_UPP }.
```

The functions presented in Figure 2 belong to $\mathcal{F}_{\text{UPP-PA}}$. The list of abscissas of discontinuities given in the caption are jump sequences of the functions.

A UPP-PA function with initial segment T and period d is PA in $[0; T + d[$ by construction, and also PA after $T + d$ by periodicity. This point is developed in the following property.

Lemma 1 (`F_UPP_PA_JS_upto_spec` in Coq). *Let $f \in \mathcal{F}_{UPP\text{-}PA}$ with $a \in JS(f)$. For any $l \in \mathbb{Q}_+$ such that $last(a) \leqslant l$, there exists $a' \in JS$ such that $a' \in JS(f)$ and $last(a') = l$.*

7 Stability of UPP Functions by *(min, plus)* Operators

We now want to prove stability of $\mathcal{F}_{\text{UPP}}$ over *(min, plus)* operators: addition, minimum and convolution. These operators have been presented in Section 3. We need another operator on rational numbers: a notion of least common integer multiple such that, for any $d, d' \in \mathbb{Q}$, there exists $k, k' \in \mathbb{N}$ satisfying $kd = k'd' = lcm_Q(d, d')$.

Definition 7 ($lcm_{\mathbb{Q}_+^\star}$). *For all $d, d' \in \mathbb{Q}_+^\star$, for all $a, a' \in \mathbb{Z}$ and $b, b' \in \mathbb{N}^\star$ such that $d = \frac{a}{b}$ and $d' = \frac{a'}{b'}$, we define*

$$lcm_{\mathbb{Q}_+^\star}(d, d') \triangleq \frac{\text{lcm}\left(a\frac{\text{lcm}(b,b')}{b}, a'\frac{\text{lcm}(b,b')}{b'}\right)}{\text{lcm}(b, b')} \tag{7}$$

where lcm *is the least common multiple on $\mathbb{Z}$.*

Lemma 2 (`dvdq_lcml` in Coq). *For $d, d' \in \mathbb{Q}_+$, there is $k \in \mathbb{N}$ s.t. $lcm_{\mathbb{Q}_+^\star}(d, d') = k\,d$.*

We state this lemma in Coq as follows.

```
Definition lcm_Q (d d' : Q) : Q :=
  fracq (lcmz (numq d * (lcmz (denq d) (denq d') %/ denq d))
              (numq d' * (lcmz (denq d) (denq d') %/ denq d')),
         lcmz (denq d) (denq d')).
Program Definition lcm_posQ (d d' : Q+*) : Q+* := mk_posQ (lcm_Q d d') _.
Lemma dvdq_lcml d d' : ∃ k : nat, lcm_posQ d d' = k * d.
```

We first define `lcm_Q`: the definition of $lcm_{\mathbb{Q}_+^\star}$ on $\mathbb{Q}$. The functions `fracq`, `numq` and `denq` are respectively the constructor and destructors of `Q`. The command `Program Definition` is similar to `Definition` except that it accepts holes `_` and automatically generates the corresponding proof obligations.

To ease notations, we want to transform this binary operator, into a set operator such as $\sum_{i=1}^{3} i = (1+2)+3$. There exists a library in Coq designed with this objective: the `bigop` theory of *Mathcomp* [2]. To fully use this library, we need to prove that $lcm_{\mathbb{Q}_+^\star}$ satisfies the monoid laws. In other words, we need to prove that $lcm_{\mathbb{Q}_+^\star}$ is associative and has a neutral element. However, $lcm_{\mathbb{Q}_+^\star}$ does not have a neutral element. The lcm on $\mathbb{N}$ has a neutral element 1. It is not the case for $lcm_{\mathbb{Q}_+^\star}$: for instance $lcm_{\mathbb{Q}_+^\star}\left(1, \frac{2}{3}\right) = 2$. To get out of it, we need to extend the definition of $lcm_{\mathbb{Q}_+^\star}$:

```
Definition olcm_posQ (x y : option Q+*) : option Q+* := match x, y with
 | None, _ ⇒ y | _, None ⇒ x | Some x, Some y ⇒ Some (lcm_posQ x y)
 end.
```

The `option` type is used to extend the type of `Q+*` with a `None` element. Then, this element is the neutral element for this optional definition of $lcm_{\mathbb{Q}_+^\star}$. We add then a `Notation` for the big operator.

```
Notation "\biglcm_posQ_ ( i < n ) F" :=
  (odflt one_posQ (\big[olcm_posQ/None]_(i < n) some F)) : ring_scope.
```

`\big[oclm_posQ\None]_(i < n) some F` is the iterated application of `oclm_posQ` for all `i` such that `i < n` on `some F`. The function `odflt` removes the `option` when it is `Some` and returns a default value otherwise.

The following lemmas prove stability of $\mathcal{F}_{\text{UPP}}$ by addition, minimum and convolution.

Lemma 3 (`F_UPP_n_add` in Coq). *Let $n \in \mathbb{N}^\star$, $f \in \mathcal{F}^n_{UPP}$ with initial segments $T \in \mathbb{Q}_+^n$, periods $d \in (\mathbb{Q}_+^\star)^n$ and increments $c \in \mathbb{Q}^n$ respectively. The sum $\sum_i f_i$ is a UPP function with an initial segment $\max_i\{T_i\}$, a period $\underset{i}{lcm_{\mathbb{Q}_+^\star}}(d_i)$ and an increment $\underset{i}{lcm_{\mathbb{Q}_+^\star}}(d_i)\left(\sum_i \frac{c_i}{d_i}\right)$.*

Lemma 4 (`F_UPP_n_min` in Coq). *Let $n \in \mathbb{N}^\star$ and $f \in \mathcal{F}^n_{UPP}$ with initial segments $T \in \mathbb{Q}_+^n$, periods $d \in (\mathbb{Q}_+^\star)^n$ and increments $c \in \mathbb{Q}^n$ respectively. Defining:*

$$s \triangleq \min_{i\in[0;n-1]}\left(\frac{c_i}{d_i}\right) \qquad I \triangleq \left\{ i \in [0;n-1] \middle| \frac{c_i}{d_i} = s \right\} \tag{8}$$

and assuming there exists $M \in \mathbb{Q}$ and $m \in \mathbb{Q}^n$ such that:

$$\exists i \in I, \forall t \in [T_i; T_i + d_i[\,, f_i(t) \leqslant M + s\,t \tag{9}$$

$$\forall i \notin I, \forall t \in [T_i; T_i + d_i[\,, m_i + \frac{c_i}{d_i} t \leqslant f_i(t) \tag{10}$$

the function $\min_{i=1}^{n}\{f_i\}$ *is UPP with an initial segment* $\tilde{T}$*, a period* $\tilde{d}$ *and an increment* $\tilde{c}$ *with* $\tilde{d} \triangleq \underset{i\in I}{lcm_{\mathbb{Q}_+^\star}}(d_i)$*,* $\tilde{c} \triangleq \tilde{d}\,s$ *and*

$$\tilde{T} = \max\left(\max_{i\notin I}\left(\frac{M-m_i}{\frac{c_i}{d_i}-s}\right), \max_{j\in[0;n-1]}\{T_j\}\right).$$

These lemmas are a straightforward generalization of Proposition 6 in [7] where it is proved for binary addition and minimum. This generalization is useful for the next lemma on convolution of two UPP functions.

Remark 2. In the case of PA functions, it is easy to find values for M and m_i satisfying (9) and (10) by computing the bounds $\sup_{t\in[T_i;T_i+d_i[}\{f_i(t)-s\,t\}$ and $\inf_{t\in[T_i;T_i+d_i[}\left\{f_i(t)-\frac{c_i}{d_i}t\right\}$.

Lemma 5 (`F_UPP_conv` in Coq). *Let* $f, f' \in \mathcal{F}_{UPP}$ *with initial segments* $T, T' \in \mathbb{Q}_+$*, periods* $d, d' \in \mathbb{Q}_+^\star$ *and increments* $c, c' \in \mathbb{Q}$ *respectively. For all* $M, M', m, m' \in \mathbb{Q}$ *such that*

$$M \geqslant \sup_{t\in[T,T+d[}\left\{f(t)-\frac{c}{d}(t+T')\right\}+f'(T') \tag{11}$$

$$m' \leqslant \inf_{t\in[0,T[}\left\{f(t)-\frac{c'}{d'}t\right\}+\inf_{t\in[T',T'+d'[}\left\{f'(t)-\frac{c'}{d'}t\right\} \tag{12}$$

and similarly for M' *and* m*, by permuting the primed and non-primed variables, the convolution* $f * f'$ *is a UPP function with a period* $\tilde{d} \triangleq lcm_{\mathbb{Q}_+^\star}(d,d')$*, an increment* $\tilde{c} \triangleq \tilde{d}\min\left(\frac{c}{d},\frac{c'}{d'}\right)$ *and an initial segment:*

$$\tilde{T} = \begin{cases} T+T'+lcm_{\mathbb{Q}_+^\star}(d,d') & \text{if } \frac{c}{d}=\frac{c'}{d'} \\ \max\left(\frac{M-m'}{\frac{c'}{d'}-\frac{c}{d}}, T+T'+lcm_{\mathbb{Q}_+^\star}(d,d')\right) & \text{if } \frac{c}{d}<\frac{c'}{d'} \\ \max\left(\frac{M'-m}{\frac{c}{d}-\frac{c'}{d'}}, T+T'+lcm_{\mathbb{Q}_+^\star}(d,d')\right) & \text{if } \frac{c'}{d'}<\frac{c}{d} \end{cases} \tag{13}$$

This lemma is proved into Coq as `F_UPP_conv`. It generalizes Proposition 6 of [7] by expliciting the initial value giving a value for $\tilde{T}$. Remark 2 also applies here.

8 Stability of UPP-PA Functions by *(min, plus)* Operators

We are now focusing on stability of $\mathcal{F}_{\text{UPP-PA}}$ by *(min, plus)* operators. Let us first define the union of two jump sequences.

Definition 8 (Union of two JS, `union`). *For any* $n, m \in \mathbb{N}^*, a \in JS_n, b \in JS_m$*, the tuple of size* $\#\left(\{a_i | 0 \leqslant i < n\} \cup \{b_j | 0 \leqslant j < m\}\right)$ *containing the elements of* $\{a_i | 0 \leqslant i < n\} \cup \{b_j | 0 \leqslant j < m\}$ *sorted by increasing order, is called* union *of the jump sequences* a *and* b*. This union is denoted* $a \cup b$*.*

If jump sequences are implemented by lists, the union can be implemented by the merge part of a merge sort, followed by a removal of duplicates. We state this definition in Coq as follows.

```
Program Definition union (a b : JS) := @Build_JS (undup (merge ⩽ a b)) _.
```

The following Lemma gives a jump sequence for the sum of PA functions.

Lemma 6 (JS of n-ary Addition, `JS_of_n_add`). *For $n \in \mathbb{N}^\star$, for $f \in \mathcal{F}^n$ and for $a \in JS^n$, if for all i, $a_i \in JS(f_i)$ and all the last points of a are equal ($\forall i, j, last(a_i) = last(a_j)$), then $\bigcup_i a_i \in JS\left(\sum_i f_i\right)$.*

We state this lemma in Coq as follows.

```
Lemma JS_of_n_add n (f : 'I_n.+1 → F) (a : 'I_n.+1 → JS) :
  (∀ i, JS_of (a i) (f i)) → (∀ i j, last 0 (a i) = last 0 (a j)) →
  JS_of (\bigcup_i a i) (\sum_i f i).
```

The term `bigcup_i` is the notation for $\bigcup_i$. Thanks to Lemma 1, the equality of last points can always be satisfied. Stability of $\mathcal{F}_{\text{UPP-PA}}$ by n-ary addition can then be derived from this Lemma and Lemmas 1 and 3.

Whereas the jump sequence of a sum is the union of the jump sequences, the minimum can introduce new points as shown in Figure 4. The following definition gives such a jump sequence.

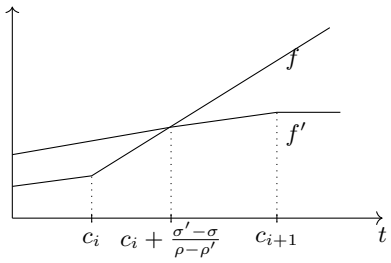


Fig. 4: Example of point added by the min operator in a JS. f and f' are respectively (ρ, σ)-*affine* and (ρ', σ')-*affine* on $]c_i; c_{i+1}[$ with different slopes ρ and ρ'. Since we have $c_i + \frac{\sigma'-\sigma}{\rho-\rho'} \in]c_i; c_{i+1}[$, this point must be added to the jump sequence.

Definition 9 (Union min, `union_min`). *Let f and $f' \in \mathcal{F}$ with $a \in JS(f)$ and $a' \in JS(f')$ such that $last(a) = last(a')$. Set $c \triangleq a \cup a'$. We define the $\cup_{\min}$ operator as*

$$\cup_{\min}(f, f', a, a') \triangleq c \cup \left\{ c_i + \frac{\sigma' - \sigma}{\rho - \rho'} \middle| \begin{array}{l} \exists i, i < \#c - 1 \\ \wedge\ f \text{ is } (\rho, \sigma) - \text{affine on }]c_i, c_{i+1}[\\ \wedge\ f' \text{ is } (\rho', \sigma') - \text{affine on }]c_i, c_{i+1}[\\ \wedge\ \rho \neq \rho' \wedge\ c_i < c_i + \frac{\sigma'-\sigma}{\rho-\rho'} < c_{i+1}. \end{array} \right\} \tag{14}$$

Using this $\cup_{\min}$ operator, we can establish a JS for n-ary minimum.

Lemma 7 (`JS_of_n_min` in Coq). *For all $n \in \mathbb{N}^\star$ and $f \in \mathcal{F}^n$, if for all i, $a_i \in JS(f_i)$ and all the last points of a are equal then*

$$\left(\bigcup_{i,j \in [0,n-1]} \cup_{\min}(f_i, f_j, a_i, a_j) \right) \in JS\left(\min_i\{f_i\}\right). \tag{15}$$

Just as we mentioned for the addition, this Lemma and Lemmas 1 and 4 are sufficient to prove stability of $\mathcal{F}_{\text{UPP-PA}}$ by n-ary minimum under mild conditions [1].

We are now interested in the convolution of two UPP-PA functions. Like in [7], we rely on the property that: $\forall f, g, h \in \mathcal{F}, \min(f, g) * h = \min(f * h, g * h)$. Then, any UPP-PA function can be decomposed as the minimum of elementary functions whose convolution is easy to compute.

In the following, we give such a decomposition.

Definition 10 (Cutting Operator, `cutting_operator`). *Given $f \in \mathcal{F}$, $a \in JS(f)$ and $i \in \mathbb{N}$ such that $i < \#a - 1$, we define the cutting operator:*

$$(f \downarrow a)_i \triangleq t \mapsto \begin{cases} f(t) & \text{if } t \in [a_i; a_{i+1}[\\ +\infty & \text{otherwise .} \end{cases} \tag{16}$$

We state this definition in Coq as follows.

```
Definition cutting_operator (f : F) a i : F := fun t ⇒
  if i.+1 < size a && (nth 0 a i ⩽ t < nth 0 a i.+1) then f t else +∞ .
```

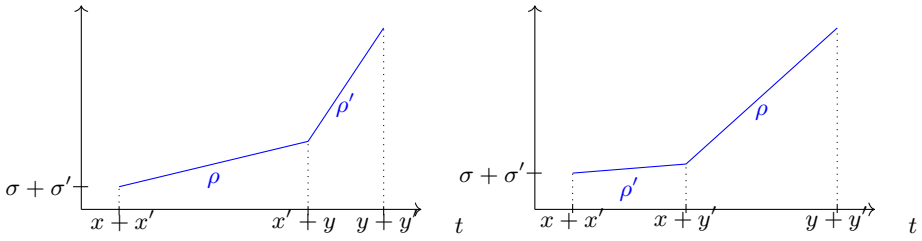


Fig. 5: Convolution of two segments. Let f and f' be two functions that are respectively (ρ, σ)-*affine* on $[x; y[$ and (ρ', σ')-*affine* on $[x'; y'[$ and $+\infty$ elsewhere. We plot the two cases of $f * f'$ on $[x + x', y + y'[$: left is for $\rho < \rho'$ and right is $\rho' < \rho$.

The convolution of two functions $(f \downarrow a)_i$ and $(f' \downarrow a')_j$ can be computed by case disjunction in the same way as in Figure 5 but considering possible discontinuities lead to more than two sub-cases.

[1] existence of m and M for Lemma 4

We need a last definition to specify the previous *cutting operator*.

Definition 11 (Cutting Below, `cutting_below`). *Let $f \in \mathcal{F}$ and $l \in \mathbb{R}_+$, we denote $f_{<l}$ the function that is equal to f up to l and $+\infty$ afterwards.*

Lemma 8 (`cutting_operator_spec` in Coq). *Given $f \in \mathcal{F}$ and $a \in JS(f)$, we have $f_{<last(a)} = \min_{i<\#a-1} (f \downarrow a)_i$.*

We can now give the decomposition of the convolution using these operators.

Lemma 9 (Piecewise Affine Convolution, `PA_conv`). *Let $f, f' \in \mathcal{F}$ with $a \in JS(f)$ and $a' \in JS(f')$ and let l such that $l = last(a) = last(a')$. We have*

$$(f * f')_{<l} = \left(\min_{i,j} \left((f \downarrow a)_i * (f' \downarrow a')_j \right) \right)_{<l}. \tag{17}$$

9 Finite Equality Criteria on UPP-PA

In Sections 5 to 8, we proved in Coq slight variations of results from the literature. Here are the main results: the finite equality tests briefly introduced in Figure 2.

Definition 12 (Equality on a Segment, `eq_segment`). *For all $a \in JS, i \in \mathbb{N}$ and $f, g \in \mathcal{F}$, we define eq_segment(a, i, f, g), the following property:*

$$f(a_i) = g(a_i) \wedge \exists x, y \in]a_i; a_{i+1}[, x \neq y \wedge f(x) = g(x) \wedge f(y) = g(y). \tag{18}$$

We state this definition in Coq as follows.

```
Definition eq_segment (a : JS) i (f g : F) := f (a i) = g (a i)
  ∧ ∃ x y : R+, a i < x < a i.+1 ∧ a i < y < a i.+1 ∧ x ≠ y ∧ f x = g x ∧ f y = g y.
```

This definition is useful to check equality on an interval. Given two functions f, g both *affine* on $]a_i; a_{i+1}[$, *eq_segment*(a, i, f, g) ensures that $f = g$ on $[a_i; a_{i+1}[$.

Combined with previous results, we get an equality criteria for the addition.

Proposition 1 (`UPP_PA_n_add` in Coq). *For all $n \in \mathbb{N}^\star$, $f \in \mathcal{F}^n_{UPP\text{-}PA}$, $f' \in \mathcal{F}_{UPP\text{-}PA}$ with initial segments $T \in \mathbb{Q}^n_+$ and $T' \in \mathbb{Q}_+$, periods $d \in (\mathbb{Q}^\star_+)^n$ and $d' \in \mathbb{Q}^\star_+$, increments $c \in \mathbb{Q}^n$ and $c' \in \mathbb{Q}$ respectively, we define $l \triangleq \max\{\max_i\{T_i\}, T'\} + lcm_{\mathbb{Q}^\star_+}\left(\underset{i}{lcm_{\mathbb{Q}^\star_+}}(d_i), d' \right)$, and $u \triangleq (\bigcup_i a_i) \cup a'$, where for all i, $a_i \in JS(f_i)$ and $last(a_i) = l$, $a' \in JS(f')$ and $last(a') = l$. If $\sum_i \left(\frac{c_i}{d_i}\right) = \frac{c'}{d'}$ then*

$$\forall i < \#u - 1, eq\_segment\left(u, i, \sum_j f_j, f'\right) \tag{19}$$

is a sufficient condition for $\sum_i f_i = f'$.

The condition (19) happens to be also necessary but we do not need to prove it.

Remark 3. This criteria can be computed in finite time. a_i and a' can be obtained using Lemma 1. To check *eq_segment*(a, i, f, f'), one can take $x = \frac{a_i + a_{i+1}}{2}$ and $y = \frac{a_i + x}{2}$.

We get similar criteria for the minimum and the convolution.

Proposition 2 (`UPP_PA_n_min` in Coq). *Let $n \in \mathbb{N}^\star$ and $f \in \mathcal{F}^n_{UPP\text{-}PA}$. For all $f' \in \mathcal{F}_{UPP\text{-}PA}$ with initial segment $T' \in \mathbb{Q}_+$, periods $d' \in \mathbb{Q}_+^\star$ and increment $c' \in \mathbb{Q}$, assume M and m satisfying the hypotheses of Lemma 4 and define $\tilde{T}, \tilde{d}$ and $\tilde{c}$ as in Lemma 4. We define $l \triangleq \max(\tilde{T}, T') + lcm_{\mathbb{Q}_+^\star}(\tilde{d}, d')$ and $u \triangleq \left(\bigcup_{i,j} \cup_{\min}(f_{<l_i}, f_{<l_j}, a_i, a_j)\right) \cup a'$, where for all i, $a_i \in JS(f_i)$ and $last(a_i) = l$, $a' \in JS(f')$ and $last(a') = l$. If $\frac{\tilde{c}}{\tilde{d}} = \frac{c'}{d'}$, then:*

$$\forall i < \#u - 1, \textit{eq\_segment}\left(u, i, \min_j(f_j), f'\right) \tag{20}$$

is a sufficient condition for $\min_i(f_i) = f'$.

Proposition 3 (`F_UPP_conv` in Coq). *Let $f, f' \in \mathcal{F}_{UPP\text{-}PA}$. For all $f'' \in \mathcal{F}_{UPP\text{-}PA}$ with initial segment $T'' \in \mathbb{Q}_+$, period $d'' \in \mathbb{Q}_+$ and increment $c'' \in \mathbb{Q}$, assume M, M', m and $m' \in \mathbb{Q}$ satisfying hypotheses of Lemma 5 and define $\tilde{T}, \tilde{d}, \tilde{c}$ as in Lemma 5. We define $l \triangleq \max(\tilde{T}, T'') + lcm_{\mathbb{Q}_+^\star}(\tilde{d}, d'')$. Assume $a \in JS(f)$ and $last(a) = l$, $a' \in JS(f')$ and $last(a') = l$, $a'' \in JS(f'')$ and $last(a'') = l$ and define $k \triangleq \#a - 1$ and $k' \triangleq \#a' - 1$. Assuming $\tilde{a} \in JS^{\{0,\ldots,k-1\}\times\{0,\ldots,k'-1\}}$ such that for all i, i', $\tilde{a}_{i,i'} \in JS\left((f \downarrow a)_i * (f' \downarrow a')_{i'}\right)$ and $last(\tilde{a}_{i,i'}) = l$, define*

$$u \triangleq \left(\bigcup_{(i,i'),(j,j')} \cup_{\min}\left((f \downarrow a)_i * (f' \downarrow a')_{i'}, (f \downarrow a)_j * (f' \downarrow a')_{j'}, \tilde{a}_{i,i'}, \tilde{a}_{j,j'}\right)\right) \cup a'', \tag{21}$$

if $\frac{\tilde{c}}{\tilde{d}} = \frac{c''}{d''}$ then

$$\forall j < \#u - 1, \textit{eq\_segment}\left(u, i, \min_{i,j}\left((f \downarrow a)_i * (f' \downarrow a')_j\right), f''\right) \tag{22}$$

is a sufficient condition for $f * f' = f''$.

Just as for Proposition 1, these sufficient criteria can be checked in finite time.

10 Implementation

The implementation consists of 6.3k lines of Coq code. It uses the rational numbers defined in the MathComp library [16] and the real numbers from Coq's standard library [21]. These real numbers are linked to the algebraic structures from

MathComp thanks to the `Rstruct.v` file of the MathComp Analysis library [18]. This enables in particular the use of the big operators from MathComp [2]. The extended real numbers $\overline{\mathbb{R}}$ and a few other definitions on real numbers are based on the Coquelicot library [3]. The real numbers from the standard library and Coquelicot could probably now be fully replaced by the MathComp Analysis library, which was in an early development stage when we started this work but now looks much more usable. This would avoid many painful translations back and forth between the two diferent formalizations.

To obtain executable Coq programs, some adjustments were required, such as making the ρ and σ of Definition 5 explicit in the jump sequences. The final executable version consist of 9k lines of Coq (including the previous formalization) and uses the refinement of MathComp's rational numbers by the one in the bignums library [14] provided by the CoqEAL library [12].

Here is an example proof on the sum of the two functions f and g from Figure 2. We first declare f and g:

```
Let f := F_of_sequpp (mk_sequpp 4 (* T *) 4 (* d *) 3 (* c *) [:: (0, (0, (2, 0)));
  (1, (2, (0, 2))); (2, ( 2, (0, 3))); (4, ( 3, (0, 5))); (6, ( 5, (0, 6)))]).
Let g := F_of_sequpp (mk_sequpp 4 4 (4/11) [:: (0, (0, (1/3, 0))); (3, (1, (1/11, 1)))]).
```

Then a function h that we want to prove equal to $f + g$ (this function could be obtained from an external oracle):

```
Let h := F_of_sequpp (mk_sequpp 4 4 (37/11) [:: (0, ( 0, (7/3, 0)));
  (1, (7/3, (1/3, 7/3))); (2, (8/3, (1/3, 11/3))); (3, (4, (1/11, 4)));
  (4, (45/11, (1/11, 67/11))); (6, (69/11, (1/11, 80/11)))]).
```

We can then use our new tactic `nccoq` to automatically prove the equality:

```
Goal f + g = h. Proof. nccoq. Qed.
```

This tactic performs proofs by reflection: it reduces the goal to prove down to a computation which is then performed by Coq and whose success concludes the proof. This reduction is done with the help of the machinery provided by the CoqEAL library [12].

11 Conclusion

Confidence in latency bounds computed by *Network Calculus* tools [8,20] relies, among other parts, on the correctness of the evaluation of algebraic expressions on *(min, plus)* operators [1,6]. Instead of developing another toolbox, we developed, formalized and proved equality criteria that can be checked in finite time for each algebraic operation involved in actual computation of *Network Calculus* bounds.

The expected usage of this library is to delegate the evaluation of arbitrary algebraic expressions to an external tool [1] before checking the final result with our Coq contribution. This external tool would then act as an untrusted oracle.

References

1. RealTime-at-Work online Min-Plus interpreter for Network Calculus. `https://www.realtimeatwork.com/minplus-playground`. Accessed: 2020-11-18.
2. Y. Bertot, G. Gonthier, S. Ould Biha, and I. Pasca. Canonical big operators. In O. A. Mohamed, C. Muñoz, and S. Tahar, editors, *Theorem Proving in Higher Order Logics*, pages 86–101, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
3. S. Boldo, C. Lelay, and G. Melquiond. Coquelicot: A user-friendly library of real analysis for coq. *Math. Comput. Sci.*, 9(1):41–62, 2015.
4. S. Bondorf and J. B. Schmitt. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proc. of the International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '14, pages 44–49, December 2014.
5. A. Bouillard, M. Boyer, and E. Le Corronc. *Deterministic Network Calculus: From Theory to Practical Implementation.* 10 2018.
6. A. Bouillard, B. Cottenceau, B. Gaujal, L. Hardouin, S. Lagrange, M. Lhommeau, and E. Thierry. COINC library: a toolbox for the network calculus. In *Proceedings of the 4th international conference on performance evaluation methodologies and tools, ValueTools*, volume 9, 2009.
7. A. Bouillard and E. Thierry. An Algorithmic Toolbox for Network Calculus. *Discrete Event Dynamic Systems: Theory and Applications*, 18, 03 2008.
8. M. Boyer, J. Migge, and M. Fumey. PEGASE, A Robust and Efficient Tool for Worst Case Network Traversal Time. In *Proc. of the SAE 2011 AeroTech Congress & Exhibition*, Toulouse, France, 2011. SAE International.
9. M. Boyer, N. Navet, and M. Fumey. Experimental assessment of timing verification techniques for AFDX. In *6th European Congress on Embedded Real Time Software and Systems*, Toulouse, France, Feb. 2012.
10. M. Boyer, N. Navet, X. Olive, and E. Thierry. The PEGASE Project: Precise and Scalable Temporal Analysis for Aerospace Communication Systems with Network Calculus. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 122–136, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
11. F. Cerqueira, F. Stutz, and B. B. Brandenburg. PROSA: A case for readable mechanized schedulability analysis. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 273–284. IEEE, 2016.
12. C. Cohen, M. Dénès, and A. Mörtberg. Refinements for free! In G. Gonthier and M. Norrish, editors, *Certified Programs and Proofs*, volume 8307, pages 147–162. Springer, 2013.
13. The Coq development team. *The Coq proof assistant reference manual*, 2020. Version 8.12.
14. B. Grégoire and L. Théry. A Purely Functional Library for Modular Arithmetic and Its Application to Certifying Large Prime Numbers. In *IJCAR*, pages 423–437, 2006.
15. E. Mabille, M. Boyer, L. Fejoz, and S. Merz. Towards certifying network calculus. In *Proc. of the 4th Conference on Interactive Theorem Proving (ITP 2013)*, Rennes, France, July 2013.
16. A. Mahboubi and E. Tassi. *Mathematical Components.* 2018.
17. L. Rakotomalala, M. Boyer, and P. Roux. Formal Verification of Real-time Networks. In *JRWRTC 2019, Junior Workshop RTNS 2019*, Toulouse, France, Nov. 2019.

18. D. Rouhling. *Formalisation Tools for Classical Analysis - A Case Study in Control Theory. (Outils pour la Formalisation en Analyse Classique - Une Étude de Cas en Théorie du Contrôle)*. PhD thesis, University of Côte d'Azur, Nice, France, 2019.
19. H. Sariowan, R. L. Cruz, and G. C. Polyzos. SCED: A generalized scheduling policy for guaranteeing quality-of-service. *IEEE/ACM transactions on networking*, 7(5):669–684, October 1999.
20. J. Schmitt and F. Zdarsky. The DISCO network calculator: a toolbox for worst case analysis. page 8, 01 2006.
21. V. Semeria. Nombres réels dans Coq. In *JFLA*, pages 104–111, 2020.
22. E. Wandeler. Modular performance analysis and interface based design for embedded real time systems. 2006.
23. E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox, 2006.

This appendix contains pen and paper versions of the proofs formalized in Coq in our code development.

A UPP-PA and Piecewise Affine functions

Definition 13 (JS below, `JS_below`). *Let $a \in JS$ and $l \in \mathbb{Q}_+$, the JS below of a at l is the sequence c where, for all i such that $a_i < l$, $c_i = a_i$ and $c_{last(c)} = l$. We denote the JS below of a at l by $a_{\leqslant l}$.*

Definition 14 (UPP-PA JS upto definition, `F_UPP_PA_JS_upto_def`). *Let $f \in \mathcal{F}_{UPP\text{-}PA}$ with T, d as initial segment and period, $a \in JS(f)$ and $l \in \mathbb{Q}_+$. We call the sequence* UPP-PA JS upto *the sequence c where, for $a_d \triangleq \{x \in a | x > T\}$, $n \triangleq \left\lceil \frac{l-(T+d)}{d} \right\rceil$ and for all i:*

$$c \triangleq a \bigcup_{n' \in [1;n]} \{x + n'd | x \in a_d\} \tag{23}$$

We denote it the sequence a upto l.

Proof (of Lemma 1, `F_UPP_PA_JS_upto_spec`). Take $a'_{\leqslant l} \triangleq c_{\leqslant l}$. $a' \in JS(f)$ is proved by induction on n, defined in Definition 14 and $last(a') = l$ comes from Definition of $c_{\leqslant l}$.

B Stability of UPP functions by *(min, plus)* operators

Proof (of Lemma 2). Let $d, d' \in \mathbb{Q}_+^\star$. Let's prove that it exists $k \in \mathbb{N}$ such that $lcm_{\mathbb{Q}_+^\star}(d, d') = k\,d$. By Definition of lcm, it exists $k \in \mathbb{N}$ such that

$$\operatorname{lcm}\left(a\frac{\operatorname{lcm}(b,b')}{b}, a'\frac{\operatorname{lcm}(b,b')}{b'}\right) = k\left(a\frac{\operatorname{lcm}(b,b')}{b}\right)$$

Thus, dividing both sides by $\operatorname{lcm}(b, b')$,

$$lcm_{\mathbb{Q}_+^\star}(d, d') = k\frac{\left(a\frac{\operatorname{lcm}(b,b')}{b}\right)}{\operatorname{lcm}(b,b')} = kd.$$

Lemma 10 (Commutativity of $lcm_{\mathbb{Q}_+^\star}$, `lcm_nnQC`). *$lcm_{\mathbb{Q}_+^\star}$ is commutative.*

Proof. This comes from the commutativity of lcm.

Lemma 11 (`dvdq_lcm`). *For all $d, d' \in \mathbb{Q}_+^\star$, $m \in \mathbb{Q}$, $k, k' \in \mathbb{N}$, if $m = k\,d$ and $m = k'\,d'$, then there exists $k'' \in \mathbb{N}$ such that $m = k''\,lcm_{\mathbb{Q}_+^\star}(d, d')$.*

Proof. Noting $\frac{a}{b} \triangleq d$ and $\frac{a'}{b'} \triangleq d'$ as well as $x \triangleq a\,\frac{\mathrm{lcm}(b,b')}{b}$, and $y \triangleq a'\,\frac{\mathrm{lcm}(b,b')}{b'}$, we have $m\ \mathrm{lcm}(b,b') = k\,x$ and $m\ \mathrm{lcm}(b,b') = k'\,y$. Notice that x and y are integers. Let's prove that $\mathrm{lcm}(x,y)$ divides $k\,x$. By definition of lcm this is true when both x and y divide $k\,x$, that is when y divides $k\,x$ which is true since $k'\,y = k\,x = m\,lcm(b,b')$. Thus, there exists some k'' such that $m\ \mathrm{lcm}(b,b') = k\,x = k''\ \mathrm{lcm}(x,y)$ and dividing by $\mathrm{lcm}(b,b')$, we get $m = k''\,\frac{\mathrm{lcm}(x,y)}{\mathrm{lcm}(b,b')} = k''\ lcm_{\mathbb{Q}_+^\star}(d,d')$.

Lemma 12 (`dvdq_ge_lcm`). *For all $d,d',m \in \mathbb{Q}_+^\star$, $k,k' \in \mathbb{N}$, if $m = k\,d$ and $m = k'\,d'$, then $lcm_{\mathbb{Q}_+^\star}(d,d') \leqslant m$.*

Proof. According to Lemma 11, there exists k'' such that $m = k''\ lcm_{\mathbb{Q}_+^\star}(d,d')$. If $k'' = 0$ then $m = 0$ which is impossible so $k'' \geqslant 1$.

Lemma 13 (Associativity of $lcm_{\mathbb{Q}_+^\star}$, `lcm_nnQA`). *$lcm_{\mathbb{Q}_+^\star}$ is associative.*

Proof. Given $a,b,c \in \mathbb{Q}_+^\star$, we want to prove that $lcm_{\mathbb{Q}_+^\star}\left(a, lcm_{\mathbb{Q}_+^\star}(b,c)\right) = lcm_{\mathbb{Q}_+^\star}\left(lcm_{\mathbb{Q}_+^\star}(a,b), c\right)$. By antisymmetry of $\leqslant$, it is enough to prove two inequalities. Let's focus on the proof of $lcm_{\mathbb{Q}_+^\star}\left(a, lcm_{\mathbb{Q}_+^\star}(b,c)\right) \leqslant lcm_{\mathbb{Q}_+^\star}\left(lcm_{\mathbb{Q}_+^\star}(a,b), c\right)$, the proof of the other inequality being similar. Let's denote $m \triangleq lcm_{\mathbb{Q}_+^\star}\left(lcm_{\mathbb{Q}_+^\star}(a,b), c\right)$ By Lemma 2, there exist k such that $m = k\,lcm_{\mathbb{Q}_+^\star}(a,b)$ and k' such that $lcm_{\mathbb{Q}_+^\star}(a,b) = k'\,a$ and k'' such that $lcm_{\mathbb{Q}_+^\star}(a,b) = k''\,b$ and k''' such that $m = k'''\,c$. Thus $m = k\,k'\,a = k\,k''\,b = k'''\,c$. Lemma 11 gives some k'''' such that $m = k''''\ lcm_{\mathbb{Q}_+^\star}(b,c)$ and Lemma 12 concludes.

Lemma 14 (`dvdq_biglcm`). *For all $n \in \mathbb{N}$, $d \in (\mathbb{Q}_+^\star)^n$, $P \subseteq [0;n-1]$ and for all $i \in P$, it exists $k \in \mathbb{N}$ such that $\underset{j\in P}{lcm_{\mathbb{Q}_+^\star}}(d_j) = kd_i$.*

Proof. If P is a singleton then $k \triangleq 1$ works, otherwise, by commutativity and associativity of $lcm_{\mathbb{Q}_+^\star}$, we have $\underset{j\in P}{lcm_{\mathbb{Q}_+^\star}}(d_j) = lcm_{\mathbb{Q}_+^\star}\left(d_i,\ \underset{j\in P, j\neq i}{lcm_{\mathbb{Q}_+^\star}}(d_j)\right)$ and Lemma 2 concludes.

Lemma 15 (UPP extension, `UPP_extension`). *For all $f \in \mathcal{F}_{UPP}$ with $T \in \mathbb{Q}_+, d \in \mathbb{Q}_+^\star$ and $c \in \mathbb{Q}$ respectively initial segment, period and increment, for all $k \in \mathbb{N}^\star$, f is also UPP with $T' \geqslant T$, kd and kc respectively initial segment, period and increment.*

Proof. Let's assume that we have $f \in \mathcal{F}_{\mathrm{UPP}}$ with T,d and c such that : $\forall t \geqslant T, f(t+d) = f(t)+c$. Let k be a natural number different from 0 and let's prove that

$$\forall t \geqslant T', f(t+kd) = f(t)+kc$$

By induction on k, first case is the definition itself of UPP. Then let's assume that is true for k, we have:

$$\begin{aligned}\forall t \geqslant T, f(t+(k+1)d) &= f(t+kd+d)\\ &= f(t+kd)+c\\ &= f(t)+kc+c = f(t)+(k+1)c.\end{aligned}$$

Proof (of Lemma 3, `F_UPP_n_add`). Set $\tilde{T} \triangleq \max_i\{T_i\}$, $\tilde{d} \triangleq \underset{i}{lcm}_{\mathbb{Q}_+^\star}(d_i)$ and $\tilde{c} \triangleq \tilde{d}\sum_i \frac{c_i}{d_i}$. Let's first prove that for all i and for all $t \geqslant \tilde{T}$:

$$f_i\left(t+\tilde{d}\right) = f_i(t) + \tilde{d}\,\frac{c_i}{d_i}.$$

From Lemma 14, we have k such that $\tilde{d} = k\,d_i$. So, since $t \geqslant T_i$, according to Lemma 15, we have:

$$f_i(t+\tilde{d}) = f_i(t+k\,d_i) = f_i(t)+k\,c_i = f_i(t)+k\,d_i\,\frac{c_i}{d_i} = f_i(t)+\tilde{d}\,\frac{c_i}{d_i}.$$

Thus, for all $t \geqslant \tilde{T}$:

$$\begin{aligned}\left(\sum_i f_i\right)\left(t+\tilde{d}\right) &= \sum_i f_i(t) + \tilde{d}\,\frac{c_i}{d_i}\\ &= \left(\sum_i f_i\right)(t) + \tilde{d}\sum_i \frac{c_i}{d_i}\\ &= \left(\sum_i f_i\right)(t) + \tilde{c}.\end{aligned}$$

Proof (of Lemma 4, `F_UPP_n_min`). We know that it exists $i_0 \in I$ such that: $\forall t \in [T_{i_0}; T_{i_0}+d_{i_0}[\,, f_{i_0}(t) \leqslant M + s\,t$. Let's prove that: $\forall t \geqslant T_{i_0}, f_{i_0}(t) \leqslant M + s\,t$. Let $t \geqslant T_{i_0}$ and define $k \triangleq \left\lfloor \frac{t-T_{i_0}}{d_{i_0}} \right\rfloor$. Thus $t - k\,d_{i_0} \in [T_{i_0}, T_{i_0}+d_{i_0}[$ and, according to Lemma 15:

$$\begin{aligned}f_{i_0}(t) &= f_{i_0}\left((t-k\,d_{i_0})+k\,d_{i_0}\right) = f_{i_0}(t-k\,d_{i_0})+k\,c_{i_0}\\ &\leqslant M + s(t-k\,d_{i_0}) + k\,c_{i_0} = M + s\,t\end{aligned}$$

since $s = \frac{c_{i_0}}{d_{i_0}}$.

Furthermore, we know that: $\forall i \notin I, \forall t \in [T_i; T_i+d_i[\,, m_i + \frac{c_i}{d_i}t \leqslant f_i(t)$. Let's prove that: $\forall i \notin I, \forall t \geqslant T_i, m_i + \frac{c_i}{d_i}t \leqslant f_i(t)$. Let $i \notin I$ and $t \geqslant T_i$ and define $k \triangleq \left\lfloor \frac{t-T_i}{d_i} \right\rfloor$. Thus $t - k\,d_i \in [T_i, T_i+d_i[$ and, according to Lemma 15:

$$\begin{aligned}f_i(t) &= f_i\left((t-k\,d_i)+k\,d_i\right) = f_i(t-k\,d_i)+k\,c_i\\ &\geqslant m_i + \frac{c_i}{d_i}(t-k\,d_i) + k\,c_i = m_i + \frac{c_i}{d_i}\,t.\end{aligned}$$

For all $t \geqslant \tilde{T}$, for all $i \notin I$, we have $t \geqslant \frac{M-m_i}{\frac{c_i}{d_i}-s}$ and since $s < \frac{c_i}{d_i}$ we have $\left(\frac{c_i}{d_i} - s\right) t \geqslant M - m_i$. So $M + s\,t \leqslant m_i + \frac{c_i}{d_i} t$. Combined with previous results, this means that $f_{i_0}(t) \leqslant f_i(t)$. Thus

$$\forall t \geqslant \tilde{T}, \min_{i=0}^{n-1}\{f_i(t)\} = \min_{i \in I}\{f_i(t)\}.$$

Proving that for all $t \geqslant \tilde{T}$ and for all $i \in I$ we have $f_i\left(t + \tilde{d}\right) = f_i(t) + \tilde{c}$ will then conclude the proof. According to Lemma 14 there is k such that $\tilde{d} = k\,d_i$ so, according to Lemma 15:

$$\begin{aligned} f_i\left(t + \tilde{d}\right) &= f_i(t + k\,d_i) = f_i(t) + k\,c_i = f_i(t) + k\,d_i\frac{c_i}{d_i} \\ &= f_i(t) + \tilde{d}\,s = f_i(t) + \tilde{c}. \end{aligned}$$

Lemma 16 (`F_UPP_conv_f1_f1'`, `F_UPP_conv_f2_f1'`, `F_UPP_conv_f1_f2'`, `F_UPP_conv_f2_f2'`, `F_UPP_conv_aux`)**.** *Let $f, f' \in \mathcal{F}_{UPP}$ functions with initial segments $T, T' \in \mathbb{Q}_+^\star$, periods $d, d' \in \mathbb{Q}_+^\star$ and increments $c, c' \in \mathbb{Q}$ respectively. The convolution of f and f' satisfies*

$$f * f' = \min(f_1 * f_1', f_2 * f_1', f_1 * f_2', f_2 * f_2') \tag{24}$$

where

$$f_1 \triangleq t \mapsto \begin{cases} f(t) & \text{when } t < T \\ +\infty & \text{otherwise} \end{cases}$$

$$f_2 \triangleq t \mapsto \begin{cases} f(t) & \text{when } t \geqslant T \\ +\infty & \text{otherwise} \end{cases}$$

f_1' and f_2' are defined similarly, replacing f by f'.

Each term of the minimum above is UPP

- *$f_1 * f_1'$ from $T + T'$ with period d and increment c*
- *$f_2 * f_1'$ from $T + T'$ with period d and increment c*
- *$f_1 * f_2'$ from $T + T'$ with period d' and increment c'*
- *$f_2 * f_2'$ from $T + T' + lcm_{\mathbb{Q}_+^\star}(d, d')$ with period $lcm_{\mathbb{Q}_+^\star}(d, d')$ and increment $lcm_{\mathbb{Q}_+^\star}(d, d')\,min\left(\frac{c}{d}, \frac{c'}{d'}\right)$.*

Proof. This is the same as Proposition 6 in [7].

First, by distributivity of $*$ over min

$$\begin{aligned} f * f' &= \min(f_1, f_2) * \min(f_1', f_2') \\ &= \min\{f_1 * f_1', f_2 * f_1', f_1 * f_2', f_2 * f_2'\}. \end{aligned}$$

Then, let's prove that for all $t \geqslant T + T'$, we have $(f_1 * f_1')(t) = +\infty$. Indeed, $(f_1 * f_1')(t) = \inf_{u+v=t}\{f_1(u) + f_1'(v)\}$ and either $u \geqslant T$ (so $f_1(u) = +\infty$) or $v \geqslant T'$ (so $f_1'(v) = +\infty$).

Let's prove that for all $t \geqslant T + T'$, we have $(f_2 * f_1')(t+d) = (f_2 * f_1')(t) + c$. For all $t \geqslant T + T'$:

$$\begin{aligned}(f_2 * f_1')(t+d) &= \inf_{0 \leqslant u \leqslant T'} \{f_2(t+d-u) + f_1'(u)\} \\ &= \inf_{0 \leqslant u \leqslant T'} \{f_2(t-u) + c + f_1'(u)\} \\ &= (f_2 * f_1')(t) + c.\end{aligned}$$

$(f_1 * f_2')$ is similar.

Let's prove that for all $t \geqslant T + T' + lcm_{\mathbb{Q}_+^\star}(d, d')$, we have:

$$(f_2 * f_2')(t + lcm_{\mathbb{Q}_+^\star}(d,d')) = (f_2 * f_2')(t) + lcm_{\mathbb{Q}_+^\star}(d,d') \min\left(\frac{c}{d}, \frac{c'}{d'}\right).$$

For all $t \geqslant T + T' + lcm_{\mathbb{Q}_+^\star}(d, d')$:

$$\begin{aligned}&(f_2 * f_2')(t + lcm_{\mathbb{Q}_+^\star}(d,d')) \\ &= \inf_{T \leqslant u \leqslant t + lcm_{\mathbb{Q}_+^\star}(d,d') - T'} \left\{f_2(u) + f_2'(t + lcm_{\mathbb{Q}_+^\star}(d,d') - u)\right\} \\ &= \min\left(\inf_{T \leqslant u \leqslant t - T'} \left\{f_2(u) + f_2'(t + lcm_{\mathbb{Q}_+^\star}(d,d') - u)\right\},\right. \\ &\qquad \left.\inf_{T + lcm_{\mathbb{Q}_+^\star}(d,d') \leqslant u \leqslant t - T' + lcm_{\mathbb{Q}_+^\star}(d,d'))} \left\{f_2(u) + f_2'(t + lcm_{\mathbb{Q}_+^\star}(d,d') - u)\right\}\right) \\ &= \min\left(\inf_{T \leqslant u \leqslant t - T'} \left\{f_2(u) + f_2'(t + lcm_{\mathbb{Q}_+^\star}(d,d') - u)\right\},\right. \\ &\qquad \left.\inf_{T' \leqslant v \leqslant t - T} \left\{f_2(t + lcm_{\mathbb{Q}_+^\star}(d,d') - v) + f_2'(v)\right\}\right) \\ &= \min\left(\inf_{T \leqslant u \leqslant t - T'} \left\{f_2(u) + f_2'(t-u) + \frac{c'}{d'} lcm_{\mathbb{Q}_+^\star}(d,d')\right\},\right. \\ &\qquad \left.\inf_{T' \leqslant v \leqslant t - T} \left\{f_2(t-v) + \frac{c}{d} lcm_{\mathbb{Q}_+^\star}(d,d') + f_2'(v)\right\}\right) \\ &= \min\left((f_2 * f_2')(t) + \frac{c'}{d'} lcm_{\mathbb{Q}_+^\star}(d,d'), (f_2 * f_2')(t) + \frac{c}{d} lcm_{\mathbb{Q}_+^\star}(d,d')\right) \\ &= (f_2 * f_2')(t) + \min\left(\frac{c}{d}, \frac{c'}{d'}\right) lcm_{\mathbb{Q}_+^\star}(d,d').\end{aligned}$$

Proof (of Lemma 5, `F_UPP_conv`). According to Lemma 16, and reusing its notations, it is enough to prove that the following function is UPP:

$$\min\{f_1 * f_1', f_2 * f_1', f_1 * f_2', f_2 * f_2'\}.$$

When $\frac{c}{d} = \frac{c'}{d'}$, according to Lemma 4, it is enough that:

$$\forall t \in \left[T + T' + lcm_{\mathbb{Q}_+^\star}(d,d'), T + T' + 2\, lcm_{\mathbb{Q}_+^\star}(d,d')\right[, (f_2 * f_2')(t) \leqslant M + \frac{c}{d}t.$$

For any $t \in \Big[T+T'+lcm_{\mathbb{Q}_+^\star}(d,d'), T+T'+2\,lcm_{\mathbb{Q}_+^\star}(d,d')\Big[$, we have:

$$\begin{aligned}(f_2 * f_2')(t) - \frac{c}{d}t &= \inf_{u+v=t}\{f_2(u)+f_2'(v)\} - \frac{c}{d}t \\ &\leqslant f(t-T') - \frac{c}{d}t + f'(T') \\ &\leqslant \sup_{t' \in \left[T+lcm_{\mathbb{Q}_+^\star}(d,d'), T+2lcm_{\mathbb{Q}_+^\star}(d,d')\right[} \left\{f(t') - \frac{c}{d}(t'+T')\right\} + f'(T') \\ &\leqslant \sup_{t' \in [T,T+d[} \left\{f(t') - \frac{c}{d}(t'+T')\right\} + f'(T') \\ &\leqslant M.\end{aligned}$$

When $\frac{c}{d} < \frac{c'}{d'}$, according to Lemma 4, it is enough that:

$$\forall t \in \Big[T+T'+lcm_{\mathbb{Q}_+^\star}(d,d'), T+T'+2\,lcm_{\mathbb{Q}_+^\star}(d,d')\Big[, (f_2 * f_2')(t) \leqslant M + \frac{c}{d}t$$

and

$$\forall t \in [T+T', T+T'+d'[, m' + \frac{c}{d}t \leqslant (f_1 * f_2')(t).$$

The former is proved just as above, let's focus on the latter. For any $t \in [T+T', T+T'+d'[$:

$$\begin{aligned}(f_1 * f_2')(t) - \frac{c'}{d'}t &= \inf_{u+v=t}\left\{f_1(u) - \frac{c'}{d'}u + f_2'(v) - \frac{c'}{d'}v\right\} \\ &= \inf_{u+v=t, u<T, v\geqslant T'}\left\{f(u) - \frac{c'}{d'}u + f(v) - \frac{c'}{d'}v\right\} \\ &\geqslant \inf_{u\in[0,T[}\left\{f(u) - \frac{c'}{d'}u\right\} + \inf_{v\in[T',T'+d'[}\left\{f(v) - \frac{c'}{d'}v\right\} \\ &\geqslant m'.\end{aligned}$$

When $\frac{c'}{d'} < \frac{c}{d}$, the proof is similar to the previous case.

C Stability of UPP-PA Functions by *(min, plus)* Operators

Lemma 17 (Union JS, `union_JS_of_1`). *For all $f \in \mathcal{F}$ and $a, a' \in JS$ such that $a \in JS(f)$ and $last(a) = last(a')$, we have $(a \cup a') \in JS(f)$.*

Proof. Let i a natural number. Let's prove that f is *affine* on $](a \cup a')_i; (a \cup a')_{i+1}[$.

We know that it exists j such that $](a \cup a')_i; (a \cup a')_{i+1}[\subseteq]a_j; a_{j+1}[$. By hypotheses, f is *affine* on $]a_j; a_{j+1}[$ and so it is on $](a \cup a')_i; (a \cup a')_{i+1}[$.

Lemma 18 (JS of add, `JS_of_add`). *For two PA functions f and f' with $a \in JS(f)$ and $a' \in JS(f')$ such that $last(a) = last(a')$ then $a \cup a' \in JS(f+f')$.*

Proof. Let $i \in \mathbb{N}$. Using lemma 17, we know that $(a \cup a') \in JS(f)$ and $last(a) = last(a')$. Similarly with commutativity, we have $(a \cup a') \in JS(f')$. Then, with $last(a) = last(a')$ we know that f and f' are *affine* on $](a \cup a')_i; (a \cup a')_{i+1}[$ and so is $f + f'$.

Proof (of Lemma 6). `F_PA_n_add`] Let $n \in \mathbb{N}^\star$ and $f \in \mathcal{F}^n$. By induction on n, initial case is proved using $a_1 \in JS(f_1)$. Then, using Lemma 18, we have

$$a_{n+1}\left(\bigcup_i^n a_i\right) \in JS\left(f_{n+1} + \sum_i^n f_i\right)$$

and concludes the proof.

Lemma 19 (Subset of PA functions at t). *Let f be a PA function with $a \in JS(f)$. For i a natural number $x, y \in \mathbb{Q}_+$ such that $x < y$ and $]x; y[\subseteq]a_i; a_{i+1}[$, then for any $t \in]a_i; a_{i+1}[$*

- *if $f(t) = +\infty$ then $\forall t' \in]x; y[\,, f(t') = +\infty$ else*
- *if $f(t) = -\infty$ then $\forall t' \in]x; y[\,, f(t') = -\infty$*
- *otherwise, $f(t)$ is (ρ, σ)-affine on $]a_i; a_{i+1}[$ and*

$$\exists \sigma' \in \mathbb{Q}, \forall t' \in]x; y[\,, f(t') = \rho(t' - x) + \sigma'$$

Proof. Let $f \in \mathcal{F}_{\text{PA}}$ with $a \in JS(f)$, $i \in \mathbb{N}$ and $x, y \in \mathbb{Q}_+$ such that $x < y$ and $]x; y[\subseteq]a_i; a_{i+1}[$.

The first two cases are simple. For any $t \in]a_i; a_{i+1}[$ we know that $f(t) = +\infty$ or $f(t) = -\infty$. Then, obviously, for all $t' \in]x; y[\subseteq]a_i; a_{i+1}[$, $f(t') = +\infty$ and $f(t') = -\infty$ respectively.

For the case where, for any $t \in]a_i; a_{i+1}[$, $f(t)$ is rational meaning that it exists ρ and σ such that f is (ρ, σ)-*affine* on $]a_i; a_{i+1}[$. Let's define $\sigma' \triangleq \sigma + \rho(x - a_i)$.

We know that: $\forall t \in]a_i; a_{i+1}[\,, f(t) = \rho(t - a_i) + \sigma$ so, we have

$$\begin{aligned} \forall t' \in]x; y[\,, & f(t') = \rho(t' - x) + \sigma + \rho(x - a_i) \\ & f(t') = \rho(t' - a_i) + \sigma \end{aligned}$$

Lemma 20 ($\cup_{\min}$ Property). *Let $f, f' \in \mathcal{F}_{PA}$ with $a \in JS(f)$ and $a' \in JS(f')$, define $c \triangleq \cup_{\min}(f, f', a, a')$ and let $i \in \mathbb{N}$. If $c_i \notin (a \cup a')$, then*

$$(\forall t \in]c_i; c_{i+1}[\,, f(t) < f'(t)) \vee (\forall t \in]c_i; c_{i+1}[\,, f'(t) < f(t))$$

Proof. For $c \triangleq \cup_{\min}(f, f', a, a')$, with the hypothesis $c_i \notin (a \cup a')$ and Definition 9, we know that

f is (ρ, σ)-*affine* on $]c_i; c_{i+1}[$
f' is (ρ', σ')-*affine* on $]c_i; c_{i+1}[$

$\rho \neq \rho'$ and, for $x \triangleq c_i + \frac{\sigma'-\sigma}{\rho-\rho'}$, we have $x \in]c_i; c_{i+1}[$ and c_{i+1}.

By case disjunction on $\rho \neq \rho'$, we first assume that $\rho < \rho'$. By definition of c_{i+1}, $c_{i+1} \leqslant (a \cup a')_{k+1}$ so $]c_i; c_{i+1}[\subseteq](a \cup a')_k; (a \cup a')_{k+1}[$. So for all $t \in]c_i; c_{i+1}[$:

$$\begin{aligned} f'(t) - f(t) &= (\rho' - \rho)(t - (a \cup a')_k) + \sigma' - \sigma \\ &= (\rho' - \rho)(t - c_{i'}) + \sigma' - \sigma \\ &= (\rho' - \rho)\left(t - \left(c_{i'} + \frac{\sigma' - \sigma}{\rho - \rho'}\right)\right) \\ &= \underbrace{(\rho' - \rho)}_{>0}\underbrace{(t - c_i)}_{>0} \end{aligned}$$

so $\forall t \in]c_i; c_{i+1}[\,, f'(t) > f(t)$.

Then, let's assume that $\rho' < \rho$. Similarly, $\forall t \in]c_i; c_{i+1}[\,, f(t) > f'(t)$ and concludes the proof.

Lemma 21 ($\cup_{\min}$ Property). *Let f and f' be two PA function with $a \in JS(f)$ and $a' \in JS(f')$ and set $c \triangleq \cup_{\min}(f, f', a, a')$. For $i \in \mathbb{N}$, if it exists $\rho, \rho', \sigma, \sigma' \in \mathbb{Q}$ such that $\rho \neq \rho'$ and f and f' are respectively $(\rho, \sigma)-$affine and $(\rho', \sigma')-$affine on $]c_i; c_{i+1}[$, then:*

$$(\forall t \in]c_i; c_{i+1}[\,, f(t) < f'(t)) \vee (\forall t \in]c_i; c_{i+1}[\,, f'(t) < f(t))$$

Proof. Define $c \triangleq \cup_{\min}(f, f', a, a')$ and assume that it exists $\rho, \rho', \sigma, \sigma'$ such that $\rho \neq \rho'$ and f and f' are respectively (ρ, σ)-*affine* and (ρ', σ')-*affine* on $]c_i; c_{i+1}[$.

First assume that $c_i \notin (a \cup a')$. The direct application of Lemma 20 concludes.

Let's then assume that $c_i \in (a \cup a')$. and define $x \triangleq c_i + \frac{\sigma'-\sigma}{\rho-\rho'}$. For all $t \in]c_i; c_{i+1}[$ we have:

$$\begin{aligned} f(t) - f'(t) &= (\rho - \rho')(t - c_i) + \sigma - \sigma' \\ &= (\rho - \rho')\left(t - \left(c_i + \frac{\sigma' - \sigma}{\rho - \rho'}\right)\right) \\ &= (\rho - \rho')(t - x) \end{aligned}$$

By definition, $x \notin]c_i; c_{i+1}[$ since x is a point added in c if $c_i < c_i + \frac{\sigma'-\sigma}{\rho-\rho'} < c_{i+1}$.

Let's first assume that $x \leqslant c_i$. As $c_i < t$, we have $0 < t - x$. So, let distinguish two cases from $\rho \neq \rho'$. If $\rho < \rho'$ then: $\forall t \in]c_i; c_{i+1}[\,, f(t) < f'(t)$. Otherwise, $\rho' < \rho$ and: $\forall t \in]c_i; c_{i+1}[\,, f'(t) < f(t)$.

Otherwise, $c_{i+1} \leqslant x$. As $t < c_{i+1}$, we have $t - x < 0$. Let's distinguish two cases from $\rho \neq \rho'$. First is $\rho < \rho'$. Then, for all $t \in]c_i; c_{i+1}[\,, f'(t) < f(t)$. Otherwise, for all $t \in]c_i; c_{i+1}[\,, f(t) < f'(t)$.

Lemma 22 (JS of min, `JS_of_min`). *Let f and $f' \in \mathcal{F}$ with $a \in JS(f)$ and $a' \in JS(f')$, if $last(a) = last(a')$ then we have $\cup_{\min}(f, f', a, a') \in JS(\min(f, f'))$.*

Proof. Let i be a natural number and pose $c \triangleq \cup_{\min}(f, f', a, a')$. Assume that $last(a) = last(a')$ and pose $m_i \triangleq \frac{c_i + c_{i+1}}{2}$.

Then, let's assume that $f(m_i) = -\infty$. By Definition, we know that for j a natural number, we have $]c_i; c_{i+1}[\subseteq](a \cup a')_j; (a \cup a')_{j+1}[$ and by using Lemma 17, we have j' such that

$$]c_i; c_{i+1}[\subseteq](a \cup a')_j; (a \cup a')_{j+1}[\subseteq]a_{j'}; a_{j'+1}[$$

So, by using Definition of PA functions, we have $\forall t \in]c_i; c_{i+1}[, f(t) = -\infty$ and so $\forall t \in]c_i; c_{i+1}[, \min(f, f')(t) = -\infty$ since $-\infty$ is absorbing for min.

By using commutativity of min, we have the same reasoning for $f'(m_i) = -\infty$.

Then, let's assume that $f(m_i) = +\infty$. By using the same previous reasoning, we have $\forall t \in]c_i; c_{i+1}[, f(t) = +\infty$. It remains two cases possible for $f'(m_i)$. First is $f'(m_i) = +\infty$: we have $\forall t \in]c_i; c_{i+1}[, f'(t) = +\infty$ and so $\forall t \in]c_j; c_{j+1}[, \min(f, f)(t) = +\infty$.

Second, if $f'(m_i) \in \mathbb{R}$, then for a j' such that $]c_i; c_{i+1}[\subseteq]a'_{j'}; a'_{j'+1}[$, it exists ρ' and σ' such that f' is $(\rho', \sigma') - affine$ on $]c_i; c_{i+1}[$. Then, g is also $(\rho', \sigma') - affinei$ on $]c_i; c_{i+1}[$.

We have the same reasoning for $f'(m_i) = +\infty$ by using commutativity of min.

Finally, if $f(m_i) \in \mathbb{R}$, using the Definition of $\cup_{\min}$ and Lemma 17, we have $\rho, \sigma \in \mathbb{Q}$ such that f is (ρ, σ)-*affine* on $]c_i; c_{i+1}[$. The only case remaining for $f'(m_i)$ is $f'(m_i) \in \mathbb{R}$ and so, with the same use of Lemma 17, it exists $\rho', \sigma' \in \mathbb{Q}$ such that f' is (ρ', σ')-*affine* on $]c_i; c_{i+1}[$.

We want to prove that exists $\tilde{\rho}$ and $\tilde{\sigma}$ such that, on the interval $]c_i; c_{i+1}[$, the function $g \triangleq \min(f, f')$ is $(\tilde{\rho}, \tilde{\sigma})$-*affine* on $]c_i; c_{i+1}[$.

We want to prove that exists $\tilde{\rho}$ and $\tilde{\sigma}$ such that, on the interval $]c_i; c_{i+1}[$, the function $g \triangleq f \wedge f'$ is $(\tilde{\rho}, \tilde{\sigma})$-*affine* on $]c_i; c_{i+1}[$.

Let's first assume that $\rho \neq \rho'$. So, using Lemma 21, we know that:

$$(\forall t \in]c_i; c_{i+1}[, f(t) < f'(t)) \vee (\forall t \in]c_i; c_{i+1}[, f'(t) < f(t))$$

So, in the first case, $\tilde{\rho} \triangleq \rho, \tilde{\sigma} \triangleq \sigma$ and in the other case $\tilde{\rho} \triangleq \rho', \tilde{\sigma} \triangleq \sigma'$.

Let's now assume that $\rho = \rho'$. So, let's define $\tilde{\rho} \triangleq \rho$ and $\tilde{\sigma} \triangleq \sigma \wedge \sigma'$. Since $\wedge$ is distributive over $+$, we have

$$\begin{aligned} \forall t \in]c_i; c_{i+1}[, g(t) &= \min\{\rho(t - c_i) + \sigma, \rho(t - c_i) + \sigma'\} \\ &= \rho(t - c_i) + \min\{\sigma, \sigma'\} \end{aligned}$$

and that concludes the proof.

Lemma 23 (JS for n-ary minimum, `JS_of_n_min_aux`). *For all $n \in \mathbb{N}^\star$, $f \in \mathcal{F}^n_{PA}$ and $a \in JS^{n \times n}$ such that for all i, j, $a_{i,j} \in JS(\min(f_i, f_j))$. We have:*

$$\left(\bigcup_{(i,j) \in [0;n-1]^2} a_{i,j} \right) \in JS\left(\min_i f_i \right). \tag{25}$$

Proof. Define $c \triangleq \bigcup_{(i,j)\in[0;n-1]^2}(a_{i,j})$ and $F \triangleq \min_i(f_i)$. We want to prove that $c \in JS(F)$ that is, for all $k < \#c - 1$:

$$\begin{aligned}&(\forall t \in]c_k; c_{k+1}[\,, F(t) = +\infty)\\ \vee\ &(\forall t \in]c_k; c_{k+1}[\,, F(t) = -\infty)\\ \vee\ &(\exists \rho, \sigma, \forall t \in]c_k; c_{k+1}[\,, F(t) = \rho(t - c_k) + \sigma)\,.\end{aligned}$$

Let $k < \#c - 1$ and define the interval $I_k \triangleq]c_k; c_{k+1}[$ and a point $t_0 \triangleq \frac{c_k + c_{k+1}}{2}$. First assume that $F(t_0) = -\infty$. So, it exists i_0 such that $f_{i_0}(t_0) = -\infty$. By using Lemma 17 and since $c = a_{i_0,i_0} \cup \left(\bigcup_{(i,j)\neq(i_0,i_0)}(a_{i,j})\right)$, we know that it exists k_{i_0} such that:

$$I_k \subseteq \left](a_{i_0,i_0})_{k_{i_0}}; (a_{i_0,i_0})_{k_{i_0}+1}\right[.$$

By definition of PA functions and since a_{i_0,i_0} is a JS for f_{i_0}, we have:

$$\forall t \in \left](a_{i_0,i_0})_{k_{i_0}}; (a_{i_0,i_0})_{k_{i_0}+1}\right[, f_{i_0}(t) = -\infty$$

and so, $\forall t \in I_k, f_{i_0}(t) = -\infty$. Thus

$$\forall t \in I_k, F(t) = -\infty$$

and that concludes that case.

Then assume that $F(t_0) = +\infty$. So, for all i, $f_i(t_0) = +\infty$. Let $i \in [0; n-1]$. Using the same reasoning as in the previous case, it exists $k_i \in \mathbb{N}$ such that $I_k \subseteq](a_{i,i})_{k_i}; (a_{i,i})_{k_i+1}[$ and

$$\forall t \in I_k, f_i(t) = +\infty.$$

So, we have, for all $t \in I_k$, $F(t) = +\infty$ and it concludes this case.

Finally assume that $F(t_0) \in \mathbb{R}$. Then it exists i such that $F(t_0) = f_i(t_0)$ and by Definition, with the definition of PA and Lemma 19, we know that for some $\rho_i \in \mathbb{Q}$ and $\sigma_i \in \mathbb{Q}$, f_i is (ρ_i, σ_i)-*affine* on I_k. To conclude this case and the proof, let's prove that:

$$\forall j \in [0; n-1]\,, \forall t \in I_k, f_i(t) \leqslant f_j(t) \tag{26}$$

Let j be a natural number. First assume that $f_j(t_0) = +\infty$. Since f_j is PA, by Definition and Lemma 19, we can conlude that $\forall t \in I_k, f_j(t) = +\infty \geqslant f_i(t)$. Then assume that $f_j(t_0) = -\infty$. This case is resolved by contradiction since we know that $F(t_0) \in \mathbb{R}$ and $F(t_0) \leqslant f_j(t_0)$.

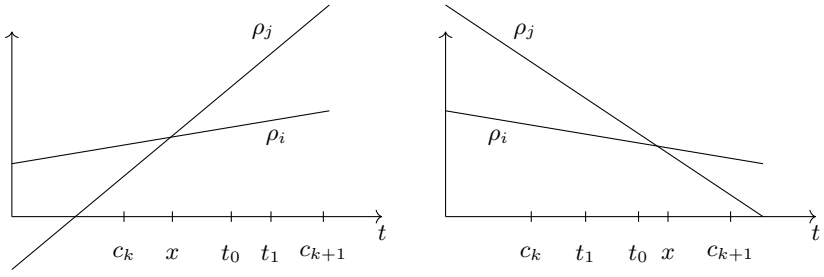
Finally, we have the case where $f_j(t_0) \in \mathbb{R}$. We know that $\min(f_i, f_j)$ is a PA function with $a_{ij} \in JS(\min(f_i, f_j))$. By Definition and 19, we know that it exists ρ_j and σ_j such that f_j is (ρ_j, σ_j)-*affine* on I_k. So we want to prove that:

$$\forall t \in I_k, \rho_i(t - c_k) + \sigma_i \leqslant \rho_j(t - c_k) + \sigma_j.$$

First prove that $\forall t \in I_k, \min(f_i, f_j) \in \mathbb{R}$. The cases $\forall t \in I_k, \min(f_i, f_j) = +\infty$ and $\forall t \in I_k, \min(f_i, f_j) = -\infty$ are impossible since f_i and f_j are resp. $(\rho_i,$

σ_i)-*affine* and (ρ_j, σ_j)-*affine* on I_k. So, for some $\rho_{i,j}, \sigma_{i,j} \in \mathbb{Q}$, we know that $\min(f_i, f_j)$ is ($\rho_{i,j}$, $\sigma_{i,j}$)-*affine* on I_k.

Then, let's do a disjunction on the order between ρ_i and ρ_j. First assume that $\rho_i = \rho_j$. We know that $F(t_0) = f_i(t_0) \leqslant f_j(t_0)$, and so $\sigma_i \leqslant \sigma_j$. This case is concluded with compatibility of addition.



(a) $\rho_i < \rho_j$ (b) $\rho_j < \rho_i$

Fig. 6: Illustration of impossible cases of x and c_k

Then assume that $\rho_i < \rho_j$. Define a point $x \triangleq c_k + \frac{\sigma_i - \sigma_j}{\rho_j - \rho_i}$ such that $f_i(x) = f_j(x)$. An illustration is given in Figure 6a.

From $f_i(t_0) \leqslant f_j(t_0)$, we have:

$$\begin{aligned} \rho_i(t_0 - c_k) + \sigma_i \leqslant \rho_j(t_0 - c_k) + \sigma_j &\implies \sigma_i - \sigma_j \leqslant (\rho_j - \rho_i)(t_0 - c_k) \\ &\implies x \leqslant t_0 \text{ since } \rho_i < \rho_j \end{aligned}$$

Let's define a point $t_1 \triangleq \frac{t_0 + c_{k+1}}{2}$. By definition we have $t_0 \leqslant t_1$ and with $x \leqslant t_0$, we finally obtain $x \leqslant t_1$.

$$\begin{aligned} x \leqslant t_1 \Leftrightarrow c_k + \frac{\sigma_i - \sigma_j}{\rho_j - \rho_i} &\leqslant t_1 \\ \implies \sigma_i - \sigma_j &\leqslant (t_1 - c_k)(\rho_j - \rho_i) \\ \implies f_i(t_1) &\leqslant f_j(t_1) \\ \implies f_i(t_1) &= \min(f_i, f_j)(t_1) \end{aligned}$$

Knowing that $f_i(t_0) = \min(f_i, f_j)(t_0)$, we can establish this equation system

$$\begin{cases} \rho_i(t_0 - c_k) + \sigma_i = \rho_{i,j}(t_0 - c_k) + \sigma_{i,j} \\ \rho_i(t_1 - c_k) + \sigma_i = \rho_{i,j}(t_1 - c_k) + \sigma_{i,j} \end{cases}$$

We subtract the first line to the second to obtain

$$\rho_i(t_1 - t_0) = \rho_{i,j}(t_1 - t_0)$$

and so $\rho_i = \rho_{i,j}$ since $t_1 \neq t_0$. We apply this results to the first or the second line to obtain $\sigma_i = \sigma_{i,j}$. We can conclude that $f_i = f_i \wedge f_j$ on I_k and $f_i \leqslant f_j$ on I_k.

Now assume that $\rho_j < \rho_i$. Define a point x as we did previously such that $\rho_i(x - c_k) + \sigma_i = \rho_j(x - c_k) + \sigma_j$, illustrated in Figure 6b.

From $f_i(t_0) \leqslant f_j(t_0)$, we have

$$\begin{aligned}\rho_i(t_0 - c_k) + \sigma_i \leqslant \rho_j(t_0 - c_k) + \sigma_j &\implies (\rho_i - \rho_j)(t_0 - c_k) \leqslant \sigma_j - \sigma_i \\ &\implies t_0 \leqslant x \text{ since } \rho_j < \rho_i\end{aligned}$$

Define a point $t_1 \triangleq \frac{c_k + t_0}{2}$. From $t_0 \leqslant x$ and $t_1 \leqslant t_0$, we have:

$$\begin{aligned}t_1 \leqslant x \Leftrightarrow t_1 &\leqslant c_k + \frac{\sigma_j - \sigma_i}{\rho_i - \rho_j} \\ &\implies f_i(t_1) \leqslant f_j(t_1) \\ &\implies f_i(t_1) = f_i(t_1) \wedge f_j(t_1)\end{aligned}$$

Knowing that $f_i(t_0) = (f_i \wedge f_j)(t_0)$ on t_0 and t_1, we have this equation system:

$$\begin{cases}\rho_i(t_0 - c_k) + \sigma_i = \rho_{i,j}(t_0 - c_k) + \sigma_{i,j} \\ \rho_i(t_1 - c_k) + \sigma_i = \rho_{i,j}(t_1 - c_k) + \sigma_{i,j}\end{cases}$$

and it is concluded as we did previously with $\rho_i = \rho_{i,j}, \sigma_i = \sigma_{i,j}$ and, in conclusion, $f_i = \min(f_i, f_j)$ on I_k and so $f_i \leqslant f_j$ on I_k.

Proof (of Lemma 7, `JS_of_n_min`). It is a direct application of Lemmas 22 and 23.

Proof (of Lemma 8). Let $t \in \mathbb{R}_+$. To prove $f_{<last(a)}(t) = \min\limits_{i<\#a-1} \left((f \downarrow a)_i\right)(t)$, first assume that $t \notin a$. Then, we have $last(a) < t$ so $f_{<last(a)}(t) = +\infty$ and, for all i, $t \notin [a_i; a_{i+1}[$ thus $(f \downarrow a)_i = +\infty$ and concludes this case.

Then assume that $t < last(t)$. So, it exists i such that $t \in [a_i; a_{i+1}[$. We obviously have $i > 0$ since a is not empty. Thus, it remains to prove that:

$$f_{<last(a)}(t) = \min\left((f \downarrow a)_i(t), \min_{j \neq i, j<\#a-1}\left((f \downarrow a)_j\right)(t)\right) \tag{27}$$

Let $j < \#a - 1$. Since $j \neq i$ and $t \in [a_i; a_{i+1}[$, we have $t \notin [a_j; a_{j+1}[$ and so $\min\limits_{j \neq i, j<\#a-1}\left((f \downarrow a)_j\right)(t) = +\infty$. Finally, $f_{<last(a)}(t) = f(t)$ and $(f \downarrow a)_i(t) = f(t)$ by Definition 10 and 11 respectively.

Lemma 24 (Cutting Below Convolution, `cutting_below_conv`). *Let $f, f' \in \mathcal{F}$ and $l \in \mathbb{Q}_+$. We have $(f * f')_{<l} = \left(f_{<l} * f'_{<l}\right)_{<l}$.*

Proof. Prove that for $t \in \mathbb{Q}_+$, $(f * f')_{<l}(t) = \left(f_{<l} * f'_{<l}\right)_{<l}(t)$. It is verified by Definition 11 for $l \leqslant t$. So, we have $t < l$. So, let's prove that:

$$\inf_{u>0, u \leqslant t}\{f(u) + f'(t-u)\} = \inf_{u>0, u \leqslant t}\{f_{<l}(u) + f'_{<l}(t-u)\}$$

which is a alternative definition of convolution. It is verified since $u > 0$ and so $t - u < l$.

Proof (of Lemma 9). Using Lemma 24 and 8, we have

$$(f * f')_{<l} = \left(\min_i \left((f \downarrow a)_i\right) * \min_i \left((f' \downarrow a')_i\right)\right)_{<l} \tag{28}$$

and the proof is concluded since $*$ is distributive over min.

D Finite Equality Criteria on UPP-PA

Lemma 25 (Inter JS Equality criterion on PA, `PA_equality_on_segment`). *Let f and f' be two PA functions. Let a and a' be JS such that $a \in JS(f)$, $a' \in JS(f')$ and $last(a) = last(a')$. We define $c \triangleq a \cup a'$. For all $i < \#c - 1$, we have:*

$$eq\_segment(c, i, f, f') \implies (\forall t \in [c_i; c_{i+1}[\,, f(t) = f'(t))$$

Proof. This follows from Definition 12.

Lemma 26 (`UPP_same_d_c_equality`). *For all $f, f' \in \mathcal{F}_{UPP}$ with initial segment T and T' respectively, the same period d and the same increment c, let $\theta \triangleq \max\{T, T'\}$. If for all $t \in [0; \theta + d[$, $f(t) = f'(t)$, then we have*

$$\forall t, f(t) = f'(t).$$

Proof. Let f and f' be two UPP function with initial segment T and T' respectively, the same period $d > 0$ and the same increment c. Let's define $\theta \triangleq \max\{T, T'\}$. Let's assume that for all $t \in [0; \theta + d[$, $f(t) = f'(t)$. Let's prove that for all $t \in \mathbb{R}_+$, we have $f(t) = f'(t)$. Let $t \in \mathbb{R}_+$.

If $t < \theta$, then by hypothesis, $f(t) = g(t)$.

If $t \geqslant \theta$, then let's define $k \in \mathbb{N}$ a natural number such that:

$$k \triangleq \left\lfloor \frac{t - \theta}{d} \right\rfloor$$

Then, because $\forall a \in \mathbb{R}, \lfloor a \rfloor \leqslant a < \lfloor a \rfloor + 1$, we have $k \leqslant \frac{t-\theta}{d} < k + 1$. Since $0 < d$, we have $kd \leqslant t - \theta < (k + 1)d$ and finally

$$\theta \leqslant t - kd < \theta + d$$

Thus, according to Lemma 15:

$$\begin{aligned} f(t) &= f((t - kd) + kd) = f(t - kd) + kc \\ &= f'(t - kd) + kc = f'((t - kd) + kd) = f'(t). \end{aligned}$$

Lemma 27 (`UPP_equality`). *For all $f, f' \in \mathcal{F}_{UPP}$ with initial segment T and T', period d and d' and increment c and c' respectively. Let $\theta \triangleq \max\{T, T'\}$ and $\delta \triangleq lcm_{\mathbb{Q}_+^\star}(d, d')$. If $\frac{c}{d} = \frac{c'}{d'}$ and for all $t \in [0; \theta + \delta[$, $f(t) = f'(t)$ then we have*

$$\forall t, f(t) = f'(t).$$

Proof. Using Lemmas 15 and 2, f and f' are UPP with same period δ and increment $\frac{c}{d}(lcm_{\mathbb{Q}_+^\star}(d, d')) = \frac{c'}{d'}(lcm_{\mathbb{Q}_+^\star}(d, d'))$. The conclusion follows from Lemma 26.

Lemma 28 (Finite Equality Criterion on $UPP - PA$, `UPP_PA_equality`). *For all $f, f' \in \mathcal{F}_{UPP\text{-}PA}$ with initial segment T and T', period d and d' and increment c and c' respectively, we define $\theta \triangleq \max\{T, T'\}$, $\delta \triangleq lcm_{\mathbb{Q}_+^\star}(d, d')$, $l \triangleq \theta + \delta$ and $u \triangleq a \cup a'$. Assume $a \in JS(f)$ and $a' \in JS(f')$ such that $last(a) = last(a') = l$. If $\frac{c}{d} = \frac{c'}{d'}$, then:*

$$\forall i \leqslant prev(a \cup a', \theta + \delta), eq\_segment(a \cup a', i, f, f')$$

is a sufficient condition for $f = f'$.

Proof. It is a direct application of Lemmas 27 and 25. Note that a and a' can be found using Lemma 1.

Proof (of Proposition 1, `UPP_PA_n_add`). From Lemma 3, $\sum_i f_i$ is UPP from $\max_i\{T_i\}$ with period $\underset{i}{lcm_{\mathbb{Q}_+^\star}}(d_i)$ and increment $\underset{i}{lcm_{\mathbb{Q}_+^\star}}(d_i)\left(\sum_i \frac{c_i}{d_i}\right)$. From Lemma 6, $\bigcup_i a_i \in JS\left(\sum_i f_i\right)$. The conclusion follows from Lemma 28.

Proof (of Proposition 2, `UPP_PA_n_min`). From Lemma 4, $\min_i(f_i)$ is UPP from $\tilde{T}$ with period $\tilde{d}$ and increment $\tilde{c}$. From Lemma 7, $u \in JS(\min_i(f_i))$. The conclusion follows from Lemma 28.

Proof (of Proposition 3, `UPP_PA_conv`). From Lemma 5, $f * f'$ is UPP from $\tilde{T}$ with period $\tilde{d}$ and increment $\tilde{c}$. From Lemma 9, $u \in JS(f * f')$. The conclusion follows from Lemma 28.