A New Model for Hard Real-Time Systems

Patrick Meumeu Yomsi
INRIA Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: patrick.meumeu@inria.fr

Yves Sorel
INRIA Rocquencourt
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex - France
Email: yves.sorel@inria.fr

Abstract

In this paper we study hard real-time systems composed of periodic preemptive tasks and address the scheduling problem on specific hardware platforms. For such a system, the failure to satisfy any constraint may have disastrous consequences. An important starting point for the analysis of such systems is the basic task model considered. In this paper we introduce a new model, relative to the ones in the literature, that is able to obtain stronger schedulability conditions. This new model provides a formal and rigorous specification of a task and helps us to describe the schedulability principle when tasks are scheduled by using a fixed-priority scheduling policy.

Keywords: Real-time systems, Task models, Fixed-priority scheduling theory.

1 Introduction

This paper focuses on hard real-time monoprocessor systems and addresses the scheduling problem of periodic preemptive tasks on specific hardware platforms without cache, pipeline, or complex internal architecture, when tasks are scheduled according to a fixed-priority scheduling policy. Up to now, many models and concepts necessary to describe and analyse hard real-time systems have been proposed. Rich and extensive state of the art work has been performed in order to justify the considered assumptions, etc. Over the years, preemptive periodic task models [1, 2] have proven remarkably useful for the modelling of hard real-time systems — systems where the failure to satisfy any constraint may have disastrous consequences [2]. Unfortunately, none of the previously proposed models has been designed to take into account an issue such as the exact cost of preemption [3]. This weakness in current existing models may lead to erroneous conclusions in terms of schedulability decisions. This in turn can affect the correct behavior of the system at runtime, or in any case lead to resources being wasted. In addition, this model allows the unification in one framework of different models such as Liu & Layland's and Mok's models. In this paper, we introduce a new model to solve the general scheduling problem of hard real-time systems while taking into account the exact cost of the Real-Time Operating System (*RTOS*) [4]. Indeed, the preemption cost represents only one half of the RTOS cost which consists of two parts. A constant part, easy to determine, which corresponds to the cost of the scheduler, is associated with the activation and termination of tasks. The activation of a task includes the context switch necessary to make possible the preemption of another task and the choice of the task with the highest priority. Thus, this cost only depends on the number of tasks. A variable part, which is more difficult to determine, is associated with the occurrence of every preemption of the current task in order for it to resume later on. Thus, this cost depends on the number of preemptions for every task. We consider all the possible scenarios of first activation for all the tasks (i.e. *simultaneous* or *non-simultaneous*). Currently, there exists a wide gap between the scheduling theory and its implementation in operating systems running on specific hardware platforms. This paper provides a first step toward bridging the gap between real-time scheduling theory and implementation realities. Surely, this gap must be bridged for any meaningful validation of timing correctness. Throughout the paper, we assume that all timing characteristics are non negative integers, i.e. they are multiples of some elementary time interval (for example the "CPU tick", the smallest indivisible CPU time unit). The general scheduling problem in this case consists in filling the available time units left after the schedule of some tasks with the execution time units of the other tasks. Based on our new model, we are able to propose a schedulability analysis which uses a binary operation $\oplus$ whose operands are called *otasks*. In the following, we show the correspondence between notions and concepts belonging to the classical real-time scheduling theory and those introduced in our model. Also, we introduce the binary operation $\oplus$ when priorities are assigned according to a fixed-priority scheduling policy such as *Rate Monotonic*, *Deadline Monotonic*, *Audsley* [1, 5].

The remainder of the paper is structured as follows: section 2 gives definitions and properties used throughout this paper and presents the new *"otask model"* that will help us to perform a new methodology for the schedulability of a hard real-time system. Section 3 provides the correspondence between periodic tasks and periodic otasks. Section 4 gives the schedulability principle when priorities are assigned according to a fixed-priority scheduling policy. We conclude and propose future work in section 5.

2 Definitions and properties

In this section we introduce some definitions and properties in order to provide the reader with the framework of our new model for hard real-time systems.

First of all, we must specify a generator containing all the legal symbols which can be used. We define a *generator* Σ as being a finite set of symbols. As such, everywhere in this paper,

the generator that will be considered is $\Sigma = \{a, e\}$. In the context of scheduling theory, we will always associate the symbol "a" to a time unit which is *available* and the symbol "e" to a time unit which is either *executed* or *executable* depending on the cases that we will detail later on. Now, given this *generator* Σ, we define an *otask* as an *ordered multiset* consisting of a certain number of elements (possibly zero) all belonging to Σ. The fundamental difference we make here between the notion of *ordered multiset* and the common notion of *multiset* [6] is that the order of elements in an otask is important in our case. In fact, this will allow us to make the difference between any two otasks and in particular between otasks obtained from permutations of the elements of another otask. We define an *otask system* Γ as a set of otasks on the generator Σ. It is worth noticing that the definition of an *otask system* is quite general and allows us to consider highly structured otask systems such as periodic, aperiodic or hybrid otask systems. The proximity of the terminologies used here to those in the literature expresses the idea of a relationship between them which we will detail later on. Indeed, we will consider an otask with specific properties to represent a *real-time task*, and thus a *system of real-time tasks* is a particular *system of otasks*. We consider a unique set of otasks where each real-time task corresponds to one and only one otask. This correspondence will allow us to derive results on real-time task systems from those obtained on otask systems. In order to illustrate the previous definitions here are some examples of otasks on Σ: $\tau_1 = \{a\}$, $\tau_2 = \{e, a, a\}$, $\tau_3 = \{a, e, a\}$ and $\tau_4 = \{a, a, e, e, a\}$. The otask with no symbols, is denoted by Λ ($\Lambda = \{\} = \emptyset$). Λ is always an otask on Σ.

The set of all possible otasks on Σ will be denoted by Σ^*. Hence, any otask system Γ is necessarily a subset of Σ^*. If τ is an otask on Σ, then the *cardinal* of τ is the number of elements in τ and will be denoted by $|\tau|$. Now, let x and y be two otasks on Σ. The *concatenation* of x and y is the otask xy obtained by writing the symbols of x and the symbols of y consecutively. As an example, if $x = \{a, e, e\}$ and $y = \{e, a\}$ then the otask xy is given by $xy = \{a, e, e, e, a\}$ and the otask yx is given by $yx = \{e, a, a, e, e\}$. We have $xy \neq yx$ because of the importance of the order of the elements in an otask. Consequently, the *concatenation* operation is not *Commutative*, i.e. there are otasks x and y on Σ such that xy is different from yx. However, this operation is *associative*, i.e. for all otasks x, y and z on Σ, $(xy)z = x(yz)$. The advantage of the *associativity* is that it allows us to concatenate several otasks without worrying about the order in which the concatenation operations are carried out. Note that for any otask x, the concatenation of x and Λ equals x, i.e. $x\Lambda = \Lambda x = x$. If there exist two otasks w and z such that $y = wxz$, then x is called a *sub-otask* of y. We call the operation leading to obtain a sub-otask from an otask an *extraction*. In order to illustrate the latter definition, the otask $\{a, e, e\}$ is a sub-otask of each otask $\{e, e, a, e, e, a, a\}$ and $\{a, a, a, e, e\}$ since we have for example $\{e, e, a, e, e, a, a\} = \{e, e\}\{a, e, e\}\{a, a\}$, but is not a sub-otask of $\{a, e, a, e\}$.

From now on, the superscripts will represent the number of times an element is concatenated. These elements can either be simple otasks, or even otask systems. Thus, if $x \in \Sigma^*$, and $\Gamma \subseteq \Sigma^*$, then: $x^k = xx\cdots x$, $\Sigma^k = \Sigma\Sigma\cdots\Sigma = \{x \in \Sigma^* / |x| = k\}$, $\Gamma^k = \Gamma\Gamma\cdots\Gamma$, where in each case, there are k factors that are concatenated.

Keeping in mind that we are interested in scheduling periodic task sets, we consider otasks with an infinite cardinal. An otask where a sub-otask with a finite cardinal can be extracted and the otask is an infinite concatenation of this sub-otask from a certain element relatively to the first one, will be termed *periodic*. An otask with an infinite number of the symbol "e", and with a minimum number of symbols between two consecutive sequence of symbols "e", will be termed *sporadic*. Finally, an otask that contains a finite number of symbols "e" will be termed *aperiodic*. Hereafter, we will only consider *periodic* otasks. An otask system Γ consisting only of periodic otasks will be called a *system of periodic otasks*.

So far, the notion of "time", which is central to scheduling theory, has not yet been considered in this paper. In order to overcome this, we consider an index along an oriented time axis which is a temporal reference for all otasks. On this axis, we identify un instant of reference t_0, for example we can choose $t_0 = 0$. Hence, in addition to the cardinal and the order of elements that can help us to differentiate between any two otasks on the generator Σ, the *start date* $r \in \mathbb{Z}$ and *end date* $f \in \mathbb{Z}$ of each otask w.r.t. the reference time t_0 are important and may also help us to differentiate between two otasks. We will denote by $\tau_{(r,f)}$ the otask on Σ which starts at date r and finishes at date f. This notation allows us to describe both finite and infinite cardinal otasks. If $f = r$ then $\tau = \Lambda$. If $f = \infty$ then $|\tau| = \infty$ and in this case we denote by convention $\tau_{(r,\infty)} = \tau_r$ as there is no ambiguity concerning the end date of τ.

By definition, a *periodic otask* τ_{per} on Σ is an infinite cardinal otask with a start date r such that there exists a finite cardinal sub-otask τ, and τ_{per} is an infinite concatenation of τ from a certain time instant $\beta \geq r$. We denote each periodic otask by:

$$\tau_{per} = \zeta_{(r,\beta)}\, \tau_{\beta}^{\infty} \tag{1}$$

where the integer β represents the smallest time instant such that relation 1 is satisfied, $\zeta_{(r,\beta)}$ represents a finite cardinal otask called *initial part* of τ_{per} and τ_{β}^{∞} represents an infinite cardinal otask, infinite concatenation of τ from date β called *periodic part* of τ_{per}.

For the sake of clarity, the infinite cardinal otask $\tau_1 = \{a, a, e, \underline{e, e, a, e, a}_{\tau}, \underline{e, e, a, e, a}_{\tau}, \cdots, \underline{e, e, a, e, a}_{\tau}, \cdots\}_0 = \{a, a, e\}_{(0.3)}\{e, e, a, e, a\}_3^{\infty}$ is a periodic otask on $\Sigma = \{a, e\}$ whose initial part is $\{a, a, e\}_{(0.3)}$ and the periodic part is $\{e, e, a, e, a\}_3^{\infty}$. A periodic otask system on Σ is given for example by the set $\Gamma = \{\{a, a, e\}_{(0.3)}\{e, e, a, e, e\}_3^{\infty}, \{a, e, e, a\}_{(12.16)}\{e, a, a, e, e, a, a, a\}_{16}^{\infty}\}$.

Given a periodic otask, there are two different forms in which it may be written: the *factored form* and the *developed form*. For the case of otask τ_1, $\{a, a, e\}_{(0.3)}\{e, e, a, e, a\}_3^{\infty}$ will be referred to as the *factored form* and $\{a, a, e, \underline{e, e, a, e, a}_{\tau}, \underline{e, e, a, e, a}_{\tau}, \cdots, \underline{e, e, a, e, a}_{\tau}, \cdots\}_0$ will be referred to as the *developed form*. Now, let τ_{per} be a periodic otask on Σ. If the existence of the integer β such that $\tau_{per} = \zeta_{(r,\beta)}(\tau)_{\beta}^{\infty}$ is unique by definition, the existence of the finite sub-otask τ is not unique. Indeed, the otask τ_1 can also be written $\tau_1 = \{a, a, e\}_{(0.3)}\{\underline{e, e, a, e, a}_{\tau}, \underline{e, e, a, e, a}_{\tau}\}_3^{\infty}$ and we have $5 = |\{e, e, a, e, a\}| \neq |\{e, e, a, e, a, e, e, a, e, a\}| = 10$.

We define the *pattern* of a periodic otask $\tau_{per} = \zeta_{(r,\beta)}\, \tau_{\beta}^{\infty}$ to be the *minimum* finite cardinal sub-otask τ_{min} of the otask τ such that $\tau_{per} = \zeta_{(r,\beta)}(\tau_{min})_{\beta}^{\infty}$. The *pattern* of any periodic otask always exists and is unique. Consequently, any two periodic otasks $\tau_1 = \zeta_{i_{(r_i,\beta_i)}}(\tau_i)_{\beta_i}^{\infty}$ and $\tau_2 = \zeta_{j_{(r_j,\beta_j)}}(\tau_j)^{\infty}$ on Σ are *equal*

if and only if on the one hand $\zeta_{i_{(r_i,\beta_i)}} = \zeta_{j_{(r_j,\beta_j)}}$ and on the other hand τ_1 and τ_2 have the same *pattern*. In the same vein, τ_1 and τ_2 are said to be *equivalent* if and only if they have the same developed form. In the remainder of this paper, any periodic otask τ_{per} on Σ will be denoted by $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$ where $\zeta_{(r,\beta)}$ is the initial part, τ is the pattern and $T = |\tau|$ is the *period* of τ_{per}.

We assume that the pattern τ of τ_{per} contains at least one symbol "*e*". Indeed, if $\tau = \{a\}$ then we consider that τ_{per} equals the finite cardinal otask $\zeta_{(r,\beta)}$, i.e. $\tau_{per} = \zeta_{(r,\beta)}\{a\}_\beta^\infty = \zeta_{(r,\beta)}$ which is not interesting as it is not periodic. Similarly, we assume that the pattern τ contains at least one symbol "*a*". Indeed, if it is not the case, by identifying the symbol "*e*" to a time unit which is either *executed* or *executable*, then the periodic otask $\zeta_{(r,\beta)}\{e\}_\beta^\infty$ corresponds to one whose elements do not change from a certain date, the date β. This situation is not interesting since our goal is to compose otasks by replacing the available time units of a otask, i.e. the symbols "*a*", by the executable time units of another otask, the symbols "*e*".

It is worth noticing that any periodic otask $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$ of period T on Σ is equivalent to an infinite number of periodic otasks of period T on Σ. Indeed, since we have $\tau \neq \Lambda$ and $|\tau|$ finite by definition, then there exists a finite otask $x \neq \Lambda$ and a finite otask y such that $|\tau| = |x| + |y|$ and $\tau_{(\beta,\beta+|\tau|)} = x_{(\beta,\beta+|x|)}y_{(\beta+|x|,\beta+|\tau|)}$. Thus:

$$\begin{aligned}
\tau_{per} &= \zeta_{(r,\beta)}\tau_\beta^\infty \\
&= \zeta_{(r,\beta)}(xy)_\beta^\infty \\
&= \zeta_{(r,\beta)}(xy)(xy)(xy)\cdots(xy)(xy)\cdots \\
&= \zeta_{(r,\beta)}x(yx)(yx)(yx)\cdots(yx)(yx)\cdots \\
&= \left(\zeta_{(r,\beta)}x_{(\beta,\beta+|x|)}\right)(yx)_{\beta+|x|}^\infty = \tau'_{per}
\end{aligned}$$

The otask τ'_{per} is periodic of period T and its initial part is $\zeta_{(r,\beta)}x_{(\beta,\beta+|x|)}$ and its periodic part is $(yx)_{\beta+|x|}^\infty$. As the otasks x and y are arbitrary, we can repeat this process as many times as we want. In particular, for any $k \in \mathbb{N}$, we also have

$$\tau_{per} = \zeta_{(r,\beta)}\underbrace{\tau\tau\tau\cdots\tau}_{k\ times}\tau_{\beta+k|\tau|}^\infty = \zeta_{(r,\beta)}\tau_{(\beta,\beta+k|\tau|)}^k\tau_{\beta+k|\tau|}^\infty \qquad (2)$$

where $\tau_{(\beta,\beta+k|\tau|)}^k$ denotes the finite otask beginning at date β and ending at date $\beta + k|\tau|$, it corresponds to the otask τ concatenated k times. We will say that τ_{per} is in the *canonical form* if and only if the first element of the finite cardinal otask τ is the symbol "*e*". Thanks to everything we have presented up to now, the periodic otask $\tau_2 = \{a,a,e\}_{(0,3)}\{a,a,e,e,a,e,a\}_3^\infty$ is not in the *canonical form*, but it is *equivalent* to the otask $\tau'_2 = \{a,a,e,a,a\}_{(0,5)}\{e,e,a,e,a,a,a\}_5^\infty$ which is in the *canonical form*. This transformation is useful as the schedule will consist in replacing symbols "*a*" belonging to an otask by symbols "*e*" belonging to another otask. From now on, for each periodic otask $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$, we consider the equivalent periodic canonical otask $\tau'_{per} = \zeta_{(r,\beta')}\tau_{\beta'}^\infty$ where β' is the smallest integer greater than β. We define the *relative deadline* D of a periodic otask $\tau_{per} = \zeta_{(r,\beta)}\tau_\beta^\infty$ to be an integer value equal to $\beta - r$ for the initial part of τ_{per}, and equal to the cardinal of a single sub-otask, possibly the pattern itself, containing at least all the symbols "*e*" of the pattern for the periodic part of τ_{per}. D is at most equal to T.

Since the definition of the relative deadline does not present any ambiguity for the initial part, it is not necessary to represent it graphically. However, for the periodic part, the deadline will be represented by a checkmark: ⌟. At this point we have everything we need to introduce our model of periodic otasks.

Figure 1 illustrates a periodic otask with relative deadline D and period T. Each shaded box corresponds to the symbol "*e*" and each non-shaded box to the symbol "*a*" in the generator Σ. The initial part which is finite, is between the dates r and β. The pattern of the periodic part, which repeated infinitely, is comprised between β and $\beta + T$. In this figure, D can take 5 possible values relative to the position of the last symbol "*e*" in the periodic part of the otask. These values are $\{T, T-1, T-2, T-3, T-4\}$. Note that in our model, the value of the relative deadline for the periodic part of any periodic otask is *less than or equal* to its period.

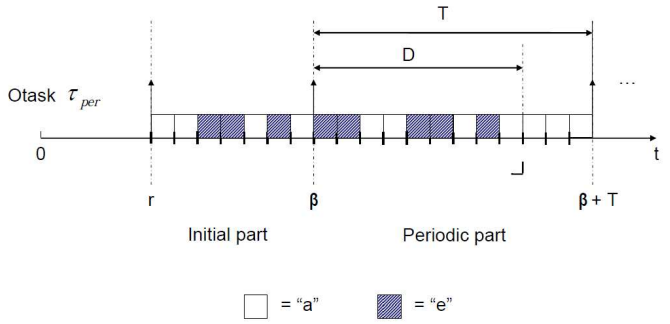

Figure 1. Model of a periodic otask.

We call *date of sub-activation of rank l for the initial part* of τ_{per} denoted by r_{in}^l (resp. *date of first sub-activation of rank l for the periodic part* of τ_{per} denoted by $r_p^{l,1}$) the date of occurence of the first symbol "*e*" belonging to the sequence of rank l in the initial part of τ_{per} relatively to r (resp. the date of occurence of the first symbol "*e*" belonging to the sequence of rank l of the pattern for the periodic part of τ_{per} relative to β). Identically, we call *sub-execution time of rank l of the initial part* (resp. *sub-execution time of rank l of the periodic part*) denoted C_{in}^l (resp. C_p^l) the cardinal of the sub-otask which consists only of symbols "*e*" corresponding to the sequence of rank l. Figure 2 below clarifies these notions of *date of sub-activation* and *sub-execution time* for a periodic otask.

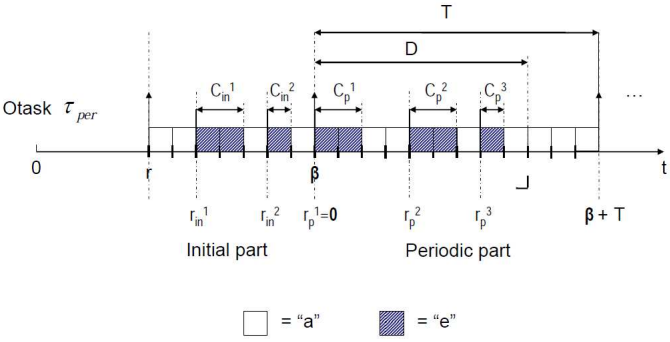

Figure 2. dates of sub-activations and sub-execution times for a periodic otask.

3 Model of periodic tasks

The study of a periodic real-time system by using a periodic otask system requires that each periodic task is describable uniquely as a periodic otask, that is to say that two distinct periodic tasks must match two distinct periodic otasks. In this

section, we choose to build such a correlation by describing how each otask can be generated from the temporal characteristics of each real-time task and operations on simpler otasks.

Let $\Gamma_n = \{\tau_1, \tau_2, \cdots, \tau_n\}$ be a system of n periodic tasks where $\tau_i = (r_i^1, C_i, D_i, T_i)$ and $C_i \leq D_i \leq T_i$. Based on the characteristics a periodic task, r_i^1 is the date of first activation, C_i is the Worst Case Execution Time (WCET) without any approximation of the preemption cost, D_i is the relative deadline and T_i is the period of τ_i. Relation 3 provides the periodic otask $o\tau_i$ which corresponds to the periodic task τ_i.

$$o\tau_i = \left\{ \underbrace{\overbrace{e, e, \cdots, e}^{C_i}, a, a, a, \cdots, \overbrace{a}^{D_i}, a, \cdots, a, a}_{T_i} \right\}_{r_i^1}^{\infty} \quad (3)$$

where r_i^1 means that the pattern of otask $o\tau_i$ begins at the date r_i^1, corresponding to the date of first activation of task τ_i. It thus follows that the otask $o\tau_i$ is a *particular otask* since it is *canonical*. It consists of a periodic part but has not got a non-trivial initial part, indeed its initial part equals Λ. Furthermore it is *regular*, that is to say that the pattern contains a single sequence of C_i symbols "*e*" followed by a single sequence of $T_i - C_i$ symbols "*a*". The D_i first symbols of the pattern represent the relative deadline of the otask $o\tau_i$. The value of D_i delimits the interval before which C_i symbols "*e*" of $o\tau_i$ must have been executed. In equality 3 each repetition of the pattern from the date r_i^1 corresponds to an instance of the task τ_i. The pattern of rank k starting at the date $r_i^k = r_i^1 + (k-1)T_i$ corresponds to the k^{th} instance. Figure 3 illustrates a periodic task as a particular periodic otask given in figure 2.

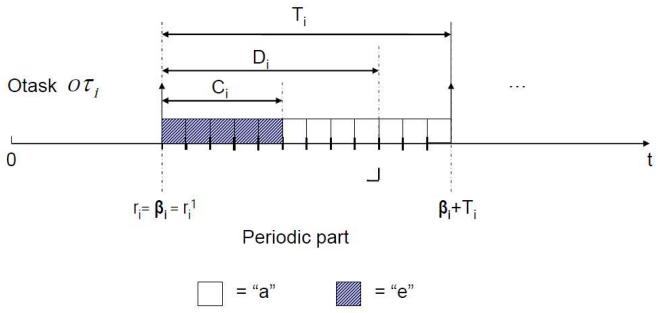

Figure 3. Correspondence between a periodic task and a periodic otask.

4 Schedulability principle

Our main objective is the schedulability analysis of a system of periodic tasks by considering the corresponding otask system. For this purpose, we will combine otasks by using an *associative non commutative binary scheduling operation* that we denote by $\oplus$ in order to get an otask that will help us decide the schedulability. When we write $x \oplus y$ where x and y are two periodic otasks, this means by convention that the first operand (otask x) has a higher priority than the second operand (otask y), therefore the operation $\oplus$ is *not commutative*, i.e. $x \oplus y \neq y \oplus x$. Now we have everything we need to explain the difference between *executed* and *executable* symbols "*e*". In the expression $x \oplus y$, the elements "*e*" of otask x are called *executed* and those of otask y are called *executable*. The intuitive idea that we propose to perform the operation $\oplus$ will therefore consist in replacing some elements "*a*" of a copy z of otask x by elements "*e*" of otask y, leading to the result $z = x \oplus y$. Although there are not enough "*a*" for all the *executable* "*e*", $x \oplus y = \Lambda$ is defined. When performing operation $\oplus$ the date of sub-activation of each sequence of executable symbols "*e*" of otask y gives the earliest date of the symbols "*a*" to replace in otask z.

For any otask system $O\Gamma_n = \{o\tau_1, o\tau_2, \cdots, o\tau_n\}$ arranged according to decreasing priorities relative to an algorithm such as *Rate Monotonic* or *Deadline Monotonic*, since $\oplus$ is a binary operation, it will be used as many times as there are otasks in $O\Gamma_n$ in order to guarantee, or not, the schedulability of the system. The operations $\oplus$ will be applied from the otask with the highest priority to the otask with the lowest priority. This process will produce an intermediate result otask at each step which corresponds to the otask with the highest priority, i.e. the left-hand operand of the next the operation $\oplus$. Consequently, if $\mathcal{R}_n$ is the scheduling otask result of $O\Gamma_n$, then $\mathcal{R}_n$ is obtained by successive iterations:

$$\begin{cases} \mathcal{R}_1 = \Lambda \oplus o\tau_1 = o\tau_1 \\ \mathcal{R}_i = \mathcal{R}_{i-1} \oplus o\tau_i, \quad 2 \leq i \leq n \end{cases}$$

The otask $o\tau_i$ will be said *schedulable* with respect to the considered priorities policy if and only if $\mathcal{R}_i \neq \Lambda$ and the system $O\Gamma_n$ will be said *schedulable* if and only if all the otasks are schedulable. If this is not the case, then the system $O\Gamma_n$ is said *not schedulable*.

5 Conclusion and future work

In this paper we have introduced a new model, relative to the ones in the literature, that provides a more accurate specification of a real-time task. It helped us to describe the schedulability analysis when tasks are scheduled using a fixed-priority scheduling policy by defining an associative non commutative binary scheduling operation that we have denoted by $\oplus$. Future work will use this new model in order to provide schedulability conditions when the RTOS cost is taken into account. Indeed, the preemption cost which is the variable part of the RTOS cost necessitates an accurate model.

References

[1] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.

[2] R. Pellizzoni and G. Lipari. Feasibility analysis of real-time periodic tasks with offsets. *Real-Time Systems*, 2005.

[3] I. J. Bate. *Scheduling and Timing Analysis of Safety Critical Real-Time Systems*. PhD thesis, University of York, 1998.

[4] P. Meumeu Yomsi. *Prise en compte du coût exact de la préemption dans l'ordonnancement temps réel monoprocesseur avec contraintes multiples*. PhD thesis, Université de Paris Sud, Spécialité Physique, 02/04/2009.

[5] N.C. Audsley, A. Burns, M.F. Richardson, Tindell K., and A.J. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 1993.

[6] Z. Manna and R. Waldinger. *The Logical Basis for Computer Programming*. volume 1: Deductive Reasoning. Addison-Wesley, Reading, Massachusetts, 1985.