Nordic Journal of Computing 14(2008), 173–201.

FORMALISING SCHEDULING THEORIES IN DURATION CALCULUS

QIWEN XU
Faculty of Science and Technology
University of Macau
Macau SAR, P.R. China
qwxu@umac.mo

NAIJUN ZHAN*
Lab. of Computer Science, Institute of Software
Chinese Academy of Sciences
100080, Beijing, P.R. China
znj@ios.ac.cn

Abstract. Traditionally many proofs in real time scheduling theory were informal and lacked the rigor usually required for good mathematical proofs. Some attempts have been made towards making the proofs more reliable, including using formal logics to specify scheduling algorithms and verify their properties. In particular, Duration Calculus, a real time interval temporal logic, has been used since timing requirements in scheduling can be naturally associated with intervals. This paper aims to improve the work in this area and give a summary. Static and dynamic priority scheduling algorithms are formalised in Duration Calculus and classical theorems for schedulability analysis are proven using the formal proof system of Duration Calculus.

ACM CCS Categories and Subject Descriptors: X.X.X [not known yet]

Key words: Real time scheduling, schedulability conditions, temporal logics, duration calculus, formal proof

1. Introduction

In the classical theory for real time scheduling, scheduling algorithms were invented and then schedulability conditions, i.e., conditions that decide whether or not the set of tasks will meet their timing requirements, were established. For example, in the seminal work by Liu and Layland [Liu and Layland 1973], two scheduling algorithms, i.e., Rate Monotonic Scheduler (RM) and Earliest Deadline First (EDF), were proposed, and schedulability conditions for them were studied. The correctness of the schedulability conditions is not trivial, and therefore needs to be proved as mathematical theorems.

However, in many cases, including those in relatively recent book [Buttazzo 1997], the proofs lack the rigor usually required for good mathematical proofs.

*The author is partly supported by the projects NKBRPC-2002cb312200, NSFC-60493200, NSFC-60721061, NSFC-60573007, NSFC-90718041, and NSFC-60736017.

Received October 2005; accepted November 2008. Communicated by Ralph-Johan Back.

This is evident in the way in which new concepts were formed, definitions were given and arguments were conducted. In the extreme case, argument was given by diagrams representing execution of tasks, e.g., in the "critical instance" theorem for RM, one can find reasoning like "as shown in Figure" [Liu and Layland 1973,Buttazzo 1997]. Intuitive understanding is surely important, but does not provide the same level of assurance as solid mathematical proofs. Not surprisingly, mistakes sometimes occur and what is indeed true remains uncertain.

Recently, some attempts have been made towards making the work in this area more rigorous. Devillers and Goossens [Devillers and Goossens 2000,Goossens 1999] found several errors and incomplete places in the proofs of Liu and Layland, including the "critical instance" theorem. Goossens [Goossens 1999] studied various scheduling algorithms in details and a considerable amount of effort was put on proofs. Although the work by Devillers and Goossens was a lot more rigorous, the level of formality was still not very high. For example, just as in the earlier work, some definitions were given in natural languages, instead of more precise mathematical terms. Reasoning was in natural language and soundness of some deduction steps is not immediately clear.

In the meantime, completely formal proofs for scheduling theorems have been investigated and we have participated in this effort. In our approach, a mathematical logic, Duration Calculus (DC) [Zhou *et al.* 1991] has been used. DC is a real time extension of the Interval Temporal Logic (ITL) [Moszkowski 1985]. It has been widely applied to specification and verification of various real time systems. The first application of DC to scheduling was due to Zheng and Zhou [Zheng and Zhou 1994], and they proved Liu and Layland's theorem on the EDF. In [Dong *et al.* 1999], Liu and Layland's theorem on the RM scheduler was proven. The same mistake in Liu and Layland's paper reported by Devillers and Goossens [Devillers and Goossens 2000] was independently discovered and corrected. In [Zhan 2000] another proof of the theorem on EDF was given in DC, following the original proof idea of Liu and Layland.

The approach of using DC is as follows.

- Variables are introduced to model the states of the system. For example, Run_i is a Boolean variable of time, and its value is true at time t if and only if task τ_i is running at t.
- The assumptions, such as the tasks are periodic and they share one processor, is specified by a DC formula *Ass*.
- The concerned scheduling policy, is represented as a DC formula *Sch*.
- The requirement, in this case, that all task instances should be completed by their deadlines is modelled also as a DC formula *Req*.

  Thus, that a schedulability condition *Cond* is sufficient is formally expressed as the following logical implication in DC: $Ass \wedge Sch \wedge Cond \Rightarrow Req$.
- The proof system of DC is used to prove the theorems.

This paper aims to improve the work in this area and give a summary of it. The remainder of this paper is organized as: Section 2 gives preliminaries of this paper, including a brief review of the basic definitions and results of real-time scheduling and a short introduction to DC. In Section 3, we formally describe scheduling problems in DC, that is, we specify in DC the assumptions under which the tasks are executed, the underlining algorithms and timing requirements. Section 4 is devoted to fixed priority schedulers. In particular, we prove Liu and Layland's schedulability theorem for RM. This is based on [Dong *et al.* 1999], but we have changed the style of the proof to make it both more readable and more rigorous. Liu and Layland's schedulability theorem for EDF is proved in Section 5. The proof is based on [Zhan 2000], which has not only been improved in style but also considerably in contents. We end this paper with discussions of related work and conclusions. A number of technical lemmas concerning RM are included in the appendix where the mistake in Liu and Layland's paper is reported and corrected.

2. Preliminaries

In this section, we introduce some basic notions and results that will be used later, including a review of the basic concepts of real-time scheduling and an introduction to DC.

2.1 Scheduling real-time tasks on a uniprocessor

In this paper, we study the basic scheduling problem with the following assumptions:

- Tasks are periodic and they start at the same time;
- There is only one processor, and therefore the execution of two tasks is mutually exclusive;
- The deadline of each task is equal to its period;
- The tasks are independent in that requests of a task do not depend on the execution of other tasks;
- Execution time, i.e., the time which is taken by a processor to execute the task without interruption, is constant for a task.

These assumptions allow the complete characterization of a task by two attributes: its request period and its execution time.

A scheduling algorithm is said to be *preemptive* and *priority driven* if whenever there is a request for a task with a higher priority than the task currently being executed, the running task is immediately interrupted and the newly requested task is started. A scheduling algorithm is said to be *static* if priorities are assigned to tasks once and kept unchanged, and a scheduling algorithm is called to be *dynamic* if priorities of tasks may change during the execution.

Rate Monotonic Scheduler statically assigns priorities to tasks according to their request rates, i.e., tasks with higher request rates (shorter periods) have higher priorities. The most well-known dynamic scheduling algorithm is Earliest Deadline First (EDF). In EDF, priorities are assigned to tasks according to the deadlines of their current requests. A task is assigned the highest priority if the deadline of its current request is the nearest, and is assigned the lowest priority if the deadline of its current request is the furthest.

Given a set of tasks and a scheduling algorithm, the task set is *schedulable* by the algorithm if the requested execution time of each task instance is fulfilled before the deadline. Unless stated otherwise, throughout this paper we shall use $\tau_1, \tau_2, \cdots, \tau_m$ to denote m tasks, $C_1, C_2, \cdots, C_m$ their execution times and $T_1, T_2, \cdots, T_m$ their periods. The processor utilisation factor of the tasks is defined as $\sum_{i=1}^{m} C_i/T_i$, and is used in schedulability analysis.

NECESSARY CONDITION: *If a set of m tasks is schedulable by any scheduling algorithm, then processor utilisation factor is less than or equal to 1.*

Liu and Layland [Liu and Layland 1973] studied sufficient schedulability conditions for RM and EDF:

SUFFICIENT CONDITION FOR RM: *A set of m tasks is schedulable by RM, if the processor utilisation factor is less than or equal to* $m(2^{\frac{1}{m}} - 1)$.

SUFFICIENT CONDITION FOR EDF: *A set of m tasks is schedulable by EDF if the processor utilisation factor is less than or equal to 1.*

2.2 Duration calculus

In this subsection, we give a brief review of DC. DC, proposed by Zhou, Hoare and Ravn [Zhou *et al.* 1991], is an extension of real arithmetics and ITL [Moszkowski 1985]. A more comprehensive introduction to DC can be found in [Hansen and Zhou 1997,Zhou and Hansen 2004].

In this paper, we use $\mathbb{N}$ to denote the set of natural numbers and $\mathbb{R}$ the set of reals. DC contains the following sets of symbols:

- A set of global variables $GVar = \{x, y, \ldots\}$, and the meaning of a global variable is independent of time;
- A set of state variables $SVar = \{P, Q, \ldots\}$ that are used to model the behavior of systems;
- A set of temporal propositional letters $PLetter = \{X, Y, \ldots\}$;
- A set of global function symbols $FSymb = \{f, g, \ldots\}$;
- A set of global relation symbols $RSymb = \{G, H, \ldots\}$.

In DC, only functions and relations of real arithmetic are concerned, and therefore a DC model contains

- a total function $\underline{f_i^n} \in \mathbb{R}^n \rightarrow \mathbb{R}$ is associated with each n-ary function symbols $f_i^n \in FSymb$, and
- a total function $\underline{G_i^n} \in \mathbb{R} \rightarrow \{tt, ff\}$ is associated with each n-ary relation symbol $G_i^n \in RSymb$.

Here, tt and ff represent Boolean values *true* and *false* respectively.

The meaning of global variables is given by a value assignment,

$$\mathcal{V} \in GVar \rightarrow \text{Values}$$

associating a value with each global variable. Time is represented by the set of non-negative reals, denoted by Time. An interval is a pair of time points, where the beginning time point is no later than the ending point:

$$Intv \widehat{=} \{[b,e] \in \text{Time} \times \text{Time} \mid b \leq e\}.$$

Interpretations of state variables and propositional letters are defined as follows:

$$\mathcal{I} \in SVar \rightarrow \text{Time} \rightarrow \{0,1\},$$
$$\mathcal{J} \in PLetters \rightarrow Intv \rightarrow \{0,1\}.$$

State variables are interpreted as functions from Time to Boolean values (denoted by 0 and 1). All state variables are assumed to have *finite variability*, which means that each state variable can only change its value a finite many times over any (finite) interval. A model is a quadruple $(\mathcal{I}, \mathcal{J}, \mathcal{V}, [b,e])$.

A Boolean state expression S is constructed from (Boolean) state variables with Boolean connectives and its duration in a model $(\mathcal{I}, \mathcal{J}, \mathcal{V}, [c,d])$ is defined as

$$(\textstyle\int S)(\mathcal{I}, \mathcal{J}, \mathcal{V}, [b,e]) \widehat{=} \int_b^e (S)(\mathcal{I}, \mathcal{V})(t)dt,$$

where $(S)(\mathcal{I}, \mathcal{V})(t)$ denotes the value of S at time t under state interpretation $\mathcal{I}$ and valuation $\mathcal{V}$. The length ℓ of an interval is defined as $\ell \widehat{=} \int 1$ and it is easy to prove that $(\ell)(\mathcal{I}, \mathcal{J}, \mathcal{V}, [c,d]) = d - c$. Primitive formulae of DC are either temporal propositionals letter or those constructed from terms using comparison operators in arithmetics, such as <, = etc. DC formulae are contructed from prmitive formulae by Boolean connectives and modality operators. A symbol is called *rigid* if its meaning is independent of time and intervals; otherwise called *flexible*. Global variables, constants, function symbols and relation symbols are rigid, whereas state variables and temporal propositional letters are flexible. A term or formula is called *rigid* if it contains no flexible symbols; otherwise called *flexible*. Boolean state expression S holds almost everywhere (i.e., except possibly a finite number of points) over a non-point interval, denoted as $\lceil S \rceil$, is defined as, $\lceil S \rceil \widehat{=} \int S = \ell \wedge \ell > 0$. A point interval is characterised by $\ell = 0$, shortened as $\lceil\rceil$. The modality "chop" of ITL is defined as follows: for any formulae ϕ and ψ,

$$(\mathcal{I}, \mathcal{J}, \mathcal{V}, [b,e]) \models \phi ^\frown \psi \text{ iff there exists } m \text{ such that } b \leq m \leq e \text{ and}$$
$$(\mathcal{I}, \mathcal{J}, \mathcal{V}, [b,m]) \models \phi \text{ and } (\mathcal{I}, \mathcal{J}, \mathcal{V}, [m,e]) \models \psi.$$

The following abbreviations will be used:

$\Diamond\phi$	$\widehat{=}\ tt ^\frown (\phi ^\frown tt)$	reads: "for some sub-interval: ϕ",
$\Box\phi$	$\widehat{=}\ \neg\Diamond(\neg\phi)$	reads: "for all sub-intervals: ϕ",
$\Diamond_p\phi$	$\widehat{=}\ \phi ^\frown tt$	reads: "for some prefix: ϕ",
$\Box_p\phi$	$\widehat{=}\ \neg\Diamond_p(\neg\phi)$	reads: "for all prefixes: ϕ".

As usual, a formula ϕ is valid if for any model $(\mathcal{I}, \mathcal{J}, \mathcal{V}, [b, e])$,

$$(\mathcal{I}, \mathcal{J}, \mathcal{V}, [b, e]) \models \phi.$$

A term θ is said to be *free for* x in ϕ if x does not occur freely in ϕ within the scope of $\exists y$ or $\forall y$, where y is any variable occurring in θ.

The axioms of DC include those of ITL which are taken from the paper by Dutertre [Dutertre 1995]:

ITL1: $\ell \geq 0$

ITL2: $((\phi^\frown\psi) \wedge \neg(\phi^\frown\varphi)) \Rightarrow (\phi^\frown(\psi \wedge \neg\varphi))$
$((\phi^\frown\psi) \wedge \neg(\varphi^\frown\psi)) \Rightarrow ((\phi \wedge \neg\varphi)^\frown\psi)$

ITL3: $((\phi^\frown\psi)^\frown\varphi) \;\Leftrightarrow\; (\phi^\frown(\psi^\frown\varphi))$

ITL4: $(\phi^\frown\psi) \;\Rightarrow\; \phi$, if ϕ is a rigid formula
$(\phi^\frown\psi) \;\Rightarrow\; \psi$, if ψ is a rigid formula

ITL5: $(\exists x.\phi^\frown\psi) \;\Rightarrow\; \exists x.(\phi^\frown\psi)$, if x is not free in ψ
$(\phi^\frown\exists x.\psi) \;\Rightarrow\; \exists x.(\phi^\frown\psi)$, if x is not free in ϕ

ITL6: $((\ell = x)^\frown\phi) \;\Rightarrow\; \neg((\ell = x)^\frown\neg\phi)$
$(\phi^\frown(\ell = x)) \;\Rightarrow\; \neg(\neg\phi^\frown(\ell = x))$

ITL7: $(x \geq 0 \wedge y \geq 0) \Rightarrow ((\ell = x + y) \Leftrightarrow ((\ell = x)^\frown(\ell = y)))$

ITL8: $\phi \Rightarrow (\phi^\frown(\ell = 0))$
$\phi \Rightarrow ((\ell = 0)^\frown\phi)$

and the following axioms about durations:

DCA1: $\int 0 = 0$

DCA2: $\int 1 = \ell$

DCA3: $\int S \geq 0$

DCA4: $\int S_1 + \int S_2 = \int (S_1 \vee S_2) + \int (S_1 \wedge S_2)$

DCA5: $((\int S = x)^\frown(\int S = y)) \Rightarrow (\int S = x + y)$

DCA6: $\int S_1 = \int S_2$, provided $S_1 \Leftrightarrow S_2$ holds in propositional logic

The inference rules of DC include:

MP: if ϕ and $\phi \Rightarrow \psi$ then ψ (modus ponens)

G: if ϕ then $\forall x.\phi$ (generalization)

Q: $\forall x.\phi(x) \Rightarrow \phi(\theta)$
if either θ is free for x in $\phi(x)$ and θ is rigid
or θ is free for x in $\phi(x)$ and $\phi(x)$ is chop free.

N: if ϕ then $\neg(\neg\phi^\frown\psi)$
if ϕ then $\neg(\psi^\frown\neg\phi)$

M: if $\phi \Rightarrow \psi$ then $(\phi^\frown\varphi) \Rightarrow (\psi^\frown\varphi)$
if $\phi \Rightarrow \psi$ then $(\varphi^\frown\phi) \Rightarrow (\varphi^\frown\psi)$

IR1: Let $H(X)$ be a formula possibly containing the propositional letter X, and $S_1, \ldots, S_m$ be m state expressions with $S_1 \vee \ldots \vee S_m = 1$.

If $H(\lceil\rceil)$ and $H(X) \Rightarrow H(X \vee (X^\frown\lceil S_1\rceil) \vee \ldots \vee (X^\frown\lceil S_m\rceil))$, then $H(tt)$,

where $H(\phi)$ denotes the formula obtained from $H(X)$ by replacing X in H with ϕ.

IR2: Let $H(X)$ be a formula possibly containing the propositional letter X, and $S_1, \ldots, S_m$ be m state expressions with $S_1 \vee \ldots \vee S_m = 1$.

If $H(\lceil\rceil)$ and $H(X) \Rightarrow H(X \vee (\lceil S_1\rceil^\frown X) \vee \ldots \vee (\lceil S_m\rceil^\frown X))$, then $H(tt)$.

The above proof system is sound and relative complete in the sense that all valid formulae of ITL are assumed to be provable [Hansen and Zhou 1997, Zhou and Hansen 2004].

Using the proof system, we can easily prove the following theorems which will be used later. Below, variables x and y are assumed to be non-negative:

DC1 $(tt^\frown tt) \Leftrightarrow tt$
DC2-1 $(\phi^\frown(\psi \vee \varphi)) \Leftrightarrow ((\phi^\frown\psi) \vee (\phi^\frown\varphi))$
DC2-2 $((\phi \vee \psi)^\frown\varphi) \Leftrightarrow ((\phi^\frown\varphi) \vee (\psi^\frown\varphi))$
DC3 $(\int S \geq x) \Leftrightarrow ((\int S = x)^\frown tt)$
DC4-1 $(\phi^\frown(\ell = 0)) \Rightarrow \phi$
DC4-2 $((\ell = 0)^\frown\phi) \Rightarrow \phi$
DC5-1 $((\Box\phi) \wedge (\psi^\frown\varphi)) \Rightarrow ((\phi \wedge \psi)^\frown\varphi)$
DC5-2 $((\Box\phi) \wedge (\psi^\frown\varphi)) \Rightarrow (\psi^\frown(\phi \wedge \varphi))$
DC6 $((\Box_p\phi) \wedge (\psi^\frown\varphi)) \Rightarrow ((\phi \wedge \psi)^\frown\varphi)$
DC7-1 $(\Box\phi) \wedge (\Box\psi) \Leftrightarrow (\Box(\phi \wedge \psi))$
DC7-2 $(\Box_p\phi) \wedge (\Box_p\psi) \Leftrightarrow (\Box_p(\phi \wedge \psi))$
DC8-1 $(\Box\phi) \Rightarrow \phi$
DC8-2 $(\Box_p\phi) \Rightarrow \phi$
DC9 $((\Diamond_p\phi) \wedge (\Box\psi)) \Rightarrow (\Diamond_p(\phi \wedge \psi))$
DC10 $\exists x.(\ell = x)$
DC11 $(\lceil S_1\rceil \wedge \lceil S_2\rceil) \Leftrightarrow \lceil S_1 \wedge S_2\rceil$
DC12 $\lceil\neg S\rceil \Rightarrow (\int S = 0)$
DC13 $(\ell = 0) \Rightarrow (\int S = 0$
DC14-1 $\lceil S\rceil \Leftrightarrow (\lceil S\rceil^\frown\lceil S\rceil)$
DC14-2 $((\phi^\frown\psi) \wedge (\lceil S\rceil \vee \lceil\rceil)) \Rightarrow ((\phi \wedge (\lceil S\rceil \vee \lceil\rceil))^\frown(\psi \wedge (\lceil S\rceil \vee \lceil\rceil)))$
DC15-1 $((\phi^\frown\psi) \wedge (\int S \leq x)) \Rightarrow ((\phi \wedge (\int S \leq x))^\frown(\psi \wedge (\int S \leq x)))$

DC15-2	$((\phi^\frown\psi) \wedge (\int S < x)) \Rightarrow ((\phi \wedge (\int S < x))^\frown(\psi \wedge (\int S < x)))$
DC16-1	$((\phi_1 \wedge (\ell = x))^\frown\psi_1 \wedge (\phi_2 \wedge (\ell = x))^\frown\psi_2) \Rightarrow$ $((\phi_1 \wedge \phi_2 \wedge (\ell = x))^\frown(\psi_1 \wedge \psi_2))$
DC16-2	$(\phi_1^\frown(\psi_1 \wedge (\ell = x)) \wedge \phi_2^\frown(\psi_2 \wedge (\ell = x))) \Rightarrow$ $((\phi_1 \wedge \phi_2)^\frown(\psi_1 \wedge \psi_2 \wedge (\ell = x)))$
DC17	$((\int S < x) \wedge (\ell \geq y)) \Leftrightarrow ((\int S < x)^\frown(\ell = y))$
DC18	$((\int S \leq x) \wedge ((\int S \geq y)^\frown tt)) \Rightarrow (x \geq y)$
DC19	$((\int S \geq x)^\frown(\int S \geq y)) \Rightarrow (\int S \geq x + y)$
DC20	$(\int S \neq \ell) \Rightarrow (tt^\frown\lceil\neg S\rceil^\frown(\int S = \ell))$ and $(\int S \neq 0) \Rightarrow$ $(tt^\frown\lceil S\rceil^\frown(\int S = 0))$
DC21	$((\sum_{i=1}^m \int S_i \leq \ell)^\frown(\sum_{i=1}^m \int S_i \leq \ell)) \Rightarrow (\sum_{i=1}^m \int S_i \leq \ell)$
DC22	$((\int S \leq \lceil \ell/T \rceil C) \wedge ((\int S = \lceil \ell/T \rceil C)^\frown(\ell = x))) \Rightarrow$ $(tt^\frown((\ell = x) \wedge (\int S \leq \lceil \ell/T \rceil C)))$
DC23	$((\int S < \lfloor \ell/T \rfloor C) \wedge ((\int S = \lceil \ell/T \rceil C)^\frown(\ell = x))) \Rightarrow$ $(tt^\frown((\ell = x) \wedge (\int S < \lfloor \ell/T \rfloor C)))$
DC24	$((\int S \leq \lfloor \ell/T \rfloor C) \wedge ((\int S = \lceil \ell/T \rceil C)^\frown(\ell = x))) \Rightarrow$ $(tt^\frown((\ell = x) \wedge (\int S \leq \lfloor \ell/T \rfloor C)$

DC22, DC23 and DC24 hold due to the following properties of real numbers:

$$\lceil a/c \rceil + \lceil b/c \rceil \geq \lceil (a+b)/c \rceil \quad \text{and} \quad \lceil a/c \rceil + \lfloor b/c \rfloor \geq \lfloor (a+b)/c \rfloor,$$

where $a, b \geq 0$, $c > 0$, $\lceil a/c \rceil$ and $\lfloor a/c \rfloor$ denote respectively the smallest integer greater than or equal to a/c and the largest integer less than or equal to a/c.

The following rules can be derived:

DC25	if $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow \psi$ is a theorem, where each ϕ_i starts with $\Box$, then $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow (\Box\psi)$ is also a theorem,
DC26	if $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow \psi$ is a theorem, where each ϕ_i starts with $\Box$ or $\Box_p$then $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow (\Box_p\psi)$ is also a theorem.

As special cases, if ψ is a theorem, then $\Box\psi$ and $\Box_p\psi$ are also theorems. [1]

2.3 Proof style

Dijkstra and Scholten introduced calculational proof [Dijkstra and Scholten 1990] as a way of writing practical proofs. In the calculational style, a typical proof of a scheduling theorem in this paper is of the form exemplified in Fig. 1.

In the proof, many of the subformulae, such as Q_1 and P_4 are repeated many times. In a complex proof, the subformulae may be quite long, so the proof will take a lot of space. Moreover, the subformula that is being transformed in one step is mixed with subformulae that are not changed, and therefore the readability

[1] This is not the same as saying $(\psi \Rightarrow \Box\psi)$ and $(\psi \Rightarrow \Box_p\psi)$ are theorems. They are in fact not.

$$
\begin{array}{lllr}
 & P_1 \wedge P_2 \wedge P_3 \wedge P_4 & & (1) \\
\Rightarrow & Q_1 \wedge Q_2 \wedge P_2 \wedge P_3 \wedge P_4 & \{\text{hints for } P_1 \Rightarrow Q_1 \wedge Q_2\} & (2) \\
\Rightarrow & Q_1 \wedge (\exists x.Q_3) \wedge P_3 \wedge P_4 & \{\text{hints for } Q_2 \wedge P_2 \Rightarrow (\exists x.Q_3)\} & (3) \\
\Rightarrow & Q_1 \wedge (\exists x.(Q_3 \wedge Q_4)) \wedge P_3 \wedge P_4 & \{\text{hints for } Q_3 \wedge P_3 \Rightarrow Q_4\} & (4) \\
\Rightarrow & Q_1 \wedge (\exists x.(Q_4 \wedge Q_5)) \wedge P_4 & \{\text{hints for } Q_3 \wedge P_3 \Rightarrow Q_5\} & (5) \\
\Rightarrow & Q_1 \wedge (\exists x.(Q_4 \wedge Q_5 \wedge (R_1 \vee R_2))) \wedge P_4 & \{\ R_1 \vee R_2 \text{ is a tautology }\} & (6) \\
\Rightarrow & Q_1 \wedge (\exists x.((Q_4 \wedge R_1) \vee (Q_5 \wedge R_2))) \wedge P_4 & \{(6)\} & (7) \\
\Rightarrow & Q_1 \wedge (\exists x.(Q_6 \vee (Q_5 \wedge R_2))) \wedge P_4 & \{\text{hints for } Q_4 \wedge R_1 \wedge P_4 \Rightarrow Q_6\} & (8) \\
\Rightarrow & Q_1 \wedge (\exists x.(Q_6 \vee Q_6)) & \{\text{hints for } Q_5 \wedge R_2 \wedge P_4 \Rightarrow Q_6\} & (9) \\
\Rightarrow & Q_1 \wedge (\exists x.Q_6) & \{(9)\} & (10) \\
\Rightarrow & Q_7 & \{\text{hints for } Q_1 \wedge (\exists x.Q_6) \Rightarrow Q_7\} & (11)
\end{array}
$$

Fig. 1: A typical proof of a scheduling theorem in the calculational style.

is poor. In fact, during the process of a proof, one would most likely only wish to write down the subformula that is being transformed. In [Back *et al.* 1997], Back, Grundy and von Wright proposed a way to structure the calculational proof in which only the subformula that is being transformed is written. We adopted a similar style in our previous work [Dong *et al.* 1999,Zhan 2000], and in particular, we used labels to refer to subformulae. However, our style did not express the relation between the formulae explicitly. In this paper, we adopt the idea from [Back *et al.* 1997], but continue to use labels to refer to subformulae as compared to repeating the formulae to simplify the presentation. As an example, the previous proof template will be presented in our new style in Fig 2.

3. General Scheduler Assumptions

In this section, we specify the assumptions that hold for schedulers in general, and deduce a number of basic properties from these assumptions.

3.1 State variables

Two state variables Run_i and Std_i are introduced for each task τ_i. The intention is that Run_i has the value 1 at time t if and only if τ_i is running on the processor at the time point, and Std_i has the value 1 if and only if τ_i still needs processing time. The accumulated run time of task τ_i on an interval is given by $\int \mathrm{Run}_i$. A task is running at time t only if it has a standing request at t and this holds for every task and each time point of every interval. We therefore have the following assumption:

$$A_1 \mathrel{\widehat{=}} \bigwedge_{i=1}^{m} \Box (\lceil \mathrm{Run}_i \rceil \Rightarrow \lceil \mathrm{Std}_i \rceil).$$

$$
\begin{array}{llrr}
 & P_1 \land P_2 \land P_3 \land P_4 & & (1) \\
\Rightarrow & P_1 & \{(1)\} & (2) \\
\Rightarrow & P_2 & \{(1)\} & (3) \\
\Rightarrow & P_3 & \{(1)\} & (4) \\
\Rightarrow & P_4 & \{(1)\} & (5) \\
\Rightarrow & Q_1 \land Q_2 & \{(2) \text{ and hints for } P_1 \Rightarrow Q_1 \land Q_2\} & (6) \\
\Rightarrow & Q_1 & \{(6)\} & (7) \\
\Rightarrow & Q_2 & \{(6)\} & (8) \\
\Rightarrow & \exists x & /* \text{ begin scope } \exists\, x\ */ & \\
 & \quad Q_3 & \{(3), (8), \text{ and hints for } Q_2 \land P_2 \Rightarrow (\exists x.Q_3)\} & (9) \\
\Rightarrow & \quad Q_4 & \{(4), (9), \text{ and hints for } Q_3 \land P_3 \Rightarrow Q_4\} & (10) \\
\Rightarrow & \quad Q_5 & \{(4), (9), \text{ and hints for } Q_3 \land P_3 \Rightarrow Q_5\} & (11) \\
\Rightarrow & \quad R_1 \lor R_2 & \{R_1 \lor R_2 \text{ is a tautology}\} & (12) \\
 & \qquad \text{case 1: } R_1 & /* \text{ case split on (12) } */ & (13) \\
 & \qquad\quad \Rightarrow\ Q_6 & \{(5), (10), (13), \text{ and hints for } Q_4 \land R_1 \land P_4 \Rightarrow Q_6\} & (14) \\
 & \qquad \text{case 2: } R_2 & /* \text{ case split on (12) } */ & (15) \\
 & \qquad\quad \Rightarrow\ Q_6 & \{(5), (11), (15), \text{ and hints for } Q_5 \land R_2 \land P_4 \Rightarrow Q_6\} & (16) \\
\Rightarrow & \quad Q_6 & /* \text{ combine cases 1 and 2 } */ & (17) \\
\Rightarrow & Q_7 & \{(7), (17), \text{ and hints for } (Q_1 \land \exists x.Q_6) \Rightarrow Q_7\} & (18)
\end{array}
$$

Fig. 2: A proof of a scheduling theorem in the style with labels to refer to subformulae.

3.2 Mutual exclusion

Since there is only one processor, if one task is running, then any other task cannot be running:

$$A_2 \mathrel{\widehat{=}} \bigwedge_{i=1}^{m} \Box\,(\lceil \mathrm{Run}_i \rceil \Rightarrow \bigwedge_{j \neq i} \lceil \neg \mathrm{Run}_j \rceil).$$

In other words, there does not exist an interval such that two tasks are running in parallel.

LEMMA 1. *For any i and j, $i \neq j$, $A_2 \;\Rightarrow\; \Box(\neg \lceil Run_i \land Run_j \rceil)$.*

PROOF. From DC25, we only need to prove $A_2 \;\Rightarrow\; \neg\lceil \mathrm{Run}_i \land \mathrm{Run}_j \rceil$, and this follows immediately from DC11 and propositional logic. □

Consider a subset of tasks $\tau_{i_1}, \ldots, \tau_{i_n}$ ($n \leq m$). The single processor assumption implies that the sum of the running time of these tasks is equal to $\int \vee_{j=1}^{n} \mathrm{Run}_{i_j}$.

LEMMA 2. $A_2 \;\Rightarrow\; \Box\,(\sum_{j=1}^{n} \int Run_{i_j} = \int \bigvee_{j=1}^{n} Run_{i_j})$.

PROOF. By DC25 it is enough to prove $A_2 \Rightarrow (\sum_{j=1}^{n} \int \mathrm{Run}_{i_j} = \int \bigvee_{j=1}^{n} \mathrm{Run}_{i_j})$. This is shown as follows:

$$\begin{array}{llr}
 & A_2 & (1)\\
\Rightarrow & \sum_{j=1}^{n-1} \int(\mathrm{Run}_{i_j} \wedge \mathrm{Run}_{i_n}) = 0 \qquad \{\text{Lemma 1 and DC20}\} & (2)\\
\Rightarrow & \int \bigvee_{j=1}^{n-1}(\mathrm{Run}_{i_j} \wedge \mathrm{Run}_{i_n}) \leq \sum_{j=1}^{n-1} \int(\mathrm{Run}_{i_j} \wedge \mathrm{Run}_{i_n}) \qquad \{\text{DCA4}\} & (3)\\
\Rightarrow & \int \bigvee_{j=1}^{n-1}(\mathrm{Run}_{i_j} \wedge \mathrm{Run}_{i_n}) = 0 \qquad \{(2), (3), \text{ and DCA3}\} & (4)\\
\Rightarrow & \int \bigvee_{j=1}^{n} \mathrm{Run}_{i_j} = (\int \bigvee_{j=1}^{n-1} \mathrm{Run}_{i_j}) + \int \mathrm{Run}_{i_n} - \int((\bigvee_{j=1}^{n-1} \mathrm{Run}_{i_j}) \wedge \mathrm{Run}_{i_n}) \qquad \{\text{DCA4}\} & (5)\\
\Rightarrow & \int((\bigvee_{j=1}^{n-1} \mathrm{Run}_{i_j}) \wedge \mathrm{Run}_{i_n}) = \int \bigvee_{j=1}^{n-1}(\mathrm{Run}_{i_j} \wedge \mathrm{Run}_{i_n}) \qquad \{\text{DCA6}\} & (6)\\
\Rightarrow & \int \bigvee_{j=1}^{n} \mathrm{Run}_{i_j} = (\int \bigvee_{j=1}^{n-1} \mathrm{Run}_{i_j}) + \int \mathrm{Run}_{i_n} \qquad \{(4), (5), \text{and } (6)\} & (7)
\end{array}$$

We can repeat the above steps until the lemma is proven. □

Two obvious facts can be easily derived from this lemma: the sum of the running time of a subset of tasks is less than or equal to the length of the interval, and the equality holds if the processor is occupied completely by the tasks over the interval.

COROLLARY 1 OF LEMMA 2. $A_2 \Rightarrow \Box(\sum_{j=1}^{n} \int \mathrm{Run}_{i_j} \leq \ell)$.

COROLLARY 2 OF LEMMA 2. $A_2 \Rightarrow \Box(\lceil \bigvee_{j=1}^{n} \mathrm{Run}_{i_j} \rceil \Rightarrow (\sum_{j=1}^{n} \int \mathrm{Run}_{i_j} = \ell))$.

3.3 No overhead

We assume that if there are some tasks with standing requests, then one of the tasks must be running:

$$A_3 \mathrel{\widehat{=}} \Box(\lceil \bigvee_{i=1}^{m} \mathrm{Std}_i \rceil \Rightarrow \lceil \bigvee_{i=1}^{m} \mathrm{Run}_i \rceil).$$

3.4 Execution time bound

Task τ_i requires C_i units execution time for each of its period T_i, and in the interval of length ℓ starting from 0, there are at most $\lceil \ell/T_i \rceil$ requests for τ_i. Consequently, the accumulated running time of the task will not exceed $\lceil \ell/T_i \rceil C_i$:

$$A_4 \mathrel{\widehat{=}} \bigwedge_{i=1}^{m} \Box_p (\int \mathrm{Run}_i \leq \lceil \ell/T_i \rceil C_i).$$

Let $mult_i \mathrel{\widehat{=}} \exists k \in \mathbb{N}.(k \cdot T_i = \ell)$.

Thus, $mult_i$ holds for intervals whose lengthes are multiples of period T_i. If the request of task τ_i is still not fulfilled at a time point and it is not a period point, then the accumulated running time up to that moment cannot be equal to (should be less than) $\lceil \ell/T_i \rceil C_i$:

$$A_5 \mathrel{\widehat{=}} \bigwedge_{i=1}^{m} \Box_p \neg(((\neg mult_i) \wedge (\int \mathrm{Run}_i = \lceil \ell/T_i \rceil C_i))^\frown \lceil \mathrm{Std}_i \rceil).$$

If a task's request is not standing, then the execution time requirement has been reached:

$$A_6 \mathrel{\widehat{=}} \bigwedge_{i=1}^{m} \Box_p ((tt^\frown \lceil \neg \mathrm{Std}_i \rceil) \Rightarrow (\int \mathrm{Run}_i = \lceil \ell/T_i \rceil C_i)).$$

Denote the conjunction of all the assumptions A_1 to A_6 by A,

$$A \mathrel{\widehat{=}} A_1 \wedge A_2 \wedge A_3 \wedge A_4 \wedge A_5 \wedge A_6.$$

It represents the general assumptions about behaviours of the scheduling algorithms that we study.

3.5 Requirement

In any interval starting from 0 and of length ℓ, a task should have been granted at least $\lfloor \ell/T_i \rfloor C_i$ execution time. The requirement for task τ_i is

$$Req_i \mathrel{\widehat{=}} \Box_p(\textstyle\int \mathrm{Run}_i \geq \lfloor \ell/T_i \rfloor C_i).$$

This must hold for every task:

$$Req \mathrel{\widehat{=}} \textstyle\bigwedge_{i=1}^{m} Req_i.$$

3.6 Necessary condition

If the tasks are schedulable, then the running time requirement is satisfied over any interval, and therefore in particular over $[0, T]$, where $T = T_1 T_2 \cdots T_n$ is the product of the periods of all the tasks. The necessity of the condition is implied by the following theorem.

THEOREM 1. $(A \wedge Req \wedge (\ell = T)) \Rightarrow (\sum_{i=1}^{m} C_i/T_i \leq 1)$.

PROOF.

$$\begin{array}{llrr}
 & A \wedge Req \wedge (\ell = T) & & (1) \\
\Rightarrow & A & \{(1)\} & (2) \\
\Rightarrow & Req & \{(1)\} & (3) \\
\Rightarrow & \ell = T & \{(1)\} & (4) \\
\Rightarrow & \bigwedge_{1\leq i\leq m}(\int \mathrm{Run}_i \geq \lfloor \ell/T_i \rfloor C_i) & \{\text{definition of } Req\text{, DC7, DC8, and (3)}\} & (5) \\
\Rightarrow & \bigwedge_{1\leq i\leq m}(\int \mathrm{Run}_i \geq (T/T_i) C_i) & \{\text{(4) and (5)}\} & (6) \\
\Rightarrow & \sum_{i=1}^{m} \int \mathrm{Run}_i \geq \sum_{i=1}^{m} (T/T_i) C_i & \{(6)\} & (7) \\
\Rightarrow & \ell \geq \sum_{i=1}^{m} (T/T_i) C_i & \{\text{Corollary 1 of Lemma 2, and (7)}\} & (8) \\
\Rightarrow & \sum_{i=1}^{m} C_i/T_i \leq 1 & \{\text{(4) and (8)}\} & (9)
\end{array}$$

□

4. Static Scheduler

Without loss of generality, we assume priorities are in decreasing order from τ_1 to τ_m. For $i < j$, task τ_j cannot be running if task τ_i has a standing request, i.e.

$$Sch_S \mathrel{\widehat{=}} \textstyle\bigwedge_{1\leq i\leq m} \Box(\lceil \mathrm{Std}_i \rceil \Rightarrow \bigwedge_{i<j\leq m} \lceil \neg \mathrm{Run}_j \rceil).$$

4.1 General properties

For any task τ_k, if its execution time is not fulfilled at t within its first period, then the interval $[0, t]$ is completely occupied by $\tau_1, \ldots, \tau_k$.

LEMMA 3. *For any* $1 \le k \le m$,

$$A_3 \wedge A_5 \wedge A_6 \wedge Sch_S \;\Rightarrow\; \Box_p\,(\,((\ell \le T_k) \wedge (tt^\frown\lceil Std_k\rceil)) \;\Rightarrow\; \lceil\bigvee_{i=1}^{k} Run_i\rceil\,).$$

PROOF. According to DC26 and propositional logic, it is enough to prove

$$(A_3 \wedge A_5 \wedge A_6 \wedge Sch_S \wedge (\ell \le T_k) \wedge (tt^\frown\lceil \mathrm{Std}_k\rceil)) \;\Rightarrow\; \lceil\bigvee_{i=1}^{k}\mathrm{Run}_i\rceil.$$

This is shown as follows:

$$\begin{array}{llr}
\quad A_3 \wedge A_5 \wedge A_6 \wedge Sch_S \wedge (\ell \le T_k) \wedge (tt^\frown\lceil \mathrm{Std}_k\rceil) & & (1)\\
\Rightarrow\; A_3 \wedge A_5 \wedge A_6 & \{(1)\} & (2)\\
\Rightarrow\; \ell \le T_k & \{(1)\} & (3)\\
\Rightarrow\; tt^\frown\lceil \mathrm{Std}_k\rceil & \{(1)\} & (4)\\
\Rightarrow\; (\int\mathrm{Std}_k = \ell) \vee (\int\mathrm{Std}_k \ne \ell) & \{\text{tautology}\} & (5)\\
\quad\text{case 1: } \int\mathrm{Std}_k = \ell & \{\text{case split on (5)}\} & (6)\\
\quad\quad \Rightarrow\; \ell > 0 & \{(4)\} & (7)\\
\quad\quad \Rightarrow\; \lceil \mathrm{Std}_k\rceil & \{(6) \text{ and } (7)\} & (8)\\
\quad\text{case 2: } \int\mathrm{Std}_k \ne \ell & \{\text{case split on (5)}\} & (9)\\
\quad\quad \Rightarrow\; tt^\frown\lceil\neg\mathrm{Std}_k\rceil^\frown tt & \{(9) \text{ and DC20}\} & (10)\\
\quad\quad \Rightarrow\; tt^\frown\lceil\neg\mathrm{Std}_k\rceil^\frown\lceil\neg\mathrm{Std}_k\rceil^\frown tt & \{(10) \text{ and DC14}\} & (11)\\
\quad\quad \Rightarrow\; tt^\frown\lceil\neg\mathrm{Std}_k\rceil^\frown(\ell > 0) & \{(11)\} & (12)\\
\quad\quad \Rightarrow\; (\int\mathrm{Run}_k = C_k)^\frown(\ell > 0) & \{(3), (12), \text{ and } A_6\} & (13)\\
\quad\quad \Rightarrow\; ((\ell < T_k) \wedge (\int\mathrm{Run}_k = C_k))^\frown\lceil\mathrm{Std}_k\rceil & \{(3), (4), \text{ and } (13)\} & (14)\\
\quad\quad \Rightarrow\; ((\neg mult_k) \wedge (\ell < T_k) \wedge (\int\mathrm{Run}_k = C_k))^\frown\lceil\mathrm{Std}_k\rceil & \{(14)\} & (15)\\
\quad\quad \Rightarrow\; ff & \{(15) \text{ and } A_5\} & (16)\\
\Rightarrow\; \lceil\mathrm{Std}_k\rceil & \{\text{combine cases 1 and 2}\} & (17)\\
\Rightarrow\; \lceil\bigvee_{i=1}^{m}\mathrm{Run}_i\rceil & \{(17) \text{ and } A_3\} & (18)\\
\Rightarrow\; \wedge_{i=k+1}^{m}\lceil\neg\mathrm{Run}_i\rceil & \{(17) \text{ and } Sch_S\} & (19)\\
\Rightarrow\; \lceil\bigvee_{i=1}^{k}\mathrm{Run}_i\rceil & \{(18), (19), \text{ and DC11}\} & (20)
\end{array}$$

□

We next prove a result similar to Liu and Layland's critical instance theorem: a task can be scheduled successfully by the static scheduler if it can be done so in its first period. In another word, the task set is schedulable by the static scheduler if the tasks can be scheduled successfully in the first longest period.

THEOREM 2. *For any* $1 \le i \le m$,

$$(A \wedge Sch_S) \;\Rightarrow\; (Req_i \Leftrightarrow \Box_p((\ell \ge T_i) \Rightarrow (((\ell = T_i) \wedge Req_i)^\frown tt))).$$

PROOF. For any $1 \leq i \leq m$, it is obvious that

$$Req_i \Rightarrow \Box_p((\ell \geq T_i) \Rightarrow (((\ell = T_i) \wedge Req_i)^\frown tt)),$$

therefore we only need to prove

$$(A \wedge Sch_S \wedge \Box_p((\ell \geq T_i) \Rightarrow (((\ell = T_i) \wedge Req_i)^\frown tt))) \ \Rightarrow \ Req_i,$$

and by DC26 it is reduced to show

$$(A \wedge Sch_S \wedge \Box_p((\ell \geq T_i) \Rightarrow (((\ell = T_i) \wedge Req_i)^\frown tt))) \ \Rightarrow \ (\textstyle\int \mathrm{Run}_i \geq \lfloor \ell/T_i \rfloor C_i).$$

Suppose that there exists a k such that $\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor \cdot C_k$, then we prove this leads to contradiction:

$$
\begin{array}{llrr}
 & A \wedge Sch_S \wedge \Box_p((\ell \geq T_k) \Rightarrow (((\ell = T_k) \wedge Req_k)^\frown tt)) \wedge \int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor \cdot C_k & & (1)\\
\Rightarrow & A & \{(1)\} & (2)\\
\Rightarrow & Sch_S & \{(1)\} & (3)\\
\Rightarrow & \Box_p((\ell \geq T_k) \Rightarrow (((\ell = T_k) \wedge Req_k)^\frown tt)) & \{(1)\} & (4)\\
\Rightarrow & \int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k & \{(1)\} & (5)\\
\Rightarrow & (\ell \geq T_k) \Rightarrow (((\ell = T_k) \wedge Req_k)^\frown tt) & \{(4) \text{ and DC8}\} & (6)\\
\Rightarrow & \ell \geq T_k \wedge C_k > 0 & \{(5) \text{ and DCA3}\} & (7)\\
\Rightarrow & ((\ell = T_k) \wedge Req_k)^\frown tt & \{(6) \text{ and } (7)\} & (8)\\
\Rightarrow & ((\ell = T_k) \wedge (\int \mathrm{Run}_k = C_k))^\frown tt & \{(8), (2), \text{ def. of } Req_k, A_4, \text{ and DC8}\} & (9)\\
\Rightarrow & ((\ell = T_k) \wedge (\int \mathrm{Run}_k = C_k) \wedge (tt^\frown \lceil \mathrm{Run}_k \rceil^\frown (\int \mathrm{Run}_k = 0))^\frown tt & & (10)\\
 & & \{(7), (9), \text{ and DC20}\} & \\
\Rightarrow & \exists\, a \in \mathbb{R}. a \geq 0 \wedge & /*\ \text{begin scope } \exists\, a\ */ & \\
 & \quad ((\ell = T_k) \wedge (\int \mathrm{Run}_k = C_k) \wedge (((\ell = a) \wedge (tt^\frown \lceil \mathrm{Run}_k \rceil))^\frown & & \\
 & \quad (\int \mathrm{Run}_k = 0)))^\frown tt & \{(10) \text{ and DC10}\} & (11)\\
\Rightarrow & \quad ((\ell = a) \wedge (\ell \leq T_k) \wedge (tt^\frown \lceil \bigwedge_{i=1}^{k-1} \neg \mathrm{Std}_i \rceil) \wedge \lceil \bigvee_{i=1}^{k} \mathrm{Run}_i \rceil \wedge & & \\
 & \quad (\int \mathrm{Run}_k = C_k))^\frown tt & \{(11),\ (3),\ \text{ITL7, Lemma 3, and DCA5}\} & (12)\\
\Rightarrow & \quad ((\ell = a) \wedge (a \leq T_k) \wedge (\ell > 0) \wedge (\ell = \sum_{i=1}^{k} \int \mathrm{Run}_i) & & \\
 & \qquad \wedge \bigwedge_{i=1}^{k} (\int \mathrm{Run}_i = \lceil \ell/T_i \rceil C_i))^\frown tt & \{(12),\ \text{Corollary 2}, A_6, \text{ and DC1}\} & (13)\\
\Rightarrow & \quad ((a \leq T_k) \wedge (a > 0) \wedge (a = \sum_{i=1}^{k} \lceil a/T_i \rceil C_i))^\frown tt & \{(13)\} & (14)\\
\Rightarrow & \quad (a \leq T_k) \wedge (a > 0) \wedge (a = \sum_{i=1}^{k} \lceil a/T_i \rceil C_i) & \{(14) \text{ and ITL4}\} & (15)\\
\Rightarrow & \quad (\int \bigvee_{i=1}^{k} \mathrm{Run}_i = \ell) \vee (\int \bigvee_{i=1}^{k} \mathrm{Run}_i \neq \ell) & \{\text{tautology}\} & (16)
\end{array}
$$

case 1: $\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell$ $\{$case split on (16)$\}$ (17)

$\Rightarrow\ \exists n \in \mathbb{N}, b \in \mathbb{R}.\ 0 \le b < a\ \wedge \ell = na + b$ $/*$ begin scope $\exists n, b$ $*/$ (18)

$\Rightarrow\quad ((\ell = na) \wedge (\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell))^\frown(\ell = b)$ (19)

$\{$(17), (18), ITL7, and DC14$\}$

$\Rightarrow\quad ((\ell = na) \wedge (\sum_{i=1}^{k}\int\mathrm{Run}_i = \ell))^\frown(\ell = b)$ $\{$(19) and Lemma 2$\}$ (20)

$\Rightarrow\quad (\bigwedge_{i=1}^{k-1}(\int\mathrm{Run}_i \le \lceil na/T_i\rceil C_i))^\frown(\ell = b)$ $\{A_4\}$ (21)

$\Rightarrow\quad \lfloor \ell/T_k\rfloor \le \lceil na/T_k\rceil + \lfloor b/T_k\rfloor$ $\{$(18)$\}$ (22)

$\Rightarrow\quad \lfloor \ell/T_k\rfloor \le \lceil na/T_k\rceil$ $\{$(22) and $b < T_k\}$ (23)

$\Rightarrow\quad \int\mathrm{Run}_k < \lceil na/T_k\rceil C_k$ $\{$(5) and (23)$\}$ (24)

$\Rightarrow\quad (\int\mathrm{Run}_k < \lceil na/T_k\rceil C_k)^\frown(\ell = b)$ $\{$(18), (24), ITL7, and DC15$\}$ (25)

$\Rightarrow\quad (na < \sum_{i=1}^{k}\lceil na/T_i\rceil C_i)^\frown(\ell = b)$ $\{$(20), (21), (25), and DC16$\}$ (26)

$\Rightarrow\quad na < \sum_{i=1}^{k}\lceil na/T_i\rceil C_i$ $\{$(26) and ITL4$\}$ (27)

$\Rightarrow\quad na = n\sum_{i=1}^{k}\lceil a/T_i\rceil C_i$ $\{$(15)$\}$ (28)

$\Rightarrow\quad n\sum_{i=1}^{k}\lceil a/T_i\rceil C_i < \sum_{i=1}^{k}\lceil na/T_i\rceil C_i$ $\{$(27) and (28)$\}$ (29)

$\Rightarrow\quad ff$ $\{$(29) and arithmetics$\}$ (30)

$\Rightarrow\ ff$ $/*$ end scope $\exists n, b$ $*/$ (31)

case 2: $\int\bigvee_{i=1}^{k}\mathrm{Run}_i \ne \ell$ $\{$case split on (16)$\}$ (32)

$\Rightarrow\ tt^\frown\lceil\bigwedge_{i=1}^{k}\neg\mathrm{Run}_i\rceil^\frown(\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell)$ $\{$(32) and DC20$\}$ (33)

$\Rightarrow\ tt^\frown\lceil\bigwedge_{i=1}^{k}\neg\mathrm{Std}_i\rceil^\frown(\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell)$ $\{$(33), (3), $A_3\}$ (34)

$\Rightarrow\ (\bigwedge_{i=1}^{k}\int\mathrm{Run}_i = \lceil \ell/T_i\rceil C_i)^\frown(\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell)$ $\{$(34) and $A_6\}$ (35)

$\Rightarrow\ \exists n' \in \mathbb{N}, b' \in \mathbb{R}.\ 0 \le b' < a\ \wedge$ $/*$ begin scope $\exists n', b'$ $*/$

$(\bigwedge_{i=1}^{k}\int\mathrm{Run}_i = \lceil \ell/T_i\rceil C_i)^\frown((\ell = n'a + b') \wedge (\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell))$ $\{$(35)$\}$ (36)

$\Rightarrow\quad tt^\frown((\ell = n'a) \wedge (\int\bigvee_{i=1}^{k}\mathrm{Run}_i = \ell))^\frown(\ell = b')$ $\{$(36) and DC14$\}$ (37)

$\Rightarrow\quad tt^\frown((\ell = n'a) \wedge (\sum_{i=1}^{k}\int\mathrm{Run}_i = \ell))^\frown(\ell = b')$ $\{$(37) and Lemma 2$\}$ (38)

$\Rightarrow\quad (\bigwedge_{i=1}^{k-1}(\int\mathrm{Run}_i \le \lceil \ell/T_i\rceil C_i))^\frown(\ell = b')$ $\{$(38), A_4 and DC6$\}$ (39)

$\Rightarrow\quad tt^\frown((\ell = n'a) \wedge (\bigwedge_{i=1}^{k-1}(\int\mathrm{Run}_i \le \lceil \ell/T_i\rceil C_i)))^\frown(\ell = b')$

$\{$(36), (39) and DC22$\}$ (40)

$\Rightarrow\quad tt^\frown((\ell = n'a + b') \wedge (\int\mathrm{Run}_k < \lfloor \ell/T_k\rfloor C_k))$ $\{$(5), (36) and DC23$\}$ (41)

$\Rightarrow\quad ff$ $\{$similar to steps (22) $-$ (30)$\}$ (42)

$\Rightarrow\quad ff$ $\{$combine cases 1 and 2$\}$ (43)

$\Rightarrow\ ff$ $/*$ end scope $\exists\ a$ $*/$ (44)

We have hence deduced a contradiction. This completes the proof of the theorem. □

4.2 RM scheduler

Recall we adopt the convention that priorities are in decreasing order from τ_1 to τ_m. The RM scheduler is then specified as

$$Sch_{RM} \mathrel{\widehat{=}} Sch_S \wedge (T_1 \leq T_2 \leq \cdots \leq T_m).$$

An important concept used in Liu and Layland's proof is that of *full utilisation*. A set of tasks is said to fully utilise the processor if the task set is schedulable and any increase of the execution time for any task will cause the task set to be unschedulable. Liu and Layland, as well as most of the subsequent work, including recent papers such as [Devillers and Goossens 2000,Goossens 1999], did not further formalise the concept. Reasoning with this kind of definition is inevitably at a lower level of formality.

We studied a formal definition in [Dong *et al.* 1999], and in the proofs followed we found that the property that the task set is schedulable is not useful. We therefore did not include that property, but still used the term full utilisation in [Dong *et al.* 1999], although it is more appropriate to give it another name which we do now.

DEFINITION 1. *A set of tasks τ_1, τ_2,...,τ_m, with execution times C_1, C_2,...,C_m and periods T_1, T_2,...,T_m, is said to have non-increasable execution time, denoted as $non\_inc(C_1, \cdots, C_m, T_1, \cdots, T_m)$, iff for any $0 < x \leq T_{max}$, $\sum_{i=1}^{m} \lceil x/T_i \rceil C_i \geq x$.*

At any time point x, $\sum_{i=1}^{m} \lceil x/T_i \rceil C_i$ is the total requested execution time of all the tasks until that moment. We can prove that *non_inc* implies that the processor cannot be idle in the interval $[0, T_m]$, and consequently any increase of C_i will make the task set unschedulable by RM (in particular, τ_m will miss its deadline). However, we do not include the proofs since the results are not needed for the theorems concerned in this paper.

Denote $(C_1, \cdots, C_m)$ by C and $(T_1, \cdots, T_m)$ by T. Denote $non\_inc(C_1, \cdots, C_m, T_1, \cdots, T_m)$ by $non\_inc(C, T)$. Similarly, let C' stand for $(C'_1, \cdots, C'_m)$, T' for $(T'_1, \cdots, T'_m)$, we shall abbreviate $non\_inc(C'_1, \cdots, C'_m, T_1, \cdots, T_m)$ as $non\_inc(C', T)$ and $non\_inc(C'_1, \cdots, C'_m, T'_1, \cdots, T'_m)$ as $non\_inc(C', T')$.

Let $lub(m)$ denote the minimum of the utilisation factors over all the sets of m tasks that have non-increasable execution time. Formally,

DEFINITION 2. $lub(m) \mathrel{\widehat{=}} \min\{\sum_{i=1}^{m} C_i/T_i \mid non\_inc(C, T)\}$.

The value of $lub(m)$ for RM was discovered by Liu and Layland [Liu and Layland 1973] and is expressed by the following lemma. The calculation involves many technical details, and a corrected and improved proof is given in the appendix.

LEMMA 4. *For RM, $lub(m) = m(2^{\frac{1}{m}} - 1)$.*

The below fact follows from the property of the function.

COROLLARY 3 OF LEMMA 4. $lub(k) \geq lub(m)$ if $k \leq m$.

The value of $lub(m)$ obviously provides an upper bound for the set of m tasks, in the sense that if its utilisation factor is above the value, then it is quite possible that the task set is unschedulable (whether the task set is schedulable depends on the specific values of execution times and periods). The question is whether $lub(m)$ also provides the lower bound, that is, for any given set of m tasks, if its utilisation factor is less than $lub(m)$, then the task set is schedulable. Liu and Layland found this to be true, but just stated it without proof [Liu and Layland 1973].

However, this property does not follow from the definition directly and as far as we know was only proved recently by Devillers and Goossens [Devillers and Goossens 2000] and us [Dong *et al.* 1999] independently. [2]

THEOREM 3. (SUFFICIENCY FOR RM) $(A \wedge Sch_S \wedge (\sum_{i=1}^{m} C_i/T_i \leq lub(m))) \Rightarrow Req$.

PROOF.

$$
\begin{array}{llr}
A \wedge Sch_S \wedge (\sum_{i=1}^{m} C_i/T_i \leq lub(m)) \wedge \neg Req_k & & (1)\\
\Rightarrow A & \{(1)\} & (2)\\
\Rightarrow Sch_S & \{(1)\} & (3)\\
\Rightarrow \sum_{i=1}^{m} C_i/T_i \leq lub(m) & \{(1)\} & (4)\\
\Rightarrow \neg Req_k & \{(1)\} & (5)\\
\Rightarrow ((\ell = T_k) \wedge (\textstyle\int \mathrm{Run}_k < C_k))^\frown tt & \{(2), (3) \text{ and Theorem 2}\} & (6)\\
\Rightarrow ((\ell = T_k) \wedge (\textstyle\int \mathrm{Run}_k < C_k) \wedge (tt^\frown \lceil \mathrm{Std}_k \rceil))^\frown tt & \{(2) \text{ and } (6)\} & (7)\\
\Rightarrow (\,(\ell = T_k) \wedge \lceil \bigvee_{i=1}^{k} \mathrm{Run}_i \rceil \wedge (\textstyle\int \mathrm{Run}_k < C_k)\,)^\frown tt & & (8)\\
 & \{(2), (3), (7) \text{ and Lemma 3}\} & \\
\Rightarrow \exists C'_k.\, 0 \leq C'_k < C_k \wedge & /*\ \text{formulae below are within } \exists\ */ & \\
\Rightarrow \quad (\,(\ell = T_k) \wedge \lceil \bigvee_{i=1}^{k} \mathrm{Run}_i \rceil \wedge (\textstyle\int \mathrm{Run}_k = C'_k)\,)^\frown tt & \{(8) \text{ and DC10}\} & (9)\\
\Rightarrow \quad \forall x. 0 < x \leq T_k \ \Rightarrow & /*\ \text{formulae below are within } \forall\ */ & \\
\Rightarrow \qquad (\ell = T_k) \Rightarrow ((\ell = x)^\frown(\ell = T_k - x)) & \{\text{ITL7}\} & (10)\\
\Rightarrow \qquad (\,((\ell = x)^\frown(\ell = T_k - x)) \wedge \lceil \bigvee_{i=1}^{k} \mathrm{Run}_i \rceil \wedge (\textstyle\int \mathrm{Run}_k = C'_k)\,)^\frown tt & & (11)\\
 & \{(9) \text{ and } (10)\} & \\
\Rightarrow \qquad (\,(\ell = x) \wedge \lceil \bigvee_{i=1}^{k} \mathrm{Run}_i \rceil \wedge (\textstyle\int \mathrm{Run}_k \leq C'_k)\,)^\frown tt & & (12)\\
 & \{(11),\ \text{DC14 and DC15}\} & \\
\Rightarrow \qquad (\,(\ell = x) \wedge (\ell = \sum_{i=1}^{k} \textstyle\int \mathrm{Run}_i) \wedge (\textstyle\int \mathrm{Run}_k \leq C'_k)\,)^\frown tt & & (13)\\
 & \{(12),\ \text{DC13 and Corollary 2}\} & \\
\Rightarrow \qquad (\, x \leq (\sum_{i=1}^{k-1} \lceil x/T_i \rceil C_i) + C'_k\,)^\frown tt & \{(2) \text{ and } (13)\} & (14)\\
\Rightarrow \qquad x \leq (\sum_{i=1}^{k-1} \lceil x/T_i \rceil C_i) + C'_k & \{(14) \text{ and ITL4}\} & (15)
\end{array}
$$

[2] Liu and Layland's statement was of course given using full utilisation and Devillers and Goossens proved that. Although we used the term full utilisation in [Dong *et al.* 1999], the actual definition was non-increasable execution time.

$$\begin{array}{llrr}
\Rightarrow & non_inc(C_1, \cdots, C_{k-1}, C'_k, T_1, \cdots, T_k) & \{(15) \text{ and the def. of } non_inc\} & (16) \\
\Rightarrow & lub(k) \leq (\sum_{i=1}^{k-1} C_i/T_i) + C'_k/T_k & \{(16) \text{ and the def. of } lub(k)\} & (17) \\
\Rightarrow & (\sum_{i=1}^{k-1} C_i/T_i) + C'_k/T_k < \sum_{i=1}^{m} C_i/T_i & \{C'_k < C_k \text{ and } k \leq m\} & (18) \\
\Rightarrow & lub(k) < lub(m) & \{(4), (17) \text{ and } (18)\} & (19) \\
\Rightarrow & ff & \{(19) \text{ and Cor. of Lemma 4}\} & (20) \\
\Rightarrow ff & & \{(20)\} & (21)
\end{array}$$

This completes the proof for the theorem. □

5. Proof for EDF

EDF assigns priorities to tasks dynamically according to the distance to their deadlines. The task closer to the deadline has a higher priority.

The following formula describes that at least in the latter part of the considered interval, task τ_i is more urgent than task τ_j:

$$\text{urgent}(i, j) \widehat{=} \lceil \ell/T_i \rceil T_i < \lceil \ell/T_j \rceil T_j.$$

The subinterval on which task τ_i is more urgent than task τ_j may be very small or it may be the whole interval. Once such a subinterval exists, and task τ_j is running over it, then task τ_i cannot have a standing request:

$$Sch_{EDF} \widehat{=} \Box_p\,(\,((tt^\frown \lceil \text{Run}_j \rceil) \wedge \text{urgent}(i, j)) \;\Rightarrow\; (tt^\frown \lceil \neg \text{Std}_i \rceil)\,).$$

As for EDF, Liu and Layland discovered that for a task set to be scheduable, it is sufficient that its utilisation factor is not greater than 1 (this is of course also the necessary condition, proven formally in Section 3.6, as for all the scheduling policies).

THEOREM 4. (SUFFICIENCY FOR EDF) $(A \wedge Sch_{EDF} \wedge (\sum_{i=1}^{m} C_i/T_i \leq 1)) \Rightarrow Req.$

The proof is by contradiction. Suppose that the requirement is not satisfied, then there exists k,

$$\Diamond_p\,(\,\textstyle\int \text{Run}_k < \lfloor \ell/T_k \rfloor C_k\,).$$

We shall deduce that $\sum_{i=1}^{m} C_i/T_i > 1$. We first prove the following lemma.

LEMMA 5. $A_2 \;\Rightarrow \Box\,(\,((\bigwedge_{i=1}^{m}(\int Run_i \leq \lfloor \ell/T_i \rfloor C_i)) \;\wedge\; (\int Run_k < \lfloor \ell/T_k \rfloor C_k) \wedge$
$$(\textstyle\int \bigvee_{i=1}^{m} Run_i = \ell)) \;\;\Rightarrow\; (\sum_{i=1}^{m} C_i/T_i > 1)\,).$$

By DC25, we only need to prove

$$A_2 \;\Rightarrow (\,((\textstyle\bigwedge_{i=1}^{m}(\int \text{Run}_i \leq \lfloor \ell/T_i \rfloor C_i)) \;\wedge\; (\int \text{Run}_k < \lfloor \ell/T_k \rfloor C_k) \wedge$$

$$(\textstyle\int \bigvee_{i=1}^{m} \text{Run}_i = \ell)) \Rightarrow\; (\sum_{i=1}^{m} C_i/T_i > 1)\,).$$

PROOF.

$$A_2 \wedge (\bigwedge_{i=1}^{m}(\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i)) \wedge (\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k) \wedge (\int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell) \quad (1)$$
$$\Rightarrow (\bigwedge_{i=1}^{m}(\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i)) \wedge (\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k) \wedge (\sum_{i=1}^{m} \int \mathrm{Run}_i = \ell) \quad (2)$$
{Lemma 2}
$$\Rightarrow \sum_{i=1}^{m} \lfloor \ell/T_i \rfloor C_i > \ell \quad (3)$$
$$\Rightarrow \sum_{i=1}^{m} C_i/T_i > 1 \quad (4)$$

□

Now, we come to the proof for Theorem 4. Let

$$\alpha(i) \ \widehat{=}\ mult_i^\frown((\ell < T_i) \wedge (\int \mathrm{Run}_i = 0)), \quad (1)$$
$$\beta(i) \ \widehat{=}\ mult_i^\frown((\ell < T_i) \wedge (\int \mathrm{Run}_i \neq 0)). \quad (2)$$

PROOF.

$$A \wedge Sch_{EDF} \wedge \Diamond_p (\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k) \quad (3)$$
$$\Rightarrow \Diamond_p (\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k) \qquad \{ (3) \} \quad (4)$$
$$\Rightarrow \exists n \in \mathbb{N}, \exists r \in \mathbb{R}. 0 \leq r < T_k \wedge ((\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k) \wedge (\ell = nT_k + r))^\frown tt \quad (5)$$
$$\Rightarrow \Diamond_p \qquad /*\ \text{formulae below are within } \Diamond_p\ */$$
$$(\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k) \wedge mult_k \qquad \{(5),\ \mathrm{DC1} \text{ and def. of } \Diamond_p\} \quad (6)$$
$$\Rightarrow \int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k \qquad \{(6)\} \quad (7)$$
$$\Rightarrow mult_k \qquad \{(6)\} \quad (8)$$
$$\Rightarrow \bigwedge_{i=1}^{m}(\alpha(i) \vee \beta(i)) \qquad \{ (1) \text{ and } (2)\} \quad (9)$$
$$\Rightarrow (\bigvee_{i=1}^{m}\beta(i)) \vee (\bigwedge_{i=1}^{m}\neg\beta(i)) \qquad \{\text{tautology}\} \quad (10)$$
$$\text{case } 1: \bigwedge_{i=1}^{m}\neg\beta(i) \qquad \{\text{case split on } (10)\} \quad (11)$$
$$\Rightarrow \bigwedge_{i=1}^{m}\alpha(i) \qquad \{(9) \text{ and } (11)\} \quad (12)$$
$$\Rightarrow \bigwedge_{i=1}^{m}(\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i) \qquad \{A_4, (12), \text{ and } (1)\} \quad (13)$$
$$\Rightarrow (\int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell) \vee (\int \bigvee_{i=1}^{m} \mathrm{Run}_i \neq \ell) \qquad \{\text{tautology}\} \quad (14)$$
$$\text{case } 1.1: \int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell \qquad \{\text{case split on } (14)\} \quad (15)$$
$$\Rightarrow \sum_{i=1}^{m} C_i/T_i > 1 \qquad \{(7),\ (13),\ (15),\ \text{and Lemma 5}\} \quad (16)$$
$$\text{case } 1.2: \int \bigvee_{i=1}^{m} \mathrm{Run}_i \neq \ell \qquad \{\text{case split on } (14)\} \quad (17)$$
$$\Rightarrow tt^\frown \lceil \bigwedge_{i=1}^{m} \neg \mathrm{Run}_i \rceil^\frown (\int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell) \qquad \{(17) \text{ and DC20}\} \quad (18)$$
$$\Rightarrow tt^\frown \lceil \bigwedge_{i=1}^{m} \neg \mathrm{Std}_i \rceil^\frown (\int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell) \qquad \{(18) \text{ and } A_3\} \quad (19)$$
$$\Rightarrow (\bigwedge_{i=1}^{m} \int \mathrm{Run}_i = \lceil \ell/T_i \rceil C_i)^\frown (\int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell) \qquad \{(19) \text{ and } A_6\} \quad (20)$$
$$\Rightarrow (\bigwedge_{i=1}^{m} \int \mathrm{Run}_i = \lceil \ell/T_i \rceil C_i)^\frown ((\int \bigvee_{i=1}^{m} \mathrm{Run}_i = \ell)$$
$$\wedge \bigwedge_{i=1}^{m} (\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i) \wedge (\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor C_k)) \quad (21)$$
{(7), (13), (20), DC22, and DC23}
$$\Rightarrow (\bigwedge_{i=1}^{m} \int \mathrm{Run}_i = \lceil \ell/T_i \rceil C_i)^\frown (\sum_{i=1}^{m} C_i/T_i > 1) \quad (22)$$
{(21) and Lemma 5}
$$\Rightarrow \sum_{i=1}^{m} C_i/T_i > 1 \qquad \{(22) \} \quad (23)$$

$$\Rightarrow\ \textstyle\sum_{i=1}^{m} C_i/T_i > 1 \qquad \{\text{combine 1.1 and 1.2}\} \quad (24)$$

$$\text{case 2}:\ \textstyle\bigvee_{i=1}^{m}\beta(i) \qquad \{\text{case split on (10)}\} \quad (25)$$

$$\Rightarrow\ \Gamma\cup\Delta=\{1,\cdots,m\}\wedge\Delta\neq\emptyset,$$

$$\text{where } \Gamma\widehat{=}\{1\le i\le m \mid \alpha(i)\} \text{ and } \Delta\widehat{=}\{1\le i\le m\mid\beta(i)\}. \qquad \{(9) \text{ and } (25)\} \quad (26)$$

$$\Rightarrow\ \textstyle\bigwedge_{i\in\Delta}(\ \exists x_i\in\mathbb{R}.x_i\ge 0\ mult_i{}^\frown((\ell<T_i)$$

$$\wedge\,(tt^\frown\lceil \mathrm{Run}_i\rceil^\frown((\textstyle\int\mathrm{Run}_i=0)\wedge(\ell=x_i)))))\ ) \quad (27)$$

$$\{(2), (26), \text{DC10 and DC20}\}$$

$$\Rightarrow\ mult_u{}^\frown((\ell<T_u)\ \wedge\ (tt^\frown\lceil\mathrm{Run}_u\rceil^\frown(\ell=x)))$$

$$\wedge\,(\textstyle\bigwedge_{i\in\Delta}(mult_i{}^\frown((\ell<T_i)\wedge(tt^\frown((\int\mathrm{Run}_i=0)\wedge(\ell=x)))),$$

$$\text{where } x=\min\{x_i\mid i\in\Delta\} \text{ and } u=\min\{i\in\Delta\mid x_i=x\} \qquad \{(27)\} \quad (28)$$

$$\Rightarrow\ \textstyle\bigwedge_{i\in\Gamma}\alpha(i) \qquad \{\text{definition of }\Gamma\} \quad (29)$$

$$\Rightarrow\ \textstyle\bigwedge_{i\in\Gamma}(\int\mathrm{Run}_i\le\lfloor\ell/T_i\rfloor C_i) \qquad \{A_4 \text{ and } (1)\} \quad (30)$$

$$\Rightarrow\ (tt^\frown((\textstyle\int\bigvee_{i=1}^{m}\mathrm{Run}_i=\ell)\wedge(\ell=x)))$$

$$\vee\ (tt^\frown((\textstyle\int\bigvee_{i=1}^{m}\mathrm{Run}_i\neq\ell)\wedge(\ell=x))) \qquad \{\text{tautology}\} \quad (31)$$

$$\text{case 2.1}:\ tt^\frown((\textstyle\int\bigvee_{i=1}^{m}\mathrm{Run}_i\neq\ell)\wedge(\ell=x)) \qquad \{\text{case split on (31)}\} \quad (32)$$

$$\Rightarrow\ tt^\frown((\lceil\wedge_{i=1}^{m}\neg\mathrm{Run}_i\rceil^\frown(\textstyle\int\bigvee_{i=1}^{m}\mathrm{Run}_i=\ell))\wedge(\ell=x)) \qquad \{(32) \text{ and DC20}\} \quad (33)$$

$$\Rightarrow tt^\frown\lceil\wedge_{i=1}^{m}\neg\mathrm{Std}_i\rceil^\frown((\textstyle\int\bigvee_{i=1}^{m}\mathrm{Run}_i=\ell)\wedge(\ell<x)) \qquad \{(33) \text{ and } A_3\} \quad (34)$$

$$\Rightarrow\ (\textstyle\bigwedge_{i=1}^{m}\int\mathrm{Run}_i=\lceil\ell/T_i\rceil C_i)^\frown((\int\bigvee_{i=1}^{m}\mathrm{Run}_i=\ell)\wedge(\ell<x)) \quad \{(34) \text{ and } A_6\} \quad (35)$$

$$\Rightarrow\ (\textstyle\bigwedge_{i=1}^{m}\int\mathrm{Run}_i=\lceil\ell/T_i\rceil C_i)^\frown(\ (\int\bigvee_{i=1}^{m}\mathrm{Run}_i=\ell)\wedge(\bigwedge_{i\in\Delta}(\int\mathrm{Run}_i=0))$$

$$\wedge(\textstyle\bigwedge_{i\in\Gamma}(\int\mathrm{Run}_i\le\lfloor\ell/T_i\rfloor C_i))\wedge(\int\mathrm{Run}_k<\lfloor\ell/T_k\rfloor\cdot C_k)\ ) \quad (36)$$

$$\{(7), (28), (30), \text{DC22 and DC23}\}$$

$$\Rightarrow\ (\textstyle\bigwedge_{i=1}^{m}\int\mathrm{Run}_i=\lceil\ell/T_i\rceil C_i)^\frown(\sum_{i=1}^{m}C_i/T_i>1) \qquad \{(36) \text{ and Lemma 5}\} \quad (37)$$

$$\Rightarrow \textstyle\sum_{i=1}^{m}C_i/T_i>1 \qquad \{(37)\} \quad (38)$$

$$\text{case 2.2}:\ tt^\frown((\textstyle\int\bigvee_{i=1}^{m}\mathrm{Run}_i=\ell)\wedge(\ell=x)) \qquad \{\text{case split on (31)}\} \quad (39)$$

$$\Rightarrow\ \textstyle\bigwedge_{i\in\Gamma}\ mult_i^\frown((\ell<T_i)\wedge(\int\mathrm{Run}_i=0)) \qquad \{(29) \text{ and } (1)\} \quad (40)$$

$$\Rightarrow\ \textstyle\bigwedge_{i\in\Gamma}\ \exists x_i\in\mathbb{R}.x_i\ge 0$$

$$mult_i^\frown((\ell=x_i)\wedge(\textstyle\int\mathrm{Run}_i=0)) \qquad \{(40) \text{ and DC10}\} \quad (41)$$

$$\Rightarrow\ (x_i<x)\vee(x_i\ge x) \qquad \{\text{tautology}\} \quad (42)$$

$$\text{case 2.2.1}:\ x_i<x \qquad \{\text{case split on (42)}\} \quad (43)$$

$$\Rightarrow\ \mathrm{urgent}(i,u)^\frown(\ell=x) \qquad \{(28), (41), (43), \text{ and def. of urgent}\} \quad (44)$$

$$\Rightarrow\ tt^\frown\lceil\neg\mathrm{Std}_i\rceil^\frown(\ell=x) \qquad \{(28), (44) \text{ and } Sch_{EDF}\} \quad (45)$$

$$\Rightarrow\ (\textstyle\int\mathrm{Run}_i=\lceil\ell/T_i\rceil C_i)^\frown(\ell=x) \qquad \{(45) \text{ and } A_6\} \quad (46)$$

$$\Rightarrow\ tt^\frown((\textstyle\int\mathrm{Run}_i\le\lfloor\ell/T_i\rfloor C_i)\wedge(\ell=x)) \qquad \{(30), (46) \text{ and DC22}\} \quad (47)$$

$$\Rightarrow\ tt^\frown((\textstyle\int\mathrm{Run}_k<\lfloor\ell/T_k\rfloor C_k)\wedge(\ell=x)) \qquad \{(7), (46) \text{ and DC23}\} \quad (48)$$

$$
\begin{array}{lll}
\text{case 2.2.2: } x_i \geq x & \{\text{case split on (42)}\} & (49) \\
\quad \Rightarrow\ tt^\frown((\ell = x) \wedge (\textstyle\int \mathrm{Run}_i = 0)) & \{(41), (49), \mathrm{DCA1}, & (50) \\
 & \text{and DCA5}\} & \\
\Rightarrow\ tt^\frown((\ell = x) \wedge (\textstyle\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i)) & \{\text{combine 2.2.1 and 2.2.2}\} & (51) \\
\Rightarrow\ tt^\frown((\ell = x) \wedge (\bigwedge_{i\in\Gamma}(\textstyle\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i)) \wedge (\textstyle\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor \cdot C_k)\ ) & & (52) \\
 & \{(48) \text{ and } (51)\} & \\
\Rightarrow\ tt^\frown((\ell = x) \wedge (\bigwedge_{i\in\Delta}(\textstyle\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i))) & \{(28)\} & (53) \\
\Rightarrow\ tt^\frown((\ell = x) \wedge (\textstyle\int \bigvee_{i=1}^m \mathrm{Run}_i = \ell) \wedge (\bigwedge_{i=1}^m (\textstyle\int \mathrm{Run}_i \leq \lfloor \ell/T_i \rfloor C_i)) & & \\
\quad \wedge (\textstyle\int \mathrm{Run}_k < \lfloor \ell/T_k \rfloor \cdot C_k)\ ) & \{(39), (52), \text{and } (53)\} & (54) \\
\Rightarrow\ tt^\frown(\sum_{i=1}^m C_i/T_i > 1) & \{(54) \text{ and Lemma 5}\} & (55) \\
\Rightarrow\ \sum_{i=1}^m C_i/T_i > 1 & \{(55)\} & (56) \\
\Rightarrow\ \sum_{i=1}^m C_i/T_i > 1 & \{\text{combine 2.1 and 2.2}\} & (57) \\
\Rightarrow\ \sum_{i=1}^m C_i/T_i > 1 & \{\text{combine cases 1 and 2}\} & (58) \\
 & /*\ \text{the above is inside } \Diamond_p\ */ & \\
\Rightarrow\ \sum_{i=1}^m C_i/T_i > 1 & \{(58)\} & (59)
\end{array}
$$

We have deduced $\sum_{i=1}^m C_i/T_i > 1$, a contradiction, hence completed the proof. □

6. Related Work

There is some other work on formal verification of scheduling theorems. Wilding [Wilding 1998] verified EDF using the Nqthm theorem prover and Dutertre [Dutertre 2000] verified priority ceiling protocol in PVS.

Recently, there is a lot of work applying techniques developed in model checking, mainly on timed automata [Alur and Dill 1994], to scheduling. Roughly, they can be classified into the schedulability analysis and the controller synthesis. The idea of former is to model real-time systems, including a particular scheduling policy, by (a variant of) timed automata. The schedulability problem is formulated as the reachability problem which can be model checked, see e.g. [Fersman *et al.* 2007]. The advantage of this approach is that it can handle more general scheduling problems (e.g., tasks are non-periodic) where the traditional schedulability analysis method has no general solutions. The controller synthesis approach is to achieve schedulability by construction. It was first proposed in [Wong-Toi and Hoffmann 1992] and further studied in [Maler *et al.* 1995,Altisen *et al.* 2002]. There is also considerable amount of research on optimal scheduling using timed automata, e.g. [Alur *et al.* 2001,Abdedaim *et al.* 2006,Bouyer *et al.* 2008].

Schedulability analysis based on other formal techniques has also been studied. For example, process algebra was used to model scheduling problems and schedulability is checked by symbolic weak bisimulation [Kwak *et al.* 1998].

However, it is unlikely that the model checking based methods for scheduling can be used to prove general scheduling theorems. Model checking is usually limited to a specific system, and is not easy to be extended to system with arbitrary number of tasks and parameters (execution times etc).

7. Conclusions

In this paper, we have formalised the two classic scheduling algorithms, i.e., RM and EDF, and formally proven their schedulability theorems. Our proofs are based in a large part on the intuitions of Liu and Layland's original work [Liu and Layland 1973]. This says that there are common grounds between formal and informal proofs. However, there is a lot of work to produce formal proofs from intuitive arguments. The reward for this somewhat arduous effort is that the proofs are now much more reliable. This does not mean that formal proofs are always correct, but there are certainly fewer chances for mistakes to creep in, because now concepts and definitions are formed without ambiguity and deduction is by well-established proof rules. Therefore, formal proofs can be subject to precise scrutiny. On the contrary, in an informal proof, concepts and definitions can be ambiguous, deduction can be not much more than hand waving, consequently, mistakes are more likely to occur, and one may remain unconvinced for something which is indeed correct. This has indeed happened with the informal proof of RM.

A proof is only meaningful if the assumptions are correct. In this paper in particular, the assumptions model the environment and the scheduling algorithms. These assumptions are based on the intuitive understanding of the system (at a lower level, one may want to formally verify whether these assumptions are indeed properties of the system, but this is out of the scope of the current paper). Most of the assumptions do not pose any questions, but there is an exception with A_6. The current form stipulates that if a task is not requesting at the end of an interval, then the maximal required execution time over the whole interval has been satisfied. This is not a problem if there are no overflows happened before the last period is started. However, when there is an overflow happened before the last period, our assumption is based on the view that the missing execution time will be carried over to the subsequent periods. A particular system may be implemented in a different way, for example, the task which has not completed in a period is simply removed, or the system calls an exception handling procedure when an overflow occurs. Ideally, the formal model should include all these cases, but we have not been able to do so. We can of course formalise different models and prove the feasibility conditions separately. In fact, some of the earlier work assumed that incomplete tasks are removed. Most of the proof steps are actually similar.

In this paper, the formal logic we used is DC. DC is designed to specify and reason about real time behavior over intervals, and is a suitable tool for formalising scheduling theories because the accumulated running time of a task is associated with an interval and can be conveniently expressed. DC provides an abstraction for intervals (so one does not refer to an interval explicitly in a formula, say in the form of [b,e]). The price for this is that the logic is more demanding to learn. However, many computer professionals may not have a strong background in logic. For people who are more accustomed to set theory or classic first order logic, an alternative is to use basically the semantics of DC and this gives a first order logic with a special variable for time. A calculus similar to this, but in a set-theoretic notation, is the Timed Interval Calculus [Fidge *et al.* 1998]. Another shortcoming of DC is that to follow the principle "small is beautiful", state variables are restricted to a special kind (namely, boolean functions of time) and terms are only defined over intervals. These restrictions are not a serious problem for our paper, except the scheduling policy of EDF has to be expressed in a somewhat indirect way. One would like to be able to talk more directly about it, say by having a function which gives the next deadline value. It is possible to extend DC to include such features and in fact several variants of DC have been developed, e.g. Extended Duration Calculus [Zhou *et al.* 1993]. However, having several variants of DC may not be desirable. In this case, one may also consider using the first order logic over time corresponding to the semantics of DC, since less work is expected in extending it.

The proofs in this paper are the longest on applications of DC as far as we are aware of. We found it is difficult to be rigorous with previous styles of DC proofs in the literature to handle proofs of this size. Our way of writing DC proofs is new and we expect it to be useful in writing other long proofs in DC.

A great deal of research has been done on the theory and implementation of theorem proving systems, providing automatic or semi-automatic support to various formal logics. Proofs conducted on a theorem prover are usually much more reliable than those by human beings, because the system is carefully constructed by specialists and constantly debugged by the large number of people using it. A couple of theorem provers exist for DC, usually based on some other theorem proving systems. However, on the whole, there is not much work on theorem provers for DC. When the theorem provers for DC are further developed, it may be interesting to try the proofs in this paper mechanically.

Acknowledgements

We are deeply indebted to Prof. Zhou Chaochen for introducing us to this area and for many discussions over the years. We thank Dong Shuzhen for her work in some proofs, and Sun Haiwei for suggesting the proof for Lemma 8 using elementary mathematics, and the anonymous referees for their criticisms and constructive comments that have led to substantial improvements.

References

ABDEDAIM, Y., ASARIN, E., AND MALER, O. 2006. Scheduling with Timed Automata. *Theoretical Computer Sciences 354*, 272–300.

ALTISEN, K., GOESSLER, G., AND SIFAKIS, J. 2002. Scheduler Modeling Based on the Controller Synthesis Paradigm. *Journal of Real-Time Systems 23*, 55–84.

ALUR, R. AND DILL, D.L. 1994. A theory of Timed Automata. *Theoretical Computer Sciences 126*, 2, 183–235.

ALUR, R., LA TORRE, S., AND PAPPAS, G.J. 2001. Optimal Paths in Weighted Timed Automata. In *Proc. of HSCC'01*, Volume 2034 of *LNCS*, 49–62.

BACK, R.-J. R., GRUNDY, J., AND VON WRIGHT, J. 1997. *Formal Aspects of Computing 9*, 469–483.

BOUYER, P., BRINKSMA, ED, AND LARSEN, K.G. 2008. *Formal Methods in System Designs 32*, 1, 3–23.

BUTTAZZO, G. B. 1997. *Hard Real Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic.

DEVILLERS, R. AND GOOSSENS, J. 2000. Liu and Layland's Schedulability Test Revisited. *Information Processing Letters 73*, 157–161.

DIJKSTRA, E.W. AND SCHOLTEN, C.S. 1990. *Predicate Calculus and Program Semantics*. Springer-Verlag.

DONG, S., XU, Q., AND ZHAN, N. 1999. A formal proof for the Rate Monotonic Scheduler. In *Proc. the Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA'99)*. IEEE Computer Society Press, 500–507.

DUTERTRE, B. 1995. Complete Proof Systems for First Order Interval Temporal Logic. In *Proc. Logic in Computer Science 95*. IEEE Computer Society Press.

DUTERTRE, B. 2000. Formal Analysis of the Priority Ceiling Protocol. In *Proc. of 19th AIAA/IEEE Digital Avionics Systems Conference*, 151–160.

FERSMAN, E., KRCAL, P., PETTERSSON, P., AND WANG, Y. 2007. Tasks Automata: Schedulability, Decidability and Undecidability. *Information and Computation 205*, 1149–1172.

FIDGE, C.J., HAYS, I.J., MARTIN, A.P., AND WABENHORST, A.K. 1998. In *Proc. Mathematics of Program Construction (MPC'98)*, Volume 1422 of *LNCS*, 188–206.

GOOSSENS, J. 1999. Scheduling of Hard Real-Time Periodic Systems with Various Kinds of Deadline and Offset Constraints. Phd thesis, Université Libre de Bruxelles.

HANSEN, M.R. AND ZHOU, C.C. 1997. Duration Calculus: Logical Foundations. *Formal Aspects of Computing 9*, 283–330.

KWAK, H-H., LEE, I., PHILIPPOU, A., CHOI, J-Y., AND SOKOLSKY, O. 1998. Symbolic Schedulability Analysis of Real-time Systems. In *Proc. of IEEE Real-Time Systems Symposium'98*, 409–.

LIU, C.L. AND LAYLAND, J.W. 1973. Scheduling Algorithm for Multiprogramming in a Hard Real-time Environment. *Journal of the ACM 20*, 1, 46–61.

MALER, O., PNUELI, A., AND SIFAKIS, J. 1995. On the Synthesis of Discrete Controllers for Timed Systems. In *Proc. STACS'95*, Volume 900 of *LNCS*, 229–242.

MOSZKOWSKI, B. 1985. A temporal logic for Multi-level Reasoning About Hardware. *IEEE Computer 18*, 2, 10–19.

WILDING, M. 1998. A Machine-checked Proof of the Optimality of a Real-time Scheduling Policy. In *Proc. of CAV'98*, Volume 1427 of *LNCS*, 369–378.

WONG-TOI, H. AND HOFFMANN, G. 1992. The Control of Dense Real-time Discrete Event Systems. Technical Report STAN-CS-92-1411, Standford Univesity.

ZHAN, N. 2000. Another Formal Proof for the Deadline Driven Scheduler. In *Proc. the Seventh International Conference on Real-Time Computing Systems and Applications (RTCSA'00)*. IEEE Computer Society Press, 481–485.

ZHENG, Y. AND ZHOU, C.C. 1994. A Formal Proof of the Deadline Driven Scheduler. In *Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, Volume 863 of *LNCS*, 756–775.

ZHOU, C.C. AND HANSEN, M.R. 2004. *Duration Calculus: A Formal Approach to Real-Time Systems*. Springer-Verlag.

ZHOU, C.C., HOARE, C.A.R., AND RAVN, A.P. 1991. A Calculus of Durations. *Information Processing Letters 40*, 5, 269–276.

ZHOU, C.C., RAVN, A.P., AND HANSEN, M.R. 1993. An Extended Duration Calculus for Hybrid Systems. In *Proc. of Hybrid Systems'93, LNCS 736*, Volume 736 of *LNCS*, 36–59.

Appendix A.

We now prove Lemma 4. As in Liu and Layland's proof, we first consider the case that one period is at least half of any other period, or formally, $0 < T_i/T_j < 2$ for all $1 \leq i, j \leq m$.

In this case, $non\_inc(C, T)$ holds if the inequality holds for every period points.

LEMMA 6. *If* $0 < T_i/T_j < 2$ *for all* $1 \le i, j \le m$, *then*

$$non\_inc(C, T) \Leftrightarrow \bigwedge_{i=1}^{m}(T_i \le \sum_{j=1}^{m}\lceil T_i/T_j\rceil C_j).$$

PROOF. By the definition of $non\_inc(C, T)$ (full utilisation), the $\Rightarrow$ direction is trivial, since it just corresponds to the special cases where x is instantiated by the periods respectively. We next prove the $\Leftarrow$ direction. For notational convenience, let $T_0 = 0$.

$$\begin{aligned}
& (0 < x \le T_m) \wedge (\bigwedge_{i=1}^{m}(\sum_{j=1}^{m}\lceil T_i/T_j\rceil C_j \ge T_i)) && (1)\\
\Rightarrow\ & (\bigvee_{i=1}^{m}(T_{i-1} < x \le T_i)) \wedge (\bigwedge_{i=1}^{m}(\sum_{j=1}^{m}\lceil T_i/T_j\rceil C_j \ge T_i)) && (2)\\
\Rightarrow\ & \bigvee_{i=1}^{m}((T_{i-1} < x \le T_i) \wedge (\bigwedge_{i=1}^{m}(\sum_{j=1}^{m}\lceil T_i/T_j\rceil C_j \ge T_i))) && (3)\\
\Rightarrow\ & \bigvee_{i=1}^{m}((x \le T_i) \wedge (\bigwedge_{j=1}^{m}(\lceil x/T_j\rceil = \lceil T_i/T_j\rceil)) \\
& \quad \wedge (\sum_{j=1}^{m}\lceil T_i/T_j\rceil C_j \ge T_i)) && (4)\\
& \qquad \{0 < T_i/T_j < 2 \text{ for all } 1 \le i, j \le m\} \\
\Rightarrow\ & \bigvee_{i=1}^{m}(\sum_{j=1}^{m}\lceil x/T_j\rceil C_j \ge x) && (5)\\
\Rightarrow\ & \sum_{j=1}^{m}\lceil x/T_j\rceil C_j \ge x && (6)
\end{aligned}$$

□

The next lemma indicates that the minimal value is reached when $C_i = T_{i+1} - T_i$ for $i = 1, \ldots, m-1$ and $C_m = 2T_1 - T_m$.

LEMMA 7. *If* $0 < T_i/T_j < 2$ *for all* $1 \le i, j \le m$, *then*

$$\begin{aligned}
& \min\{\sum_{i=1}^{m}C_i/T_i \mid non_inc(C, T)\} \\
= \ & \min\{\sum_{i=1}^{m}C_i/T_i \mid (\bigwedge_{i=1}^{m-1}C_i = T_{i+1} - T_i) \wedge C_m = 2T_1 - T_m\}.
\end{aligned}$$

PROOF. By definition, it is easy to prove $non\_inc(C, T)$ holds when $C_i = T_{i+1} - T_i$ for $i = 1 \cdots m-1$ and $C_m = 2T_1 - T_m$ and therefore follows that

$$\begin{aligned}
& \min\{\sum_{i=1}^{m}C_i/T_i \mid non_inc(C, T)\} \\
\le \ & \min\{\sum_{i=1}^{m}C_i/T_i \mid \bigwedge_{i=1}^{m-1}C_i = T_{i+1} - T_i \wedge C_m = 2T_1 - T_m\}
\end{aligned}$$

We next prove the converse direction. For this, it is enough to prove that if $non\_inc(C, T)$ holds, then $\sum_{i=1}^{m} C_i/T_i \ge (\sum_{i=1}^{m-1}(T_{i+1} - T_i)/T_i) + (2T_1 - T_m)/T_m$. From Lemma 6, it results that there exist $\alpha_i \ge 1, i = 1 \ldots m$ such that

$$\begin{aligned}
C_1 + C_2 + \cdots + C_{m-1} + C_m &= \alpha_1 T_1 \\
2C_1 + C_2 + \cdots + C_{m-1} + C_m &= \alpha_2 T_2 \\
&\vdots \\
2C_1 + 2C_2 + \cdots + 2C_{m-1} + C_m &= \alpha_m T_m
\end{aligned}$$

Thus,

$$\begin{array}{rcl} C_1 & = & \alpha_2 T_2 - \alpha_1 T_1 \\ & \vdots & \\ C_{m-1} & = & \alpha_m T_m - \alpha_{m-1} T_{m-1} \\ C_m & = & 2\alpha_1 T_1 - \alpha_m T_m \end{array}$$

Hence,

$$\begin{array}{rll} & \sum_{i=1}^{m} C_i/T_i - ((\sum_{i=1}^{m-1}(T_{i+1} - T_i)/T_i) + (2T_1 - T_m)/T_m) & \\ = & \sum_{i=1}^{m-1}[(\alpha_{i+1} - 1)T_{i+1}/T_i - (\alpha_i - 1)\ ] + \ (\alpha_1 - 1)2T_1/T_m - (\alpha_m - 1) & \\ = & \sum_{i=1}^{m-1}[(\alpha_{i+1} - 1)T_{i+1}/T_i - (\alpha_{i+1} - 1)] \ + \ (\alpha_1 - 1)2T_1/T_m - (\alpha_1 - 1) & \\ \geq & 0 & \{T_i < T_{i+1}, 2T_1 > T_m\} \end{array}$$

□

Now we are ready to calculate the minimal value. The calculation in Liu and Layland's paper [Liu and Layland 1973] is complicated and missing steps. In the following, a straightforward proof using only elementary mathematics is given.

LEMMA 8. *If* $0 < T_i/T_j < 2$ *for all* $1 \leq i, j \leq m$, *then*

$$\min\{\sum_{i=1}^{m} C_i/T_i \mid (\bigwedge_{i=1}^{m-1}(C_i = T_{i+1} - T_i)) \wedge C_m = 2T_1 - T_m\} = m(2^{\frac{1}{m}} - 1).$$

PROOF. For $i = 1 \ldots m-1$, $C_i/T_i = T_{i+1}/T_i - 1$, and $C_m/T_m = 2T_1/T_m - 1$. It follows that $\sum_{i=1}^{m} C_i/T_i = (\sum_{i=1}^{m-1} T_{i+1}/T_i) + 2T_1/T_m - m$. By a simple mathematical property, the minimal value of $((\sum_{i=1}^{m-1} T_{i+1}/T_i) + 2T_1/T_m)/m$ is equal to $((T_2/T_1)(T_3/T_2)\cdots(T_m/T_{m-1})(2T_1/T_m))^{\frac{1}{m}} = 2^{\frac{1}{m}}$. Therefore, the minimal value of $\sum_{i=1}^{m} C_i/T_i = m(2^{\frac{1}{m}} - 1)$. □

This indicates Lemma 4 is true under the assumption that any period is at least half of any other period. What remains to be shown is that the lemma is still true without the assumption. Let $q_i = \lfloor T_m/T_i \rfloor$ and $T'_i = q_i T_i$ for $1 \leq i \leq m$. According to this definition, it is easy to show the following.

LEMMA 9. *(1)* $q_i \geq 1$ *for any* $1 \leq i \leq m$.

(2) $\lceil T_m/T_i \rceil \leq q_i + 1$ *for any* $1 \leq i \leq m$.

(3) $0 < T'_i/T'_j < 2$ *for any* $1 \leq i, j \leq m$.

PROOF. Obvious and omitted. □

LEMMA 10. *Let* $q_i = \lfloor T_m/T_i \rfloor$, $T'_i = q_i T_i$ *for* $1 \leq i \leq m$, *and* $C'_i = C_i$ *for* $i = 1, \ldots, m-1$, $C'_m = C_m + \sum_{i=1}^{m-1}(q_i - 1)C_i$. *If* $non\_inc(C, T)$, *then* $non\_inc(C', T')$.

PROOF. We prove the lemma by contradiction.

$$\begin{aligned}
& \neg non_inc(C', T') \\
\Rightarrow\ & \textstyle\bigvee_{i=1}^{m} \left(\sum_{j=1}^{m} \lceil T'_i / T'_j \rceil C'_j < T'_i \right) && \{\text{Lemma 6}\} \\
\Rightarrow\ & \textstyle\bigvee_{i=1}^{m} \left(\sum_{T'_j < T'_i} 2C'_j + \sum_{T'_j \geq T'_i} C'_j < T'_i \right) \\
& && \{\text{for any } j \text{ such that } T'_j < T'_i,\ \lceil T'_i/T'_j \rceil = 2, \\
& && \text{for any } j \text{ such that } T'_j \geq T'_i,\ \lceil T'_i/T'_j \rceil = 1\} \\
\Rightarrow\ & \textstyle\bigvee_{i=1}^{m} \left(\sum_{T'_j < T'_i} (q_j + 1) C_j + \sum_{T'_j \geq T'_i} q_j C_j < T'_i \right) && \{\text{definition of } C'_m\} \\
\Rightarrow\ & \textstyle\bigvee_{i=1}^{m} \left(\sum_{j=1}^{m} \lceil T'_i / T_j \rceil C_j < T'_i \right) && \{\text{Lemma 9}\}
\end{aligned}$$

This is in contradiction to $non\_inc(C, T)$, and completes the proof. □

LEMMA 11. $\min\{\sum_{i=1}^{m} C_i/T_i \mid non\_inc(C,T) \wedge 0 < T_i/T_j < 2 \textit{ for all } 1 \leq i, j \leq m\}$
$= \min\{\sum_{i=1}^{m} C_i/T_i \mid non\_inc(C,T)\}$.

PROOF. Let

$$\begin{aligned}
\mathcal{S} &= \{\textstyle\sum_{i=1}^{m} C_i/T_i \mid non_inc(C,T)\}, \\
\mathcal{S}' &= \{\textstyle\sum_{i=1}^{m} C_i/T_i \mid non_inc(C,T) \wedge 0 < T_i/T_j < 2 \text{ for each } 1 \leq i, j \leq m\}.
\end{aligned}$$

Assume $U = \sum_{i=1}^{m} C_i/T_i \in \mathcal{S}$ and $non\_inc(C,T)$. Let $q_i = \lfloor T_m/Ti \rfloor, T'_i = q_i T_i, C'_i = C_i\ (i = 1, \cdots, m-1), T'_m = T_m, C'_m = C_m + \sum_{i=1}^{m-1} (q_i - 1) C_i$. It follows from Lemma 9 that $0 < T'_i/T'_j < 2$ and from Lemma 10 that $non\_inc(C', T')$. It is easy to prove

$$U' = \textstyle\sum_{i=1}^{m} C'_i/T'_i = U + \sum_{i=1}^{m-1} C_i (q_i - 1)(1/T_m - 1/T'_i) \leq U.$$

Therefore, min $\mathcal{S}' \leq$ min $\mathcal{S}$. On the other hand, it is easy to see min $\mathcal{S}' \geq$ min $\mathcal{S}$ since $\mathcal{S}' \subseteq \mathcal{S}$. Thus, min $\mathcal{S}' =$ min $\mathcal{S}$. □

Finally, Lemma 4 follows from Lemma 8 and Lemma 11.

In Liu and Layland's original proof for what we formalise as Lemma 7, they proposed two transformations. The transformations are not used in our proof of Lemma 7, since we have found a simpler proof. However, the transformation may be of interests for other reasons. The first transformation can be expressed by the following lemma

LEMMA 12. *Assume* $0 < T_i/T_j < 2$ *for all* $1 \leq i, j \leq m$, $C_1 = T_2 - T_1 + \Delta$, $\Delta > 0, C'_1 = T_2 - T_1, C'_2 = C_2 + \Delta, C'_i = C_i\ (i = 3, \cdots, m)$, *If* $non\_inc(C,T)$, *then* $non\_inc(C', T)$.

PROOF.

$$\begin{aligned}
\textstyle\sum_{i=1}^{m} \lceil T_1/T_i \rceil C'_i &= \textstyle\sum_{i=1}^{m} C'_i = \sum_{i=1}^{m} C_i \geq T_1 \\
\textstyle\sum_{i=1}^{m} \lceil T_2/T_i \rceil C'_i &= \textstyle\sum_{i=1}^{m} C'_i + C'_1 \geq T_1 + (T_2 - T_1) = T_2 \\
&\ \vdots \\
\textstyle\sum_{i=1}^{m} \lceil T_m/T_i \rceil C'_i &= \textstyle\sum_{i=1}^{m} \lceil T_m/T_i \rceil C_i \geq T_m && \{non_inc(C,T)\} \\
non_inc(C', T) & && \{\text{Lemma 6}\}
\end{aligned}$$

□

In the second transformation, Liu and Layland made a mistake. This was reported in [Dong *et al.* 1999,Devillers and Goossens 2000]. In [Dong *et al.* 1999], we gave the following corrected version

LEMMA 13. *Assume* $0 < T_i/T_j < 2$ *for all* $1 \le i, j \le m$, $C_1 = T_2 - T_1 - \Delta$, $\Delta > 0, C'_1 = T_2 - T_1, C'_i = C_i (i = 2 \cdots m), C'_m = C_m - 2\Delta$.[3] *If* $non\_inc(C, T)$, *then* $non\_inc(C', T)$.

PROOF.

$$
\begin{array}{lcll}
\sum_{i=1}^m \lceil T_1/T_i \rceil C'_i & = & \sum_{i=1}^m \lceil T_2/T_i \rceil C_i - C'_1 \ge T_2 - (T_2 - T_1) \; = \; T_1 & \\
\sum_{i=1}^m \lceil T_2/T_i \rceil C'_i & = & \sum_{i=1}^m \lceil T_2/T_i \rceil C_i \; \ge \; T_2 & \\
 & \vdots & & \\
\sum_{i=1}^m \lceil T_m/T_i \rceil C'_i & = & \sum_{i=1}^m \lceil T_m/T_i \rceil C_i \; \ge \; T_m & \{non\_inc(C, T)\} \\
non\_inc(C', T) & & & \{\text{Lemma 6}\}
\end{array}
$$

□

This transformation only makes sense when $C_m \ge 2\Delta$ and therefore cannot be applied in all cases. However, it is possible to prove Lemma 7 along the line as follows. First, we can extend Lemma 12 to all $i \le m - 1$ (consider Lemma 12 in the case $i = 1$), and by applying it repeatedly, we obtain $C_i \le T_{i+1} - T_i$ for all $i = 1, \ldots, m - 1$. Next, we can apply a new transformation indicated by the following lemma.

LEMMA 14. *Assume* $0 < T_i/T_j < 2$ *for all* $1 \le i, j \le m$, $C_i = T_{i+1} - T_i - \Delta_i, \Delta_i \ge 0, C'_i = T_{i+1} - T_i, i = 1 \ldots m-1, C'_m = C_m - 2(\Delta_1 + \cdots \Delta_{m-1})$. *If* $non\_inc(C, T)$, *then* $C'_m \ge 0$ *and* $non\_inc(C', T)$.

PROOF.

$$
\begin{array}{ll}
 & 2C_1 + 2C_2 + \cdots + 2C_{m-1} + C_m \ge T_m \qquad \{non\_inc(C, T)\} \\
\Rightarrow & 2(C'_1 - \Delta_1) + 2(C'_2 - \Delta_2) + \cdots + 2(C'_{m-1} - \Delta_{m-1}) + C_m \ge T_m \\
\Rightarrow & 2(C'_1 + C'_2 + \cdots + C'_{m-1}) - 2(\Delta_1 + \Delta_2 + \cdots + \Delta_{m-1}) + C_m \ge T_m \\
\Rightarrow & 2(T_m - T_1) - 2(\Delta_1 + \Delta_2 + \cdots + \Delta_{m-1}) + C_m \ge T_m \\
\Rightarrow & C_m - 2(\Delta_1 + \Delta_2 + \cdots + \Delta_{m-1}) \ge 2T_1 - T_m \\
\Rightarrow & C'_m \ge 2T_1 - T_m \\
\Rightarrow & C'_m \ge 0
\end{array}
$$

and

$$
\begin{array}{rcl}
C'_1 + C'_2 + \cdots + C'_{m-1} + C'_m & \ge & T_m - T_1 + 2T_1 - T_m = T_1 \\
 & \vdots & \\
2C'_1 + 2C'_2 + \cdots + 2C'_{j-1} + C'_j + \cdots + C'_m & \ge & T_m - T_1 + T_j - T_1 + 2T_1 - T_m = T_j \\
 & \vdots & \\
2C'_1 + 2C'_2 + \cdots + 2C'_{m-1} + C'_m & \ge & 2(T_m - T_1) + 2T_1 - T_m = T_m
\end{array}
$$

□

[3] In Liu and Layland's paper, they let $C'_2 = C_2 - 2\Delta$ and $C'_m = C_m$. A counterexample was given in [Devillers and Goossens 2000]

It is easy to prove that these transformations will not increase the utilisation factor, and from the proof of Lemma 13 that $C'_m \geq 2T_1 - T_m$, hence, Lemma 7 follows.