Proving the Correctness of the Standardized Algorithm for ABR Conformance

Jean-François Monin (jeanfrancois.monin@francetelecom.fr)
France Télécom R&D DTL/MSV, Technopole Anticipa
2 av. Pierre Marzin, 22307 Lannion, France

Abstract. Conformance control for ATM cells is based on a real-time reactive algorithm which delivers a value depending on inputs from the network. This value must always fit with a well defined theoretical value. We present here the correctness proof of the algorithm standardized for the ATM transfer capability called ABR. The proof turned out to produce a key argument during the standardization process of ABR.

Keywords: specification, verification, reactive system, real-time, telecommunications.

Abbreviations: ABR – Available Bit Rate; ACR – Allowed Cell Rate; ATM – Asynchronous Transfer Mode; ATC – ATM Transfer Capability; GCRA – Generic control of Cell Rate Algorithm; DGCRA – Dynamic GCRA.

1. Introduction

We present in this paper an unusual (at least to our knowledge) application of formal methods in telecommunications, though closely related to a protocol. There is now quite a long tradition in this area of using formal languages, even standardized ones. They are based on communicating (extended) finite state machines (e.g. Estelle, SDL, Promela) or process algebras (e.g. Lotos). Verification based on model checking [19, 10] or simulation [15] has also been successfully employed. Typically, one models the protocol at hand, using one of the above formalisms, and then one tries to verify that bad things like unexpected messages, deadlocks and so on never happen. To achieve this effect one may use temporal logic formulas or observers and automated verification tools. This approach turns out to be very useful because the global behavior of a system made of several concurrent components is difficult to master.

In the problem we deal here with, complexity does not lie in parallelism or message interleaving, but in a single sequential, short, real-time and reactive algorithm. This algorithm runs on a key device for an ATM Transfer Capability called ABR (see below). It handles a small scheduler and delivers a value which depends on inputs from the network. We essentially have to prove that the value delivered by the device always agrees with (more precisely: is not smaller than) a theoretical value whose computation is not feasible under realistic assumptions. Several algorithms were studied at ITU-T, all of them involving tricky combinations of tests and updates. Our correctness proof has been a key argument in favor of one of them and freed the standardization process of ABR.

Let us emphasize that the input of the work reported here was just a piece of pseudo-code with diagrammatic and intuitive explanations on what happens in that or such case. Such hints turned out very incomplete and did not seem of much help for our purpose. We then decided to forget them and to carry out a proof on a purely technical ground, reducing the problem to small steps which are easy to verify—we are still unable to explain how the algorithm actually works. However we needed to understand how it *could* work, that is, what are the theoretical properties of an ideal version of the algorithm.

The technique used is basically the calculus of weakest preconditions of Dijkstra [11]. Indeed, the invariant provided in section 4.5 was obtained by successive strenghtenings of the desired properties. Essentially, the algorithm runs transitions wich are guarded assignments (also called *generalized substitutions* in B [2]) of the form **if** C **then** $x := E$, where C is a condition, x a variable (or a tuple of variables) and E is an expression. A property P is left invariant by such a transition if P implies the corresponding weakest precondition, which is denoted in the style of B by [**if** C **then** $x := E]P$ and reduces to $C \Rightarrow [x := E]P$, where $[x := E]P$ is obtained by replacing every free occurence of x with E in P. Roughly, the calculus consists in considering an invariant I and computing $[\mathsf{S}]I$ for each generalized substitution S; if $I \Rightarrow [\mathsf{S}]I$, nothing has to be added at this stage, but in general I is not strong enough; $I \wedge [\mathsf{S}]I$ is then the next candidate for the invariant. Eventually we reach a fixpoint where $I \Rightarrow [\mathsf{S}]I$ for each S, provided enough information is embodied in the first version of I.

In our case, the invariant must express a property of the scheduler, which predicts values in the future and thus can be seen, abstractly, as a function of time: our invariant is a higher order property. Moreover, transitions are fired in real time: the current time appears as a free variable in the invariant and we have to explain how time evolves. This lead us to a framework inspired by timed automata of Alur and Dill [4], involving two kinds of transitions: discrete transitions, which change the state and leave the current time invariant, and continuous transitions, which leave the state invariant while the time increases. In this way get a model of the behaviour of the device under study, whose state, including a special variable called `ACR`, is a function of the time. The desired property involves another function of time called Acr, which is defined in a simple and purely mathematical way by a specification $\mathcal{S}_\mathrm{d}$. Our main result states that at any time t, $\mathrm{Acr}(t)$ is less than or equal to $\mathtt{ACR}(t)$.

In order to make the result as convincing as possible—a must in the context of standardization—it is important for the specification $\mathcal{S}_\mathrm{d}$ to be very simple and declarative (our proof steps are small and explicit for the same reason). We also explore consequences of $\mathcal{S}_\mathrm{d}$: they provide insights on the behaviour of the algorithm under study. $\mathcal{S}_\mathrm{d}$ turns out technically not well suited to a direct correctness proof; however we can derive an equivalent but more tractable computational specification $\mathcal{S}_\mathrm{c}$. The invariant to be proved is stated using $\mathcal{S}_\mathrm{c}$; the proof can then be carried out thanks to preliminary lemmas related to $\mathcal{S}_\mathrm{c}$ and formalized in Coq (an automated proof assistant based on type theory [6]), as reported in [16].

This case study illustrates that even on modest real-life examples, it may well appear that currently available ready-to-use formal methods are not able to cope with every aspect of the problem. However, we can use elementary mathematics to combine concepts coming from well-known formal methods and mechanically check the different steps.

The original work is published in [17]. Its formalization in Coq presented in [16] needed minor adaptations and corrections. We present here a new version of the proof which includes recent improvements. In particular, our treatment of time now conforms to the tradition initiated by Alur and Dill, at the price of somewhat subtle changes in the invariant. The new proof has again been completely checked in Coq.

The rest of this paper is organized as follows. Section 2 describes the problem as well as the stakes for telecommunications. Section 3 states the specifications called $\mathcal{S}_\mathrm{d}$ and $\mathcal{S}_\mathrm{c}$ above and explains how to get the latter from the former. Section 4 describes the state space with its invariant and sketches the main steps of the proof (the standardized algorithm and technical details of its correctness proof are respectively given in appendix A and B). We end with concluding remarks and related work in section 5.

2. Context and Motivation

2.1. Conformance Control in ATM

In an ATM (Asynchronous Transfer Mode) network, cells, i.e. data packets sent by a user, must not exceed a rate which is defined by a contract negotiated between the user and the network. Several modes for using an ATM network, called "ATM Transfer Capabilities" (ATCs) have been defined. Each ATC may be seen as a generic contract between the user and the network, saying that the network must guarantee the negotiated *quality of service* (QoS), defined by a number of characteristics like maximum cell loss or transfer delay, provided the cells sent by the user conform to the negotiated traffic parameters (for instance, their rate must be bounded by some value). The conformance of cells sent by the user is checked using an algorithm called GCRA (generic control of cell rate algorithm). In this way, the network is protected against users misbehaviors and keeps enough resources for delivering the required QoS to well behaved users.

In fact, a new ATC cannot be accepted (as an international standard) without an efficient conformance control algorithm, and some evidence that this algorithm has the intended behavior.

For the ATC called ABR (Available Bit Rate), considered here, a simple but inefficient algorithm had been proposed in a first stage. Reasonably efficient algorithms proposed later turned out to be fairly complicated. This situation has been settled when one of them, due to Christophe Rabadan, has been proved correct in relation to the simple one: this algorithm is now part of the I.371.1 standard [1]. The first version of the proof was hand written. The main invariants discovered during this process are included in I.371.1.

2.2. The Case of ABR

In some of the most recently defined ATCs, like ABR, the allowed cell rate may vary during the same session, depending on the current congestion state of the network. Such ATCs are designed for irregular sources, that need high cell rates from time to time, but that may reduce their cell rate when the network is busy. A servo-mechanism is then proposed in order to let the user know whether he can send data or not. This mechanism has to be well defined, in order to have a clear traffic contract between user and network. The key is an adaptation of the public algorithm for checking conformance of cells.

An abstract view of the protocol ABR is given in fig. 1 (actually, resource management (RM) cells are sent by the user, but only their transmission from the network to the user is relevant here; details are available in [20]). The conformance control algorithm for ABR has two parts. The first one is called DGCRA (dynamic GCRA). It just checks that the rate of data cells emitted by the user is not higher than a value called `ACR`, the allowed cell rate. Excess cells may be discarded by DGCRA. `ACR` is itself an approximation of an ideal allowed cell rate, which is denoted by Acr, but let us first consider that `ACR` and

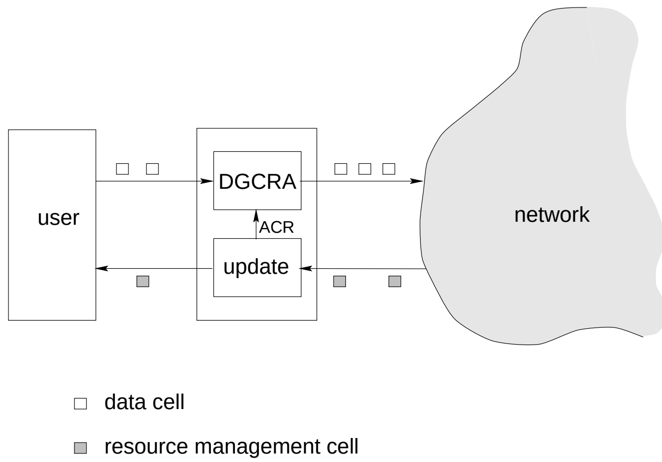

Figure 1. Conformance control

Acr are equal. In the case of ABR, Acr depends on time: its value has to be known each time a new data cell comes from the user. This part is quite simple and is not addressed here. The complexity lies in the computation of $\mathrm{Acr}(t)$ ("update" in fig. 1), which depends on the sequence of values (ER_n) carried by resource management cells coming from the network. By a slight abuse of notation, the cell carrying ER_n will be called itself ER_n.

Of course, $\mathrm{Acr}(t)$ depends only on cells ER_n whose arrival time t_n is such that $t_n < t$ (we order resource management cells so that $t_n < t_{n+1}$ for any n). In ABR, a resource management cell carries an intended allowed cell rate, that should be reached as soon as possible. At first sight, $\mathrm{Acr}(t)$ should then be simply the last ER_i received at time t, i.e. ER_i with $i = \mathrm{last}(t)$, where $\mathrm{last}(x)$ is the only integer such that $t_i \leq x < t_{i+1}$. Unfortunately, because of electric propagation time and various transmission mechanisms, the user is aware of this expected value only after a delay. Taking the user's reaction time τ observed by the control device into consideration, that is, the overall round trip time between the control device and the user, $\mathrm{Acr}(t)$ should then be ER_i with $i = \mathrm{last}(t - \tau)$. But τ may vary in turn. ITU-T considers that a lower bound τ_3 and an upper bound τ_2 for τ are established during the negotiation phase of each ABR connection. Hence, a cell arriving from the user at time t on DGCRA may legitimately have been emitted using any rate ER_i such that i is between $\mathrm{last}(t - \tau_2)$ and $\mathrm{last}(t - \tau_3)$ (see figure 2). Any rate less than or equal to any of these values, or, equivalently less than or equal to the maximum of them, should then be allowed. Therefore, $\mathrm{Acr}(t)$ is taken as the maximum of these ER_i.

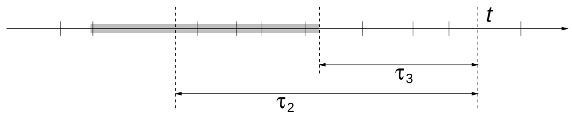

Figure 2. Cells to be taken into account at time t

Actually the standards committee did not give these explanations, but directly specified the set of ER_i under the equivalent form (2) below.

2.3. Effective Computation of $\mathrm{Acr}(t)$

ITU-T committee considered that a direct computation of $\mathrm{Acr}(t)$ is not feasible at reasonable cost with current technologies: it would amount to compute the maximum of several hundred integers each time a cell is received from the user. However, it is not difficult to see that $\mathrm{Acr}(t)$ is constant on any interval that contains no value of the form $t_n + \tau_2$ or $t_n + \tau_3$. In other words, $\mathrm{Acr}(t)$ is determined by a sequence of values. It then becomes possible to use a scheduler handling future changes of $\mathrm{Acr}(t)$. This scheduler is updated when a new cell ER_n is received. Roughly, if s is the current time, ER_n will be taken into account at time $s + \tau_3$, while ER_{n-1} will not be taken into account after $s + \tau_2$.

The control conformance algorithm considered by ITU-T exploits this idea, with the further constraint that only a small amount of memory is allocated to the scheduler. This means that some information is lost. *Filtering is performed in such a way that the actual value of the allowed cell rate, as implemented by a variable called* `ACR`*, is greater or equal to its theoretical value* $\mathrm{Acr}(t)$ *defined above.*

3. Ideal Allowed Cell Rate

3.1. Declarative Specification

The declarative specification ($\mathcal{S}_\mathrm{d}$ in the introduction) of the ideal value or Acr is given by (1) and (2) under assumption (3). We are given a sequence of RM cells (ER_i) whose arrival time are respectively (t_i); the desired allowed cell rate at time t is defined by:

$$\mathrm{Acr}(t) = \max\{\mathrm{ER}_i \mid i \in I(t)\}, \tag{1}$$

where I is the interval defined by:

$$i \in I(t) \quad \text{iff} \quad (t - \tau_2 \, < \, t_i \, \le \, t - \tau_3) \ \vee \ (t_i \, \le \, t - \tau_2 \, < \, t_{i+1}) \ . \tag{2}$$

The second disjunct of (2) means $i = \mathrm{last}(t - \tau_2)$. The t_i are taken in increasing order:

$$t_1 < t_2 < \dots t_n < \dots \tag{3}$$

This specification was officially provided to us by ITU-T, but the reader can check that (2) is equivalent to:

$$i \in I(t) \quad \text{iff} \quad \mathrm{last}(t - \tau_2) \leq i \leq \mathrm{last}(t - \tau_3) \ .$$

The following equivalent characterization of $I(t)$ is easier to handle:

$$i \in I(t) \quad \text{iff} \quad t_i + \tau_3 \ \leq \ t \ < \ t_{i+1} + \tau_2 \tag{4}$$

The initial (inefficient) ABR conformance control algorithm was a direct computation of Acr according to (1).

3.2. Specification Using only Finite Knowledge

In practice, only finite prefixes of the sequence (t_i) are available. Then we have to take into account that, after the reception of the n first RM cells (ER_i), whose arrival time are respectively (t_i), t_{i+1} makes sense only for $i < n$. However, if we consider that $t_{n+1} = \infty$, (4) boils down to $n \in I(t)$ iff $t_n \ \leq \ t - \tau_3$. With this intuition in mind we introduce the n^{th} approximation of Acr, defined by

$$\mathrm{Approx}(n, t) = \max\{\mathrm{ER}_i \mid i \in I_a(n, t)\}, \tag{5}$$

where $I_a(n, t)$ is similar to $I(t)$:

$$i \in I_a(n, t) \quad \text{iff} \quad \begin{cases} \ i \in I(t) & \wedge \quad i < n \\ \vee & \\ \ t_i \ \leq \ t - \tau_3 & \wedge \quad i = n \ . \end{cases} \tag{6}$$

The number n we consider depends on time : if s represents the current time, we have $i \leq n(s)$ if and only if $t_i \leq s$. The following lemma, whose meaning is that it is enough to compute $\mathrm{Approx}(n(s), t)$, is easy to prove :

Lemma 1.
(i) The value of Approx at t becomes stable after $t - \tau_3$:

$$\forall s, t - \tau_3 \leq s \Rightarrow \quad \mathrm{Approx}(n(s), t) = \mathrm{Approx}(n(t - \tau_3), t) \ .$$

(ii) $\mathrm{Approx}(n(s), t)$ is an exact approximation of $\mathrm{Acr}(t)$ for $t - \tau_3 \leq s$:

$$\forall t, \quad \mathrm{Acr}(t) = \mathrm{Approx}(n(t - \tau_3), t) \ .$$

Unless otherwise stated s will remain implicit in the following : we will write just n instead of $n(s)$. In the same way, variables like `Efi` (see below) actually handled by the algorithm denote a value that also depends on s, but will be noted just `Efi`. We also assume without loss of generality that ER_0 is equal to the initial value of Acr; if the algorithm starts at s_0, this amounts to stating $t_0 = s_0 - \tau_3$ and for i and t_i such that $0 < i$ and $s_0 < t_i$:

$$t_0 = s_0 - \tau_3 < s_0 < t_1 < t_2 < \ldots t_n \ . \tag{7}$$

The will use the following explicit characterization of $I_a(n,t)$:

$$i \in I_a(n,t) \text{ iff } \begin{cases} \quad t_i + \tau_3 \ \leq \ t \ < \ t_{i+1} + \tau_2 & \wedge \quad i < n \\ \vee & \\ \quad t_i + \tau_3 \ \leq \ t & \wedge \quad i = n \ . \end{cases} \tag{8}$$

In particular, for $t_n + \tau_3 \ \leq \ t \leq t'$, we have $i \in I_a(n,t')$ implies $i \in I_a(n,t)$, hence the following lemma :

Lemma 2. $\mathrm{Approx}(n,t)$ is decreasing after $t_n + \tau_3$:

$$\forall t, t', t_n + \tau_3 \ \leq \ t \leq t' \Rightarrow \mathrm{Approx}(n,t') \leq \mathrm{Approx}(n,t) \ .$$

It is also easy to see that for any t greater than $t_n + \tau_2$, the only i of $I_a(n,t)$ is n, hence the following lemma :

Lemma 3.

$$\forall t, \quad t_n + \tau_2 \ \leq \ t \quad \Rightarrow \quad i \in I_a(n,t) \text{ iff } i = n, \tag{9}$$

$$\forall t, \quad t_n + \tau_2 \ \leq \ t \quad \Rightarrow \quad \mathrm{Approx}(n,t) = \mathrm{ER}_n. \tag{10}$$

3.3. Computing Approx in an Incremental Way

Initially (at time $s_0 = t_0 + \tau_3$), we have $n = 0$. The characterization (8) of I_a yields then

$$i \in I_a(0,t) \quad \text{iff} \quad s_0 \leq t \ \wedge \ i = 0 \ .$$

Then we get (as desired):

$$\forall t, \quad s_0 \leq t \ \Rightarrow \ \mathrm{Approx}(0,t) = \mathrm{ER}_0 \ . \tag{11}$$

We now consider the arrival of ER_n. Let us fix a given t.

- If $t_n \leq t - \tau_2$, lemma 3 yields $I_a(n,t) = \{n\}$ and $\mathrm{Approx}(n,t) = \mathrm{ER}_n$.

- If $t - \tau_2 < t_n \leq t - \tau_3$, $I_a(n,t)$ includes n as well as the numbers already in $I_a(n-1,t)$: this can be shown from (8) and is intuitively clear from figure 2. We get $I_a(n,t) = I_a(n-1,t) \cup \{n\}$ and $\mathrm{Approx}(n,t) = \max(\mathrm{Approx}(n-1,t), \mathrm{ER}_n)$
- In the same way, if $t - \tau_3 < t_{n+1}$, then ER_{n+1} is outside the relevant interval for t, which yields $I_a(n+1,t) = I_a(n,t)$ and $\mathrm{Approx}(n,t) = \mathrm{Approx}(n-1,t)$

Thus we get the following way of computing the value of $\mathrm{Approx}(n,t)$ from the value of $\mathrm{Approx}(n-1,t)$ (this is what we called $\mathcal{S}_\mathrm{c}$ in the introduction).

Lemma 4. The value of $\mathrm{Approx}(n,t)$ is given by the following table.

$t < t_n + \tau_3$	$t_n + \tau_3 \leq t < t_n + \tau_2$	$t_n + \tau_2 \leq t$
$\mathrm{Approx}(n-1,t)$	$\max(\mathrm{Approx}(n-1,t), \mathrm{ER}_n)$	ER_n

A simple but useful consequence of lemma 4 is :

Lemma 5.

$$\forall\, l, t, t_n, \mathrm{ER}_n, M,$$
$$\mathrm{Approx}(l,t) \leq M \;\wedge\; \mathrm{ER}_n \leq M \quad \Rightarrow \quad \mathrm{Approx}(l \frown t_n, t) \leq M. \qquad (12)$$

4. Proof of Algorithm B'

The enhanced algorithm B (hereafter called B') proposed by France Télécom at ITU-T computes an upper bound of Approx. More precisely, at the current instant s, the state $e(s)$ handled by B' defines a function Ub of $e(s)$ and t such that $\mathrm{Ub}(e(s),t)$ is greater than $\mathrm{Approx}(n(s),t)$ for any $t \geq s$ and is not defined for $t < s$.

Running the algorithm consists of changing the current state e to a new state e'. Such a step is called a *discrete transition.* Algorithm B' defines two discrete transitions: the first, called the external transition, is fired when receiving a new RM cell (this is called an external event in the sequel); the second, called the internal transition, is fired when the current time reaches the time for a scheduled event (this is called an internal event in the sequel).

4.1. Modelling and proof principles

Our treatment of time is inspired by timed automata of [4] and the synchrony hypothesis of synchronous languages [13]. Timed automata distinguish "continuous" transitions concerning time evolution (modeled by clocks under the control of the environment) and "discrete" transitions changing the state using no time.

Considering that discrete transitions are instantaneous deserves a special discussion. Basically, we assume, as in synchronous languages, that our system reacts more quickly than its environment. This assumption depends on the technology used in the real device and can be checked on it. It is then equivalent and simpler to consider that a discrete transition takes no time: if the property of interest is true on a model where discrete transitions are instantaneous, it remains true if the exact duration of discrete transitions are taken into account. Here, the internal transition is simply a multiple assignment; moreover, its crucial part reduces to a single assignment about `ACR`. The most complicated transition is the external one; but our discussion on Approx showed that the arrival of a new RM cell at time t_k has an effect only after $t_k+\tau_3$ (see lemma 4). Thus it is enough that running the external transition takes less than τ_3. It may happen that an internal event is scheduled at a time t_k. In that case, the internal event has to be handled first.

Concerning continuous transitions, we just need to assume the existence of a clock s. We model the progress of time by an assignment $s := s'$.

A run is a sequence of pairs $\langle s_i, e_i \rangle$ where s_i the i^{th} value of the clock, and e_i is the i^{th} value of the state, with the condition that for all i, we have $s_i \leq s_{i+1}$ and $e_i = e_{i+1}$ (continuous transition) or $s_i = s_{i+1}$ and e_{i+1} is obtained from e_i by applying an internal or an external transition. In the case of a continuous transition, we also constrain s_{i+1} in a way such that no event arose between s_i and s_{i+1} (we assume that the scheduler is reliable).

4.2. Notations

States and discrete transitions are modelled using standard calculus of weakest preconditions [11] with notations taken from B [2]: discrete transitions are modeled by program assignments or "generalized substitutions" in the terminology of B, as discussed in the introduction. We use multiple assignments like $x_1, x_2 := E_1, E_2$. Applying $[x_1, x_2 := E_1, E_2]$ on a predicate P yields P where x_1 and x_2 are simultaneously replaced with E_1 and E_2. Another notation for the same multiple assignment is $x_1 := E_1 \ || \ x_2 := E_2$.

4.3. Components of the State

The state e is made of 5 variables :

- `ACR`, the current Allowed Cell Rate;
- `Efi`, the next Allowed Cell Rate if nothing new happens;
- `tfi`, the time at which the Allowed Cell Rate will become equal to `Efi` if nothing new happens;
- `Ela`, containing the value of the last known cell (ER_n);
- `tla`, the time at which the Allowed Cell Rate will become equal to `Ela` if nothing new happens.

As an optimization trick, there is a six$^{\text{th}}$ variable `Emx` whose value is just the maximum of `Efi` and `Ela`.

When running a continuous transition from $\langle s, e\rangle$ to $\langle s', e\rangle$, we cannot have $s < u < s'$ where u is `tfi` or one of the t_n. But continuous transitions can follow discrete transitions and vice-versa, hence we must allow $s = u$ and $u = s'$. It turns out that the desired property is not satisfied when $s' = \mathtt{tfi}$, that is just before running the internal transition: an internal event is precisely scheduled in order to update the allowed cell rate at this time. In order to take this into account, our model involves an additional boolean variable `utd` (for "up to date"): `utd` is true most of the time; it becomes false only after a continuous transition such that $s' = \mathtt{tfi}$. Note that, `utd` is not part of the algorithm, it is just an artefact of the model which is needed in the invariant.

4.4. Transitions

The algorithm reacts either when receiving a new ER_n, i.e. when the current time is equal to t_n, or when the current time reaches `tfi`. Each transition changes the current state; an internal event is scheduled if and only if `tfi` is greater than the current time.

4.5. Invariant

We want to ensure that algorithm B' provides an `ACR` which cannot be less than the ideal value $\mathrm{Acr}(s)$. To this effect we prove that the following property is invariant, where s denotes the current time and n denotes the number of the last RM cell received at time s :

$$\mathtt{utd} = \mathrm{true} \Rightarrow \mathrm{Approx}(n, s) \leq \mathtt{ACR}\ . \qquad (\mathrm{I}_{\mathrm{main}})$$

$\mathrm{I}_{\mathrm{main}}$ is itself a consequence of the following conjunction.

$$\texttt{utd} = \text{false} \;\Rightarrow\; s = \texttt{tfi} \tag{$\mathrm{I}_{\mathrm{utd}}$}$$

$$\texttt{Emx} = \max(\texttt{Efi}, \texttt{Ela}) \tag{$\mathrm{I}_{\mathrm{max}}$}$$

$$\texttt{Ela} = \mathrm{ER}_n \tag{$\mathrm{I}_{\mathrm{Ela}}$}$$

$$\texttt{tfi} \leq \texttt{tla} \leq t_n + \tau_2 \tag{$\mathrm{I}_{\mathrm{fil}}$}$$

$$\begin{array}{l}(\texttt{tfi} = s \Rightarrow \texttt{utd} = \text{true}) \Rightarrow \\ \texttt{tfi} \leq s \;\Rightarrow\; \forall t,\; s \leq t \;\Rightarrow\; \mathrm{Approx}(n,t) \leq \texttt{ACR}\end{array} \tag{$\mathrm{I}_{\mathrm{tfs}}$}$$

$$\texttt{ACR} < \texttt{Efi} \;\Rightarrow\; \texttt{tfi} \leq t_n + \tau_3 \tag{$\mathrm{I}_{\mathrm{Et1}}$}$$

$$\texttt{Efi} < \texttt{Ela} \;\Rightarrow\; \texttt{tla} \leq t_n + \tau_3 \tag{$\mathrm{I}_{\mathrm{Et2}}$}$$

$$\texttt{tfi} = \texttt{tla} \;\Rightarrow\; \texttt{Efi} = \texttt{Ela} \tag{$\mathrm{I}_{\mathrm{ttE}}$}$$

$$\forall t \quad s \leq t < \texttt{tfi} \;\Rightarrow\; \mathrm{Approx}(n,t) \leq \texttt{ACR} \tag{$\mathrm{I}_{\mathrm{Ub1}}$}$$

$$\forall t \quad \texttt{tfi} \leq t < \texttt{tla} \;\Rightarrow\; \mathrm{Approx}(n,t) \leq \texttt{Efi} \tag{$\mathrm{I}_{\mathrm{Ub2}}$}$$

$$\forall t \quad \texttt{tla} \leq t \;\Rightarrow\; \mathrm{Approx}(n,t) \leq \texttt{Ela}\;. \tag{$\mathrm{I}_{\mathrm{Ub3}}$}$$

We define

$$\mathrm{Inv} = \mathrm{I}_{\mathrm{utd}} \wedge \mathrm{I}_{\mathrm{max}} \wedge \mathrm{I}_{\mathrm{Ela}} \wedge \mathrm{I}_{\mathrm{fil}} \wedge \mathrm{I}_{\mathrm{tfs}} \wedge \mathrm{I}_{\mathrm{Et1}} \wedge \mathrm{I}_{\mathrm{Et2}} \wedge \mathrm{I}_{\mathrm{ttE}} \wedge \mathrm{I}_{\mathrm{Ub1}} \wedge \mathrm{I}_{\mathrm{Ub2}} \wedge \mathrm{I}_{\mathrm{Ub3}}.$$

Invariants $\mathrm{I}_{\mathrm{Ub1}}$, $\mathrm{I}_{\mathrm{Ub2}}$ and $\mathrm{I}_{\mathrm{Ub3}}$ mean that $\mathrm{Approx}(n,t) \leq \mathrm{Ub}(e,t)$ for $t \geq s$, where the function $\mathrm{Ub}(e,t)$ is defined by: $\mathrm{Ub}(e,t) = \texttt{ACR}$ for $s \leq t < \texttt{tfi}$, $\mathrm{Ub}(e,t) = \texttt{Efi}$ for $\texttt{tfi} \leq t < \texttt{tla}$, $\mathrm{Ub}(e,t) = \texttt{Ela}$ for $\texttt{tla} \leq t$. In the sequel we use the following consequence of $\mathrm{I}_{\mathrm{max}}$, $\mathrm{I}_{\mathrm{Ub2}}$ and $\mathrm{I}_{\mathrm{Ub3}}$:

$$\forall t \quad \texttt{tfi} \leq t \;\Rightarrow\; \mathrm{Approx}(n,t) \leq \texttt{Emx}\;. \tag{$\mathrm{I}_{\mathrm{Apx}}$}$$

Lemma 6. Inv implies $\mathrm{I}_{\mathrm{main}}$.

Proof. We have either $\texttt{tfi} \leq s$ or $s < \texttt{tfi}$. Apply respectively $\mathrm{I}_{\mathrm{tfs}}$ and $\mathrm{I}_{\mathrm{Ub1}}$ with $t = s$.

4.6. Initial State

Initially we have $n = 0$ and we want to show that Inv is true in the initial state defined by:

$\texttt{tfi} = \texttt{tla} = s_0, \texttt{ACR} = \texttt{Emx} = \texttt{Efi} = \texttt{Ela} = \mathrm{ER}_0$ (initial value of Acr), and $\texttt{utd} = \text{true}$ Formally, we consider the substitution $\mathsf{S_0}$:

$$\mathsf{S_0} \stackrel{\text{df}}{=} \left\{ \begin{array}{l} := \end{array} \begin{array}{llllllll} n, & \texttt{ACR}, & \texttt{Emx}, & \texttt{Efi}, & \texttt{Ela}, & \texttt{tfi}, & \texttt{tla}, & \texttt{utd} \\ 0, & \text{ER}_0, & \text{ER}_0, & \text{ER}_0, & \text{ER}_0, & s_0, & s_0, & \text{true} \end{array} \right.$$

and we show $[\mathsf{S_0}]\text{Inv}$, that is, the formula Inv where n, $\texttt{ACR}$, $\texttt{Emx}$, $\texttt{Efi}$, $\texttt{Ela}$, $\texttt{tfi}$, $\texttt{tla}$ are respectively replaced with 0, ER_0, ER_0, ER_0, ER_0, s_0, s_0. The proof is very easy.

4.7. Continuous Transitions

Let s be the current time. We consider a transition from $\langle s, e\rangle$ to $\langle s', e\rangle$ only if $s \le s'$ and there is no event between s and s'. If s' is equal to $\texttt{tfi}$, $\texttt{utd}$ is set to false (when real time reaches $\texttt{tfi}$, $\mathsf{S_i}$ must run). In the same way, if $s = t_{n+1}$, time cannot progress (an external transition must run, $n := n+1$).

Formally, we consider the following guard:

$$\begin{array}{l} s \le s' \ \wedge\ (\texttt{tfi} \le s \vee s' \le \texttt{tfi}) \ \ \wedge\ (\forall i,\ t_i \le s \vee s' \le t_i) \wedge \\ (\texttt{utd} = \text{false} \Rightarrow s' = \texttt{tfi}) \ \wedge\ s = t_{n+1} \ \Rightarrow\ s' = t_{n+1}. \end{array} \qquad (G_c)$$

The transition is modeled by the substitution:

$$\mathsf{S_C} \stackrel{\text{df}}{=} s, \texttt{utd} := s', \texttt{newutd}$$

with $\texttt{newutd} = \text{true}$ if $s' \ne \texttt{tfi}$ and $\texttt{newutd} = \text{false}$ if $s' = \texttt{tfi}$. We prove: $\text{Inv} \wedge (G_c) \Rightarrow [\mathsf{S_C}]\text{Inv}$.

Remark 1. We consider proof obligations of the form $[\mathsf{S}]\text{Inv}$, where $[\mathsf{S}]$ is a substitution. It is decomposed into $[\mathsf{S}]\text{I}_{\text{utd}}$, $[\mathsf{S}]\text{I}_{\text{max}}$, $[\mathsf{S}]\text{I}_{\text{Ela}}$, $[\mathsf{S}]\text{I}_{\text{fil}}$, $[\mathsf{S}]\text{I}_{\text{tfs}}$, $[\mathsf{S}]\text{I}_{\text{Et1}}$, $[\mathsf{S}]\text{I}_{\text{Et2}}$, $[\mathsf{S}]\text{I}_{\text{ttE}}$, $[\mathsf{S}]\text{I}_{\text{Ub1}}$, $[\mathsf{S}]\text{I}_{\text{Ub2}}$ and $[\mathsf{S}]\text{I}_{\text{Ub3}}$. Some of them are immediate, for instance $\texttt{Efi} < \text{ER}_k \Rightarrow t_k + \tau_3 \le t_k + \tau_3$ or $\texttt{Ela} < \texttt{Ela} \Rightarrow \texttt{tla} \le t_n + \tau_3$. They are skipped in order to save space.

Proof.

- $[\mathsf{S_C}]\text{I}_{\text{tfs}}$, that is $(\texttt{tfi} = s' \Rightarrow \texttt{newutd} = \text{true}) \Rightarrow \texttt{tfi} \le s' \Rightarrow$ $\forall t,\ s' \le t \ \Rightarrow\ \text{Approx}(n, t) \le \texttt{ACR}$: first remark that if $\texttt{tfi} = s'$, we get $\texttt{newutd} = \text{true}$ which is $s' \ne \texttt{tfi}$, hence a contradiction; this implies $\texttt{utd} = \text{true}$ (because otherwise, $\texttt{utd} = \text{false}$ and then $\texttt{tfi} = s'$ by (G_c)) and also that $\texttt{tfi} \le s'$ reduces to $\texttt{tfi} < s'$; using (G_c) we then get $\texttt{tfi} \le s$; for t such that $s' \le t$, we have $s \le t$ by (G_c); then we get $\text{Approx}(n, t) \le \texttt{ACR}$ thanks to I_{tfs}.

- $[\mathsf{S_C}]\text{I}_{\text{Ub1}}$, that is $\forall t\ s' \le t < \texttt{tfi} \ \Rightarrow\ \text{Approx}(n, t) \le \texttt{ACR}$: for t such that $s' \le t < \texttt{tfi}$ we have $s \le s' \le t < \texttt{tfi}$ by (G_c), then $\text{Approx}(n, t) \le \texttt{ACR}$ by I_{Ub1}.

4.8. Internal Transition

Let s be the current time with $s = \texttt{tfi}$. Let $\mathsf{S_i}$ be the substitution

$$\mathsf{S_i} \stackrel{\mathrm{df}}{=} \texttt{ACR}, \texttt{tfi}, \texttt{Efi}, \texttt{Emx}, \texttt{utd} := \texttt{Efi}, \texttt{tla}, \texttt{Ela}, \texttt{Ela}, \mathrm{true}.$$

We show: $\mathrm{Inv} \wedge s = \texttt{tfi} \Rightarrow [\mathsf{S_i}]\mathrm{Inv}$.

Remark 2. $[\mathsf{S_i}]\mathrm{I_{tfs}}$ has the form $(X \Rightarrow \mathrm{true} = \mathrm{true}) \Rightarrow Y$: this reduces to Y. The same remark applies on the external transition, because the considered substitution includes $\texttt{utd} := \mathrm{true}$.

Proof.

- $[\mathsf{S_i}]\mathrm{I_{Ela}}$, that is $\texttt{Ela} = \mathrm{ER}_n$: from $\mathrm{I_{Ela}}$.
- $[\mathsf{S_i}]\mathrm{I_{fil}}$, that is $\texttt{tla} \le \texttt{tla} \le t_n + \tau_2$: we have $\texttt{tla} \le t_n + \tau_2$ by $\mathrm{I_{fil}}$.
- $[\mathsf{S_i}]\mathrm{I_{tfs}}$, that is $\texttt{tla} \le \texttt{tfi} \Rightarrow \forall t,\ \texttt{tfi} \le t \Rightarrow \mathrm{Approx}(n,t) \le \texttt{Efi}$: $\texttt{tla} \le \texttt{tfi}$ and $\mathrm{I_{fil}}$ yields $\texttt{tfi} = \texttt{tla}$, then $\texttt{Efi} = \texttt{Ela}$ by $\mathrm{I_{ttE}}$; we then have to show that for t such that $\texttt{tla} \le t$, we have $\mathrm{Approx}(n,t) \le \texttt{Ela}$: we just apply $\mathrm{I_{Ub3}}$.
- $[\mathsf{S_i}]\mathrm{I_{Et1}}$, that is $\texttt{Efi} < \texttt{Ela} \Rightarrow \texttt{tla} \le t_n + \tau_3$: it is just $\mathrm{I_{Et2}}$.
- $[\mathsf{S_i}]\mathrm{I_{Ub1}}$, that is $\forall t\ \texttt{tfi} \le t < \texttt{tla} \Rightarrow \mathrm{Approx}(n,t) \le \texttt{Efi}$: it is just $\mathrm{I_{Ub2}}$.
- $[\mathsf{S_i}]\mathrm{I_{Ub3}}$, that is $\forall t\ \texttt{tla} \le t \Rightarrow \mathrm{Approx}(n,t) \le \texttt{Ela}$: for t such that $\texttt{tla} \le t$, we have $s \le \texttt{tla} \le t$ by (G_i); we can then apply $\mathrm{I_{Ub3}}$, which yields $\mathrm{Approx}(n,t) \le \texttt{Ela}$.

4.9. External Transition

Let $s = t_k$ be the current time, and let $n = k-1$ be the number of the last RM cell received before s. We consider a transition taking the k^{th} RM cell only if $\texttt{utd}$ is true. Formally, we have the following guard:

$$\texttt{utd} = \mathrm{true} \qquad (G_e)$$

The complete external substitution is divided into eight cases, depending on comparisons between ER_k, $\texttt{ACR}$, $\texttt{Efi}$, $\texttt{Ela}$ on the one hand, and between $t_k + \tau_2$ or $t_k + \tau_3$ and $\texttt{tfi}$ and $\texttt{tla}$ on the other hand. In each case, the considered substitution includes:

$$\mathsf{T_e} \stackrel{\mathrm{df}}{=} n, \texttt{utd} := k, \mathrm{true}.$$

For instance, consider the eight$^{\mathrm{th}}$ case:

$$\begin{array}{l}\textbf{if } t_k < \texttt{tfi} \textbf{ then else if } \texttt{ACR} \le \mathrm{ER}_k \textbf{ then else} \\ \quad \texttt{Efi} := \mathrm{ER}_k \parallel \texttt{Ela} := \mathrm{ER}_k \parallel \texttt{Emx} := \mathrm{ER}_k \\ \quad\quad \parallel \texttt{tfi} := t_k + \tau_2 \parallel \texttt{tla} := t_k + \tau_2\end{array}$$

This transitions corresponds to the case where the scheduler is empty (`tfi` is not greater than the current time and `utd` is true) ansd where a cell smaller than the current ACR is received: ER_k is then scheduled at $t_k+\tau_2$. Let S_8 be the substitution

$$\mathsf{S}_8 \stackrel{\mathrm{df}}{=} \mathsf{T_e} \parallel \texttt{Efi}, \texttt{Ela}, \texttt{Emx}, \texttt{tfi}, \texttt{tla} := \mathrm{ER}_k, \mathrm{ER}_k, \mathrm{ER}_k, t_k+\tau_2, t_k+\tau_2.$$

This transition is correct if :

$$\mathrm{Inv} \wedge (G_e) \wedge \texttt{tfi} \le t_k \wedge \mathrm{ER}_k < \texttt{ACR} \Rightarrow [\mathsf{S}_8]\mathrm{Inv}.$$

The complete pseudo-code and the proofs are given is appendixes A and B.

4.10. Main theorem

Our main result is an easy consequence of previous lemmas.

THEOREM 1. *At any time s we have* $\mathrm{Acr}(s) \le \texttt{ACR}$.

Proof. Using lemma 1 we know that $\mathrm{Acr}(s) = \mathrm{Approx}(n, s)$. As Inv is actually an invariant, lemma 6 yields $\mathrm{Approx}(n, s) \le \texttt{ACR}$ when $\texttt{utd} = \mathrm{true}$. However if $\texttt{utd} = \mathrm{false}$, we have $s = \texttt{tfi}$ by $\mathrm{I_{utd}}$ end the internal transition is immediately fired, hence the result.

4.11. Further Properties

It is easy to see that at any time, the guard of at least one transition is satisfied. Moreover, time progress is never blocked: continuous transitions with a stricly positive duration are allowed, excepted when the current time is equal to t_k, or to `tfi` with `utd` equal to false. In the latter case, an internal transition can be fired, making `utd` true. When `utd` is true, the current time is either equal to t_k or not. In the first case, an external transition can be fired, making the current time stricly less than t_k because n is incremented; `utd` remains true, hence continuous transitions with a stricly positive duration are allowed again.

The model given above also allows stuttering transitions, *i.e.* a continuous transitions with a null duration ($s' = s$). A fairness condition is then needed in order to ensure actual progress. An alternative is to replace $s \le s'$ with $s < s'$ in (G_c).

An interesting consequence of progress and of Inv is that the standardized algorithm is actually much better than the trivial algorithm which just returns the maximum of the ER_k. Indeed, if no RM cell is received before `tla`, the value of `ACR` at `tla` will be `Ela`: by $\mathrm{I}_{\mathrm{fil}}$ we know that `tla` cannot be later than $s + \tau_2$ and by $\mathrm{I}_{\mathrm{Ela}}$, `Ela` is equal to the last received ER_n. Moreover, in this situation, ER_n is precisely equal to $\mathrm{Acr}(t)$ for t greater than $s + \tau_2$.

5. Discussion and Related Work

For engineers working in the context of standardization, theorem 1 is much more convincing than the similar theorem involving Approx instead of Acr. However it is clear for us that the computational characterization of Approx (lemma 4) is much more suited for reasoning about algorithm B'. In a first attempt, we tried to prove directly the invariant Inv using (1) and (2). This resulted in shallow areas and even holes in the manual proof.

We also submitted the problem of the correctness of B' to other research teams, in order to assess other approaches. It is too early (and beyond the scope of this paper) to compare the results of these works, we just give some hints. Model checking using classical and temporal automata is experimented in the framework of FORMA[1], a project founded by the French government which aims at experimenting various formal methods on industrial case studies. In the two first attempts, the property to be checked corresponded to theorem 1, but modeling Acr contributed to an explosion of the number of states. Moreover the tools used—UPPAAL [7] and MEC [5]—allowed only fixed numeric values for τ_2, τ_3 and ER_i. Checking the algorithm could be carried through for small values. Later on, good results within two different frameworks have been obtained by L. Fribourg [12] and B. Bérard [8], with specifications based on Approx instead of Acr. In one framework, they used the parameterized temporized automata of Hytech [14], and in the other an automated proof search procedure due to Revesz [21] was extended to timed automata. In both cases τ_2, τ_3, etc. were symbolic parameters and the desired property could be checked without the help of Inv. In our case, Inv has been incrementally constructed while attempting to prove $\mathrm{I}_{\mathrm{tfs}}$ and $\mathrm{I}_{\mathrm{Ub1}}$, following the steps given in appendix B. Note that such calculations are boring and error prone: this is why we felt that the proof should be checked with a proof assistant. Indeed, our experiment with Coq [16] showed that one of the proofs of appendix B was wrong

[1] `http://www-verimag.imag.fr/FORMA`

(but could be repaired, fortunately !). A detailed comparison between the approaches mentioned above is done in [9].

A similar algorithm for ABR conformance is studied in [22]. This algorithm handles a non bounded scheduling list. Its proof, which involves intricate case analysis and inductive reasoning on lists, has been supported by the PVS [18] theorem prover. The main design decision was to write a first order specification in order to exploit the automated features of PVS. However the proof still requires around 80 lemmas. The authors hope that many steps could be simplified with the use of more automated tools. Another direction, more akin to the spirit of the work presented in this paper, would be to carry out a proof using higher-order specification and reasoning techniques, in order to derive a proof that is more synthetic and therefore easier to grasp.

Finally, let us say a word on two attempts using B. Three years ago, we (with G. Blorec) tried to use this method on this case study. At first sight B should be well suited, because of our systematic use of substitution calculus. But we failed to handle time and the very notion of scheduler in a nice way; our specification was heavy and many proof obligations could not be discharged. Recently, Abrial worked on this problem using an event oriented variant of B and he succeeded to reconstruct an algorithm different from the one standardized in I.371, but where the design decisions are much clearer [3].

Our current feeling is that specialized procedures or methods can discharge boring and painful parts in the verification process, but are really successful only on “predigested” specifications like $\mathcal{S}_{\mathrm{c}}$, in contrast with $\mathcal{S}_{\mathrm{d}}$. On the other side, general purpose frameworks and tools like type theory and Coq are helpful on the whole process but still require much more interaction from the user on the parts automatically handled by specialized methods. Work is in progress for integrating timed automa generalized with arbitrary types (including e.g. function) and automated techniques within the same tool, in the framework of a research project partially founded by the french government (Calife).

Acknowledgement

The problem has been submitted by Annie Gravey and Christophe Rabadan, who is the main author of the algorithm studied in this document. This work has benefited of interesting discussions with Francis Klay. Many improvements are also due to the comments of anonymous referees.

Appendix

A. Pseudo-code for Algorithm B'

When real time reaches t_k :

$$
\begin{array}{l}
\textbf{if } t_k < \texttt{tfi} \textbf{ then} \\
\quad \textbf{if } \texttt{Emx} \leq \mathrm{ER}_k \textbf{ then} \\
\quad\quad \textbf{if } \texttt{tfi} < t_k + \tau_3 \textbf{ then} \\
\quad\quad\quad \textbf{if } t_k + \tau_3 < \texttt{tla} \vee \texttt{tfi} = \texttt{tla} \textbf{ then} \\
\quad\quad\quad\quad \texttt{Emx} := \mathrm{ER}_k \;||\; \texttt{Ela} := \mathrm{ER}_k \;||\; \texttt{tla} := t_k + \tau_3 \\
\quad\quad\quad \textbf{else} \\
\quad\quad\quad\quad \texttt{Emx} := \mathrm{ER}_k \;||\; \texttt{Ela} := \mathrm{ER}_k \\
\quad\quad \textbf{else} \\
\quad\quad\quad \textbf{if } \texttt{ACR} \leq \mathrm{ER}_k \textbf{ then} \\
\quad\quad\quad\quad \texttt{Emx} := \mathrm{ER}_k \;||\; \texttt{Efi} := \mathrm{ER}_k \;||\; \texttt{Ela} := \mathrm{ER}_k \\
\quad\quad\quad\quad\quad\quad\quad\quad \;||\; \texttt{tfi} := t_k + \tau_3 \;||\; \texttt{tla} := t_k + \tau_3 \\
\quad\quad\quad \textbf{else} \\
\quad\quad\quad\quad \texttt{Emx} := \mathrm{ER}_k \;||\; \texttt{Efi} := \mathrm{ER}_k \;||\; \texttt{Ela} := \mathrm{ER}_k \;||\; \texttt{tla} := \texttt{tfi} \\
\quad \textbf{else} \\
\quad\quad \textbf{if } \mathrm{ER}_k < \texttt{Ela} \textbf{ then} \\
\quad\quad\quad \texttt{Efi} := \texttt{Emx} \;||\; \texttt{Ela} := \mathrm{ER}_k \;||\; \texttt{tla} := t_k + \tau_2 \\
\quad\quad \textbf{else} \\
\quad\quad\quad \texttt{Efi} := \texttt{Emx} \;||\; \texttt{Ela} := \mathrm{ER}_k \\
\textbf{else} \\
\quad \textbf{if } \texttt{ACR} \leq \mathrm{ER}_k \textbf{ then} \\
\quad\quad \texttt{Efi} := \mathrm{ER}_k \;||\; \texttt{Ela} := \mathrm{ER}_k \;||\; \texttt{Emx} := \mathrm{ER}_k \\
\quad\quad\quad\quad\quad\quad \;||\; \texttt{tfi} := t_k + \tau_3 \;||\; \texttt{tla} := t_k + \tau_3 \\
\quad \textbf{else} \\
\quad\quad \texttt{Efi} := \mathrm{ER}_k \;||\; \texttt{Ela} := \mathrm{ER}_k \;||\; \texttt{Emx} := \mathrm{ER}_k \\
\quad\quad\quad\quad\quad\quad \;||\; \texttt{tfi} := t_k + \tau_2 \;||\; \texttt{tla} := t_k + \tau_2
\end{array}
$$

When real time reaches `tfi`:

$\texttt{ACR} := \texttt{Efi} \;||\; \texttt{tfi} := \texttt{tla} \;||\; \texttt{Efi} := \texttt{Ela} \;||\; \texttt{Emx} := \texttt{Ela}$

If $\texttt{tfi} = t_k$, we run the algorithm for `tfi`, then the algorithm for t_k.

B. Proof of Algorithm B'

Recall that (G_e) is $\texttt{utd} = \text{true}$

Case 1

if $t_k < \texttt{tfi}$ **then if** $\texttt{Emx} \le \text{ER}_k$ **then if** $\texttt{tfi} < t_k+\tau_3$
then if $t_k+\tau_3 < \texttt{tla} \lor \texttt{tfi} = \texttt{tla}$
then $\texttt{Emx} := \text{ER}_k \;||\; \texttt{Ela} := \text{ER}_k \;||\; \texttt{tla} := t_k+\tau_3$

Let $\mathsf{S_1}$ be the substitution $\mathsf{S_1} \stackrel{\text{df}}{=} \mathsf{T_e} \;||\; \texttt{Emx}, \texttt{Ela}, \texttt{tla} := \text{ER}_k, \text{ER}_k, t_k+\tau_3$. This transition is correct if :

$$\text{Inv} \quad \land \quad (G_e) \quad \land$$
$$t_k < \texttt{tfi} \quad \land \qquad (G_{11})$$
$$\texttt{Emx} \le \text{ER}_k \quad \land \qquad (G_{12})$$
$$\texttt{tfi} < t_k+\tau_3 \quad \land \qquad (G_{13})$$
$$(t_k+\tau_3 < \texttt{tla} \lor \texttt{tfi} = \texttt{tla}) \quad \Rightarrow \qquad (G_{14})$$
$$[\mathsf{S_1}]\text{Inv}.$$

Proof. We assume Inv, (G_e), (G_{11}), (G_{12}), (G_{13}), (G_{14}), and we prove $[\mathsf{S_1}]\text{Inv}$.

- $[\mathsf{S_1}]\text{I}_{\text{max}}$, that is $\text{ER}_k = \max(\texttt{Efi}, \text{ER}_k)$: by I_{max} and (G_{12}).
- $[\mathsf{S_1}]\text{I}_{\text{fil}}$, that is $\texttt{tfi} \le t_k+\tau_3 \le t_k+\tau_2$: trivial from (G_{13}).
- $[\mathsf{S_1}]\text{I}_{\text{tfs}}$, that is $\texttt{tfi} \le t_k \;\Rightarrow\; \forall t,\; t_k \le t \;\Rightarrow\; \text{Approx}(k,t) \le \texttt{ACR}$: absurd hypothesis, given (G_{11}).
- $[\mathsf{S_1}]\text{I}_{\text{Et1}}$, that is $\texttt{ACR} < \texttt{Efi} \Rightarrow \texttt{tfi} \le t_k+\tau_3$: the conclusion comes from (G_{13}).
- $[\mathsf{S_1}]\text{I}_{\text{ttE}}$, that is $\texttt{tfi} = t_k+\tau_3 \Rightarrow \texttt{Efi} = \text{ER}_k$: absurd hypothesis, given (G_{13}).
- $[\mathsf{S_1}]\text{I}_{\text{Ub1}}$, that is $\forall t,\; t_k \le t < \texttt{tfi} \Rightarrow \text{Approx}(k,t) \le \texttt{ACR}$:
  for t such that $t_k \le t < \texttt{tfi}$, we get $t < t_k+\tau_3$ by (G_{13}), then lemma 4 yields $\text{Approx}(k,t) = \text{Approx}(k-1,t)$; we can then apply I_{Ub1}, and finally we get $\text{Approx}(k,t) = \text{Approx}(k-1,t) \le \texttt{ACR}$.
- $[\mathsf{S_1}]\text{I}_{\text{Ub2}}$, that is $\forall t,\; \texttt{tfi} \le t < t_k+\tau_3 \Rightarrow \text{Approx}(k,t) \le \texttt{Efi}$:
  first remark that $\text{Approx}(k,t) = \text{Approx}(k-1,t)$ by lemma 4; (G_{14}) gives either $t_k+\tau_3 < \texttt{tla}$, or $\texttt{tfi} = \texttt{tla}$;

- in the former case, $\mathrm{Approx}(k-1,t) \le \mathtt{Efi}$ by $\mathrm{I}_{\mathrm{Ub2}}$, hence the result;
- in the latter case, $\mathrm{I}_{\mathrm{ttE}}$ yields $\mathtt{Efi} = \mathtt{Ela}$, and $\mathtt{tla} = \mathtt{tfi} \le t$ yields $\mathrm{Approx}(k-1,t) \le \mathtt{Ela}$ by $\mathrm{I}_{\mathrm{Ub3}}$, hence the result.

– $[\mathsf{S}_1]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ t_k+\tau_3 \le t \Rightarrow \mathrm{Approx}(k,t) \le \mathrm{ER}_k$:
we show $\forall t,\ t_k+\tau_3 \le t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$:
for t such that $t_k+\tau_3 \le t$, we have $\mathtt{tfi} \le t$ by (G_{13}), then $\mathrm{Approx}(k-1,t) \le \mathtt{Emx} \le \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{Apx}}$ and (G_{12}); lemma 4 yields either $\mathrm{Approx}(k,t) = \max(\mathrm{Approx}(k–1,t), \mathrm{ER}_k)$ or $\mathrm{Approx}(k,t) = \mathrm{ER}_k$; in both cases, we see that $\mathrm{Approx}(k,t) = \mathrm{ER}_k$.

CASE 2

if $t_k < \mathtt{tfi}$ **then if** $\mathtt{Emx} \le \mathrm{ER}_k$ **then if** $\mathtt{tfi} < t_k+\tau_3$ **then**
  if $t_k+\tau_3 < \mathtt{tla} \ \vee\ \mathtt{tfi} = \mathtt{tla}$ **then**
  else $\mathtt{Emx} := \mathrm{ER}_k \ \|\ \mathtt{Ela} := \mathrm{ER}_k$

nLet S_2 be the substitution $\mathsf{S}_2 \ \stackrel{\mathrm{df}}{=}\ \mathsf{T_e} \ \|\ \mathtt{Emx},\mathtt{Ela} := \mathrm{ER}_k, \mathrm{ER}_k$. This transition is correct if :

$$
\begin{aligned}
\mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge & \\
t_k < \mathtt{tfi} \quad \wedge & \qquad (G_{11})\\
\mathtt{Emx} \le \mathrm{ER}_k \quad \wedge & \qquad (G_{12})\\
\mathtt{tfi} < t_k+\tau_3 \quad \wedge & \qquad (G_{13})\\
\mathtt{tla} \le t_k+\tau_3 \quad \wedge & \qquad (G_{24})\\
\mathtt{tfi} \ne \mathtt{tla} \quad \Rightarrow & \qquad (G_{25})\\
[\mathsf{S}_2]\mathrm{Inv}. &
\end{aligned}
$$

Proof. $[\mathsf{S}_2]\mathrm{I}_{\max}$, $[\mathsf{S}_2]\mathrm{I}_{\mathrm{tfs}}$ and $[\mathsf{S}_2]\mathrm{I}_{\mathrm{Et1}}$, are proved as in case 1.

– $[\mathsf{S}_2]\mathrm{I}_{\mathrm{fil}}$, that is $\mathtt{tfi} \le \mathtt{tla} \le t_k+\tau_2$: trivial from $\mathrm{I}_{\mathrm{fil}}$ (G_{24}).

– $[\mathsf{S}_2]\mathrm{I}_{\mathrm{Et2}}$, that is $\mathtt{Efi} < \mathrm{ER}_k \Rightarrow \mathtt{tla} \le t_k+\tau_3$: the conclusion is (G_{24}).

– $[\mathsf{S}_2]\mathrm{I}_{\mathrm{ttE}}$, that is $\mathtt{tfi} = \mathtt{tla} \Rightarrow \mathtt{Efi} = \mathrm{ER}_k$: absurd hypothesis, given (G_{25}).

– $[\mathsf{S}_2]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\ t_k \le t < \mathtt{tfi} \Rightarrow \mathrm{Approx}(k,t) \le \mathtt{ACR}$: using (G_{13}), lemma 4 and $\mathrm{I}_{\mathrm{Ub1}}$, we have $\mathrm{Approx}(k,t) = \mathrm{Approx}(k–1,t) \le \mathtt{ACR}$.

- $[\mathsf{S}_2]\mathrm{I}_{\mathrm{Ub2}}$, that is $\forall t,\ \mathtt{tfi} \leq t < \mathtt{tla} \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{Efi}$: $\mathrm{Approx}(k,t) = \mathrm{Approx}(k-1,t)$ by lemma 4 and (G_{24}); $\mathrm{Approx}(k-1,t) \leq \mathtt{Efi}$ by $\mathrm{I}_{\mathrm{Ub2}}$, hence the result.

- $[\mathsf{S}_2]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ \mathtt{tla} \leq t \Rightarrow \mathrm{Approx}(k,t) \leq \mathrm{ER}_k$: we have $\mathtt{tfi} \leq t$ by $\mathrm{I}_{\mathrm{fil}}$, then $\mathrm{Approx}(k\text{–}1,t) \leq \mathtt{Emx} \leq \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{Apx}}$ and (G_{12}); taking $M = \mathrm{ER}_k$ in lemma 5 gives $\mathrm{Approx}(k,t) \leq \mathrm{ER}_k$.

Case 3

$$
\begin{array}{l}
\textbf{if } t_k < \mathtt{tfi} \textbf{ then if } \mathtt{Emx} \leq \mathrm{ER}_k \textbf{ then if } \mathtt{tfi} < t_k+\tau_3 \textbf{ then} \\
\quad \textbf{else if } \mathtt{ACR} \leq \mathrm{ER}_k \textbf{ then} \\
\qquad \mathtt{Emx} := \mathrm{ER}_k \mid\mid \mathtt{Efi} := \mathrm{ER}_k \mid\mid \mathtt{Ela} := \mathrm{ER}_k \\
\qquad\qquad\qquad \mid\mid \mathtt{tfi} := t_k+\tau_3 \mid\mid \mathtt{tla} := t_k+\tau_3
\end{array}
$$

Let S_3 be the substitution

$$\mathsf{S}_3 \stackrel{\mathrm{df}}{=} \mathsf{T_e} \mid\mid \mathtt{Efi}, \mathtt{Ela}, \mathtt{Emx}, \mathtt{tfi}, \mathtt{tla} := \mathrm{ER}_k, \mathrm{ER}_k, \mathrm{ER}_k, t_k+\tau_3, t_k+\tau_3.$$

This transition is correct if :

$$
\begin{array}{cr}
\mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge & \\
t_k < \mathtt{tfi} \quad \wedge & (G_{11}) \\
\mathtt{Emx} \leq \mathrm{ER}_k \quad \wedge & (G_{12}) \\
t_k+\tau_3 \leq \mathtt{tfi} \quad \wedge & (G_{33}) \\
\mathtt{ACR} \leq \mathrm{ER}_k \qquad \Rightarrow & (G_{34}) \\
[\mathsf{S}_3]\mathrm{Inv}. &
\end{array}
$$

Proof.

- $[\mathsf{S}_3]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\ t_k \leq t < t_k+\tau_3 \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{ACR}$: using lemma 4 and $\mathrm{I}_{\mathrm{Ub1}}$, we have $\mathrm{Approx}(k,t) = \mathrm{Approx}(k-1,t) \leq \mathtt{ACR}$.

- $[\mathsf{S}_3]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ t_k+\tau_3 \leq t \Rightarrow \mathrm{Approx}(k,t) \leq \mathrm{ER}_k$: we show $\forall t,\ t_k+\tau_3 \leq t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$; we have either $t < \mathtt{tfi}$ or $\mathtt{tfi} \leq t$;

  - in the first case, $\mathrm{Approx}(k-1,t) \leq \mathtt{ACR} \leq \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{Ub1}}$; here $t_k \leq t$ comes from $t_k \leq t_k+\tau_3 \leq t$) and (G_{34});
  - in the second case, $\mathrm{Approx}(k-1,t) \leq \mathtt{Emx} \leq \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{Apx}}$ and (G_{12});

hence $\mathrm{Approx}(k-1,t) \leq \mathrm{ER}_k$ is always true; using lemma 4 we get $\mathrm{Approx}(k,t) = \mathrm{ER}_k$ for t such that $t_k+\tau_3 \leq t$.

CASE 4

if $t_k < \mathtt{tfi}$ **then if** $\mathtt{Emx} \leq \mathrm{ER}_k$
  then if $\mathtt{tfi} < t_k+\tau_3$ **then**
  else $\mathtt{Emx} := \mathrm{ER}_k \;||\; \mathtt{Efi} := \mathrm{ER}_k \;||\; \mathtt{Ela} := \mathrm{ER}_k \;||\; \mathtt{tla} := \mathtt{tfi}$

Let S_4 be the substitution

$$\mathsf{S}_4 \;\stackrel{\text{df}}{=}\; \mathsf{T_e} \;||\; \mathtt{Efi}, \mathtt{Ela}, \mathtt{Emx}, \mathtt{tla} := \mathrm{ER}_k, \mathrm{ER}_k, \mathrm{ER}_k, \mathtt{tfi}.$$

This transition is correct if :

$$\begin{array}{lr} \mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge & \\ t_k < \mathtt{tfi} \quad \wedge & (G_{11}) \\ \mathtt{Emx} \leq \mathrm{ER}_k \quad \wedge & (G_{12}) \\ t_k+\tau_3 \leq \mathtt{tfi} \quad \wedge & (G_{33}) \\ \mathrm{ER}_k < \mathtt{ACR} \qquad \Rightarrow & (G_{44}) \\ [\mathsf{S}_4]\mathrm{Inv}. & \end{array}$$

Proof.

- $[\mathsf{S}_4]\mathrm{I}_{\mathrm{fil}}$, that is $\mathtt{tfi} \leq \mathtt{tfi} \leq t_k+\tau_2$:
  we have $\mathtt{tfi} \leq t_n+\tau_2 = t_{k-1}+\tau_2 \leq t_k+\tau_2$ by $\mathrm{I}_{\mathrm{fil}}$, definition of n and (7).
- $[\mathsf{S}_4]\mathrm{I}_{\mathrm{tfs}}$, that is $\mathtt{tfi} \leq t_k \;\Rightarrow\; \forall t,\; t_k \leq t \;\Rightarrow\; \mathrm{Approx}(k,t) \leq \mathtt{ACR}$: hypothesis absurd, given (G_{11}).
- $[\mathsf{S}_4]\mathrm{I}_{\mathrm{Et1}}$, that is $\mathtt{ACR} < \mathrm{ER}_k \Rightarrow \mathtt{tfi} \leq t_k+\tau_3$: absurd hypothesis, given (G_{44}).
- $[\mathsf{S}_4]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\; t_k \leq t < \mathtt{tfi} \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{ACR}$:
  by $\mathrm{I}_{\mathrm{Ub1}}$ and $t < \mathtt{tfi}$, we have $\mathrm{Approx}(k-1,t) \leq \mathtt{ACR}$; taking $M = \mathtt{ACR}$ in lemma 5 and using (G_{44}) yields $\mathrm{Approx}(k,t) \leq \mathtt{ACR}$.
- $[\mathsf{S}_4]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t, \mathtt{tfi} \leq t \Rightarrow \mathrm{Approx}(k,t) \leq \mathrm{ER}_k$: we show
  $\forall t,\; \mathtt{tfi} \leq t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$; for t such that $\mathtt{tfi} \leq t$, we have
  $\mathrm{Approx}(k-1,t) \leq \mathtt{Emx} \leq \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{Apx}}$ and (G_{12}); we have t_k+

$\tau_3 \leq \mathtt{tfi} \leq t$ by (G_{33}), then lemma 4 yields either $\mathrm{Approx}(k,t) = \max(\mathrm{Approx}(k-1,t), \mathrm{ER}_k)$
or $\mathrm{Approx}(k,t) = \mathrm{ER}_k$; in both cases, we see that $\mathrm{Approx}(k,t) = \mathrm{ER}_k$.

CASE 5

if $t_k < \mathtt{tfi}$ **then if** $\mathtt{Emx} \leq \mathrm{ER}_k$ **then else if** $\mathrm{ER}_k < \mathtt{Ela}$
then $\mathtt{Efi} := \mathtt{Emx} \;||\; \mathtt{Ela} := \mathrm{ER}_k \;||\; \mathtt{tla} := t_k+\tau_2$

Let S_5 be the substitution

$$\mathsf{S}_5 \stackrel{\mathrm{df}}{=} \mathsf{T_e} \;||\; \mathtt{Efi}, \mathtt{Ela}, \mathtt{tla} := \mathtt{Emx}, \mathrm{ER}_k, t_k+\tau_2.$$

This transition is correct if :

$$\begin{aligned} &\mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge \\ &t_k < \mathtt{tfi} \quad \wedge && (G_{11}) \\ &\mathrm{ER}_k < \mathtt{Emx} \quad \wedge && (G_{52}) \\ &\mathrm{ER}_k < \mathtt{Ela} \quad \Rightarrow && (G_{53}) \\ &[\mathsf{S}_5]\mathrm{Inv}. \end{aligned}$$

Proof. $[\mathsf{S}_5]\mathrm{I}_{\mathrm{fil}}$ and $[\mathsf{S}_5]\mathrm{I}_{\mathrm{tfs}}$ are similar to $[\mathsf{S}_4]\mathrm{I}_{\mathrm{fil}}$ and $[\mathsf{S}_4]\mathrm{I}_{\mathrm{tfs}}$

- $[\mathsf{S}_5]\mathrm{I}_{\max}$, that is $\mathtt{Emx} = \max(\mathtt{Emx}, \mathrm{ER}_k)$: by (G_{52}).
- $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Et1}}$, that is $\mathtt{ACR} < \mathtt{Emx} \Rightarrow \mathtt{tfi} \leq t_k+\tau_3$: we have $\mathtt{Efi} < \mathtt{Ela}$ or $\mathtt{Ela} \leq \mathtt{Efi}$;
  – in the first case, $\mathtt{tfi} \leq \mathtt{tla} \leq t_n+\tau_3$ by $\mathrm{I}_{\mathrm{fil}}$ and $\mathrm{I}_{\mathrm{Et2}}$;
  – in the second case, $\mathtt{Emx} = \mathtt{Efi}$, then $\mathtt{tfi} \leq t_n+\tau_3$ by $\mathrm{I}_{\mathrm{Et1}}$ and $\mathrm{I}_{\max}$;
  in both cases, $\mathtt{tfi} \leq t_{k-1}+\tau_3 \leq t_k+\tau_3$ by definition of n and (7).
- $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Et2}}$, that is $\mathtt{Emx} < \mathrm{ER}_k \Rightarrow t_k+\tau_2 \leq t_k+\tau_3$:
  the hypothesis $\mathtt{Emx} < \mathrm{ER}_k$ is absurd given (G_{52}).
- $[\mathsf{S}_5]\mathrm{I}_{\mathrm{ttE}}$, that is $\mathtt{tfi} = t_k+\tau_2 \Rightarrow \mathtt{Emx} = \mathrm{ER}_k$:
  we have $\mathtt{tfi} \leq t_n+\tau_2 = t_{k-1}+\tau_2 < t_k+\tau_2$ by $\mathrm{I}_{\mathrm{fil}}$, definition of n and (7); then the hypothesis $\mathtt{tfi} = t_k+\tau_2$ is absurd.
- $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\ t_k \leq t < \mathtt{tfi} \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{ACR}$:
  by $\mathrm{I}_{\mathrm{Ub1}}$ and $t < \mathtt{tfi}$, we have $\mathrm{Approx}(k-1,t) \leq \mathtt{ACR}$; we also have $\mathtt{tfi} \leq t_k+\tau_3$ or $t_k+\tau_3 < \mathtt{tfi}$;

- in the first case, $\mathrm{Approx}(k,t) = \mathrm{Approx}(k-1,t) \le \mathtt{ACR}$ by lemma 4;
- in the second case, $\mathrm{ER}_k < \mathtt{Emx} \le \mathtt{ACR}$ by (G_{52}) and contraposition of $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Et1}}$ shown above; taking $M = \mathtt{ACR}$ in lemma 5 yields
  $\mathrm{Approx}(k,t) \le \mathtt{ACR}$.

– $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Ub2}}$, that is $\forall t,\ \mathtt{tfi} \le t < t_k+\tau_2 \Rightarrow \mathrm{Approx}(k,t) \le \mathtt{Emx}$:
for t such that $\mathtt{tfi} \le t$, we have $\mathrm{Approx}(k-1,t) \le \mathtt{Emx}$, by $\mathrm{I}_{\mathrm{Apx}}$; taking $M = \mathtt{Emx}$ in lemma 5 and using (G_{52}) yields $\mathrm{Approx}(k,t) \le \mathtt{Emx}$.

– $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ t_k+\tau_2 \le t \Rightarrow \mathrm{Approx}(k,t) \le \mathrm{ER}_k$:
by lemma 4, we have $\forall t,\ t_k+\tau_2 \le t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$.

Case 6

if $t_k < \mathtt{tfi}$ **then if** $\mathtt{Emx} \le \mathrm{ER}_k$ **then else if** $\mathrm{ER}_k < \mathtt{Ela}$ **then**
else $\mathtt{Efi} := \mathtt{Emx} \ ||\ \mathtt{Ela} := \mathrm{ER}_k$

Let S_6 be the substitution

$$\mathsf{S}_6 \ \stackrel{\mathrm{df}}{=}\ \mathsf{T_e} \ ||\ \mathtt{Efi}, \mathtt{Ela} := \mathtt{Emx}, \mathrm{ER}_k.$$

This transition is correct if :

$$\begin{array}{lcr} \mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge & & \\ t_k < \mathtt{tfi} \quad \wedge & & (G_{11}) \\ \mathrm{ER}_k < \mathtt{Emx} \quad \wedge & & (G_{52}) \\ \mathtt{Ela} \le \mathrm{ER}_k \qquad \Rightarrow & & (G_{63}) \\ [\mathsf{S}_6]\mathrm{Inv}. & & \end{array}$$

Proof. $[\mathsf{S}_6]\mathrm{I}_{\mathrm{Et1}}$ and $[\mathsf{S}_6]\mathrm{I}_{\mathrm{Et2}}$ are similar to $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Et1}}$ and $[\mathsf{S}_5]\mathrm{I}_{\mathrm{Et2}}$.

– $[\mathsf{S}_6]\mathrm{I}_{\max}$, that is $\mathtt{Emx} = \max(\mathtt{Emx}, \mathrm{ER}_k)$: by (G_{52}).

– $[\mathsf{S}_6]\mathrm{I}_{\mathrm{fil}}$, that is $\mathtt{tfi} \le \mathtt{tla} \le t_k+\tau_2$: we have
$\mathtt{tfi} \le \mathtt{tla} \le t_n+\tau_2 = t_{k-1}+\tau_2 \le t_k+\tau_2$ by $\mathrm{I}_{\mathrm{fil}}$, definition of n and (7).

– $[\mathsf{S}_6]\mathrm{I}_{\mathrm{tfs}}$, that is $\mathtt{tfi} \le t_k \ \Rightarrow\ \forall t,\ t_k \le t \ \Rightarrow\ \mathrm{Approx}(k,t) \le \mathtt{ACR}$:
the hypothesis is absurd given (G_{11}).

– $[\mathsf{S_6}]\mathrm{I}_{\mathrm{ttE}}$, that is $\mathtt{tfi} = \mathtt{tla} \Rightarrow \mathtt{Emx} = \mathrm{ER}_k$:
we use a weakened form of (G_{52}):

$$\mathrm{ER}_k \leq \mathtt{Emx}; \qquad (G_{52'})$$

assuming $\mathtt{tfi} = \mathtt{tla}$, we have $\mathtt{Efi} = \mathtt{Ela} = \mathtt{Emx}$ by $\mathrm{I}_{\mathrm{max}}$, then $\mathtt{Emx} = \mathtt{Ela} \leq \mathrm{ER}_k$ by (G_{63}); $\mathtt{Emx} \leq \mathrm{ER}_k$ and $(G_{52'})$ yields $\mathtt{Emx} = \mathrm{ER}_k$.

– $[\mathsf{S_6}]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\ t_k \leq t < \mathtt{tfi} \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{ACR}$:
by $\mathrm{I}_{\mathrm{Ub1}}$ and $t < \mathtt{tfi}$, we have $\mathrm{Approx}(k-1,t) \leq \mathtt{ACR}$; we also have $\mathtt{tfi} \leq t_k+\tau_3$ or $t_k+\tau_3 < \mathtt{tfi}$;

- in the first case, $\mathrm{Approx}(k,t) = \mathrm{Approx}(k-1,t) \leq \mathtt{ACR}$ by lemma 4;
- in the second case, $\mathrm{ER}_k < \mathtt{Emx} \leq \mathtt{ACR}$ by (G_{52}) and contraposition of $[\mathsf{S_6}]\mathrm{I}_{\mathrm{Et1}}$ shown above; taking $M = \mathtt{ACR}$ in lemma 5 yields
$\mathrm{Approx}(k,t) \leq \mathtt{ACR}$.

– $[\mathsf{S_6}]\mathrm{I}_{\mathrm{Ub2}}$, that is $\forall t,\ \mathtt{tfi} \leq t < \mathtt{tla} \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{Emx}$:
for t such that $\mathtt{tfi} \leq t$, we have $\mathrm{Approx}(k-1,t) \leq \mathtt{Emx}$, by $\mathrm{I}_{\mathrm{Apx}}$; taking $M = \mathtt{Emx}$ in lemma 5 and using (G_{52}) yields $\mathrm{Approx}(k,t) \leq \mathtt{Emx}$.

– $[\mathsf{S_6}]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ \mathtt{tla} \leq t \Rightarrow \mathrm{Approx}(k,t) \leq \mathrm{ER}_k$:
for t such that $\mathtt{tla} \leq t$, we have $\mathrm{Approx}(k-1,t) \leq \mathtt{Ela} \leq \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{Ub3}}$ and (G_{63}); taking $M = \mathrm{ER}_k$ in lemma 5 yields $\mathrm{Approx}(k,t) \leq \mathrm{ER}_k$.

Note that if we can ensure $t_k+\tau_3 \leq \mathtt{tla} \leq t$, we can show
$\forall t,\ \mathtt{tla} \leq t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$.

CASE 7

if $t_k < \mathtt{tfi}$ **then else if** $\mathtt{ACR} \leq \mathrm{ER}_k$ **then**

$$\begin{array}{l} \mathtt{Efi} := \mathrm{ER}_k \parallel \mathtt{Ela} := \mathrm{ER}_k \parallel \mathtt{Emx} := \mathrm{ER}_k \\ \qquad\qquad\ \parallel \mathtt{tfi} := t_k+\tau_3 \parallel \mathtt{tla} := t_k+\tau_3 \end{array}$$

Let $\mathsf{S_7}$ be the substitution

$$\mathsf{S_7} \stackrel{\mathrm{df}}{=} \mathsf{T_e} \parallel \mathtt{Efi},\mathtt{Ela},\mathtt{Emx},\mathtt{tfi},\mathtt{tla} := \mathrm{ER}_k,\mathrm{ER}_k,\mathrm{ER}_k,t_k+\tau_3,t_k+\tau_3.$$

This transition is correct if :

$$\mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge$$

$$\texttt{tfi} \leq t_k \quad \wedge \qquad (G_{71})$$

$$\texttt{ACR} \leq \mathrm{ER}_k \qquad \Rightarrow \qquad (G_{72})$$

$$[\mathsf{S}_7]\mathrm{Inv}.$$

Proof.

- $[\mathsf{S}_3]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\ t_k \leq t < t_k+\tau_3 \Rightarrow \mathrm{Approx}(k,t) \leq \texttt{ACR}$:
  we remark that $\texttt{utd} = \mathrm{true}$ by (G_e) and $\texttt{tfi} \leq s$ by (G_{71}), then $\mathrm{Approx}(k{-}1,t) \leq \texttt{ACR}$ by $\mathrm{I}_{\mathrm{tfs}}$; using lemma 4 we get $\mathrm{Approx}(k,t) = \mathrm{Approx}(k-1,t)$, hence $\mathrm{Approx}(k,t) \leq \texttt{ACR}$.

- $[\mathsf{S}_3]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ t_k+\tau_3 \leq t \Rightarrow \mathrm{Approx}(k,t) \leq \mathrm{ER}_k$:
  we show $\forall t,\ t_k+\tau_3 \leq t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$; we remark that $\texttt{utd} = \mathrm{true}$ by (G_e) and $\texttt{tfi} \leq s$ by (G_{71}), then $\mathrm{Approx}(k{-}1,t) \leq \texttt{ACR} \leq \mathrm{ER}_k$ by $\mathrm{I}_{\mathrm{tfs}}$; here $t_k < t$ comes from $t_k < t_k+\tau_3 \leq t$) and (G_{72}); using the assumption $t_k+\tau_3 \leq t$ and lemma 4, we know that $\mathrm{Approx}(k,t)$ is either equal to $\max(\mathrm{Approx}(k-1,t), \mathrm{ER}_k)$ or to ER_k, that is, in both cases, to ER_k.

CASE 8

if $t_k < \texttt{tfi}$ **then else if** $\texttt{ACR} \leq \mathrm{ER}_k$ **then else**

$$\begin{array}{lll} \texttt{Efi} := \mathrm{ER}_k \;\|\; \texttt{Ela} := \mathrm{ER}_k & \|\; \texttt{Emx} := \mathrm{ER}_k \\ \qquad\qquad\quad \|\; \texttt{tfi} := t_k+\tau_2 & \|\; \texttt{tla} := t_k+\tau_2 \end{array}$$

Let S_8 be the substitution

$$\mathsf{S}_8 \stackrel{\mathrm{df}}{=} \mathsf{T}_{\mathsf{e}} \;\|\; \texttt{Efi}, \texttt{Ela}, \texttt{Emx}, \texttt{tfi}, \texttt{tla} := \mathrm{ER}_k, \mathrm{ER}_k, \mathrm{ER}_k, t_k+\tau_2, t_k+\tau_2.$$

This transition is correct if :

$$\mathrm{Inv} \quad \wedge \quad (G_e) \quad \wedge$$

$$\texttt{tfi} \leq t_k \quad \wedge \qquad (G_{71})$$

$$\mathrm{ER}_k < \texttt{ACR} \qquad \Rightarrow \qquad (G_{82})$$

$$[\mathsf{S}_8]\mathrm{Inv}.$$

Proof.

- $[\mathsf{S}_8]\mathrm{I}_{\mathrm{tfs}}$ and $[\mathsf{S}_8]\mathrm{I}_{\mathrm{Ub2}}$ are similar to $[\mathsf{S}_7]\mathrm{I}_{\mathrm{tfs}}$ and $[\mathsf{S}_7]\mathrm{I}_{\mathrm{Ub2}}$, replacing τ_3 with τ_2.

- $[\mathsf{S}_8]\mathrm{I}_{\mathrm{Et1}}$, that is $\mathtt{ACR} < \mathrm{ER}_k \Rightarrow t_k + \tau_2 \leq t_k + \tau_3$: the hypothesis $\mathtt{ACR} < \mathrm{ER}_k$ is absurd given (G_{82}).

- $[\mathsf{S}_8]\mathrm{I}_{\mathrm{Ub1}}$, that is $\forall t,\ t_k \leq t < t_k + \tau_2 \Rightarrow \mathrm{Approx}(k,t) \leq \mathtt{ACR}$:
for t such that $t_k \leq t$, we have $\mathtt{utd} = \mathrm{true}$ by (G_e) and $\mathtt{tfi} \leq s$ by (G_{71}), then
$\mathrm{Approx}(k-1,t) \leq \mathtt{ACR}$ by $\mathrm{I}_{\mathrm{tfs}}$; taking $M = \mathtt{ACR}$ in lemma 5 and using (G_{82}) yields $\mathrm{Approx}(k,t) \leq \mathtt{ACR}$.

- $[\mathsf{S}_8]\mathrm{I}_{\mathrm{Ub3}}$, that is $\forall t,\ t_k + \tau_2 \leq t \Rightarrow \mathrm{Approx}(k,t) \leq \mathrm{ER}_k$:
by lemma 4, we have $\forall t,\ t_k + \tau_2 \leq t \Rightarrow \mathrm{Approx}(k,t) = \mathrm{ER}_k$.

References

1. 'Traffic control and congestion control in B-ISDN'. ITU-T. Recommendation I.371.1.
2. Abrial, J.-R.: 1996, *The B-Book: Assigning Programs to Meanings*. Cambridge University Press.
3. Abrial, J.-R.: 1999, 'Développement de l'algorithme ABR'. Personal communication.
4. Alur, R. and D. L. Dill: 1994, 'A theory of timed automata'. *Theoretical Computer Science* **126**(2), 183–235.
5. Arnold, A.: 1990, 'MEC: A System for Constructing and Analysing Transition Systems'. In: J. Sifakis (ed.): *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, Vol. 407 of *LNCS*. Berlin, pp. 117–132.
6. Barras, B. and all: 1997, 'The Coq Proof Assistant Reference Manual : Version 6.1'. Technical Report RT-0203, INRIA.
7. Bengtsson, J., K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi: 1996, 'UPPAAL: a tool suite for the automatic verification of real-time systems'. In: R. Alur, T. A. Henzinger, and E. D. Sontag (eds.): *Hybrid Systems III*, Vol. 1066 of *Lecture Notes in Computer Science*. pp. 232–243.
8. Bérard, B. and L. Fribourg: 1999, 'Automated verification of a parametric real-time program: the ABR conformance protocol'. In: *CAV'99*. To appear.
9. Bérard, B., L. Fribourg, F. Klay, and J.-F. Monin: 1999, 'A compared study of two correctness proofs for the standardized algorithm of ABR conformance'. Report LSV-99-7, ENS de Cachan.
10. Clark, D., E. M. Emerson, and A. P. Sistla: 1983, 'Automatic verification of finite state concurrent systems using temporal logic specifications: a practical approach'. In: *Proc. 10th ACM Symp. on Principles of Programming Languages*.
11. Dijkstra, E. W.: 1976, *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs, NJ.
12. Fribourg, L.: 1998, 'A Closed-Form Evaluation for Extended Timed Automata'. Research Report LSV-98-2, Lab. Specification and Verification, ENS de Cachan, Cachan, France. 17 pages.

13. Halbwachs, N.: 1993, *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers.
14. Henzinger, T. A., P.-H. Ho, and H. Wong-Toi: 1997, 'HYTECH: A model checker for hybrid systems'. *Lecture Notes in Computer Science* **1254**, 460–463.
15. Jard, C., J.-F. Monin, and R. Groz: 1988, 'Development of Veda, a Prototyping Tool for Distributed Algorithms'. *IEEE Transactions on Software Engineering* **14**(3), 339–352.
16. Monin, J.-F.: 1998, 'Proving a real time algorithm for ATM in Coq'. In: E. Gimenez and C. Paulin-Mohring (eds.): *Types for Proofs and Programs*, Vol. 1512 of *LNCS*. Springer Verlag, pp. 277–293.
17. Monin, J.-F. and F. Klay: 2000, 'Correctness Proof the Standardized Algorithm for ABR Conformanc'. In: J. Wing, J. Woodcock, and J. Davies (eds.): *FM'99*, Vol. 1708 of *LNCS*. Springer Verlag, pp. 662–681.
18. Owre, S., J. M. Rushby, and N. Shankar: 1992, 'PVS: A prototype verification system'. In: D. Kapur (ed.): *11th International Conference on Automated Deduction (CADE)*, Vol. 607 of *LNCS*. Springer Verlag, pp. 748–752.
19. Queille, J. P. and J. Sifakis: 1982, 'Specification and Verification of Concurrent Systems in CESAR'. In: *Proc. 5th Int'l Symp. on Programming*, Lecture Notes in Computer Science, Vol. 137. Berlin/New York: SV, pp. 337–371.
20. Rabadan, C.: 1997, 'L'ABR et sa conformité'. NT DAC/ARP/034, CNET.
21. Revesz, P. Z.: 1993, 'A closed-form evaluation for Datalog queries with integer (gap)-order constraints'. *Theoretical Computer Science* **116**(1), 117–149.
22. Rusinowitch, M., S. Stratulat, and F. Klay: 1999, 'Mechanical Verification of a Generic Incremental ABR Conformance Algorithm'. Technical Report RT-3794, INRIA.