A Formal Specification of an Information Processing System in Duration Calculus

Sheila A. Karipel and Dang Van Hung
United Nations University
International Institute for Software Technology
UNU/IIST, P.O.Box 3058, Macau

ABSTRACT

In this paper, we give a formal specification of an Information Processing System (IPS) in Duration Calculus. The IPS we are dealing with is a hard real-time system consisting of two independent subsystems that communicate through several I/O channels with the external environment. It is stated formally that the requirement for the system is met if the earliest deadline first (EDF) scheduler is used and certain conditions on the parameters are satisfied.

Keywords: Formal Specification, Duration Calculus, Earlest Deadline Driven Schedule, Information Systems.

1. INTRODUCTION

Over the years, the rapidly developing technology has made it possible for information processing systems to handle and process substantially more and more information. Though the high speed technology reduced the computation time considerably, a more precise information extraction required the application of highly advanced information processing techniques. These sophisticated and computationally intensive techniques are extremely time-bounded within the systems. This called for an extensive analysis of the design of the systems.

As one of the authors has participated in the earlier development of an information processing system (IPS) which is a hard real-time system, it provided the authors an opportunity to take it up as a case study for demonstrating the powerful benefits of using Duration Calculus (DC) as a formalism for specifying and verifying a $real$ complex system.

Hard real-time systems have been defined as those containing processes that have deadlines which cannot be missed [1]. Such deadlines have been termed hard: they have to be met under all circumstances.

Meeting hard deadlines imposes constraints on the allocation of physical and logical resources of the system at runtime. Typically, the resources are allocated by a scheduling algorithm whose purpose is to interleave the executions of processes in the system to achieve a particular goal which in the case of hard real-time systems is that no deadline is missed.

In this paper, we attempt to specify an appropriate scheduler that controls the execution of the independent tasks and resources within a subsystem. With a formal specification of IPS and the schedulers it used, we are able to verify that the system task set will meet the required deadlines.

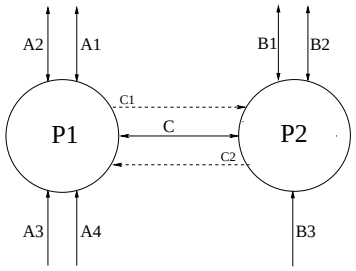

Figure 1. The Information Processing System

The IPS

The IPS as shown in figure 1 is an integration of 2 subsystems viz. P1 and P2 with identical hardware. Each subsystem has a single processing unit and up to 5 I/O channels (shown as bold directed lines). The data is exchanged in the form of "packets" between the environment and the processors at regular intervals of time. The data rates are different for each channel.

The channel C is a bi-directional communication link between the subsystems P1 and P2. However, for convenience, this bi-directional channel has been logically split into two uni-directional communication links, $C1$ and $C2$. P1 transmits through $C1$ and receives through $C2$ while P2 receives through $C1$ and transmits through $C2$.

The explanations that follow are valid only after the processors P1 and P2 have been initialised. It is assumed that the buffers mentioned below can hold only one packet at a time.

Channels and Processors

The processor P1 uses all the 5 I/O channels viz. $A1, A2, A3, A4,$ and C ($C1/C2$), out of which 2 channels ($A3$ and $A4$) are used as dedicated input channels. $A1$, $A2$, $A3$ and $A4$ are communication links between P1 and the external environment while C ($C1/C2$) is the common link between the processors P1 and P2. The processor P2 uses only 4 I/O channels ($B1$, $B2$, $B3$, and C) out of which 1 channel ($B3$) is used as a dedicated input channel. $B1$, $B2$, and $B3$ are communication links between P2 and the external environment.

The channels $A1$ and $A2$: The data packets arriving on these channels contain queries regarding the health status of the processor P1.

Once P1 is initialised, it receives a data packet on Ai from the

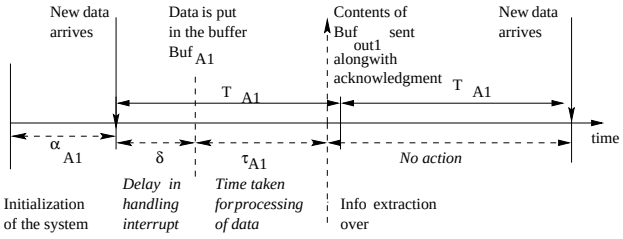

Figure 2. The Channel $A1$ of subsystem P1

external environment for $i = 1, 2$. The interrupt handler for Ai transfers the data to buffer Buf_{Ai}. The time taken for handling the interrupt is at most δ seconds which is the delay incurred due to the processor handling other interrupt(s).

P1 extracts the information from the data packet received, checks for presence of data in the buffer Buf_{outi}, and sends out an acknowledgement along with the (non-empty) contents of Buf_{outi}. All these are done in τ_{Ai} seconds such that $\delta + \tau_{Ai} < T_{Ai}$ where $2 * T_{Ai}$ is the periodicity of arrival of data on Ai.

The subsystem P1 alternately receives from and transmits to the external environment through Ai in the manner described above. The behaviour of the channel A1 is depicted as the diagram in Fig 2.

The channels $A3$ and $A4$: The data packets received on this dedicated input channel Ai, $i = 3, 4$ of processor P1 contain information I_i. Once P1 is initialised, it receives a data packet on Ai from the external environment. The interrupt handler for Ai transfers the data to buffer Buf_{Ai}. The time taken for handling the interrupt is at most δ seconds which is the delay incurred due to the processor handling other interrupt(s).

P1 extracts the information contained in the data packet received, processes it, and puts the result I_i' in the buffer Buf_{P1}. All these are done in τ_{Ai} seconds such that $\delta + \tau_{Ai} < T_{Ai}$ where T_{Ai} is the periodicity of arrival of data on Ai.

The channel $C1$ and $C2$ (C): These are unidirectional communication channels through which Pi receives data from and transmits data to Pj for $j \neq i$ and $i, j \in \{1, 2\}$. The data packets received by P1 on the channel $C1$ contain information, I_{out} (transmitted by P2 as contents of buffer Buf_{P2}). The data packets received by P2 on the channel $C2$ are the processed information I_1' or I_2' (transmitted by P1 as contents of buffer Buf_{P1}).

Once P1 is initialised, it receives a data packet on $C1$ from P2.

Once P2 is initialised, it transmits the contents of buffer Buf_{P2} (which contains dummy data initially) to P1 through C1. The information thus sent to P1 is to be forwarded to the external environment by P1 later.

The interrupt handler for $C1$ transfers the data to buffer Buf_{C1}. When the interrupt is raised due to the arrival of a data packet from P1, the interrupt handler for $C2$ transfers the data to buffer Buf_{C2}.

The time taken for handling the interrupt is at most δ seconds which is the delay incurred due to the processor handling other interrupt(s).

The task associated with this channel Ci is to extract the information contained in the data packet received, and store the results of the computation in the buffer Buf_{outi}. This is done in τ_{Ci} seconds such that $\delta + \tau_{Ci} < T_{Pi}$ where $T_{P1} + T_{P2} = T_{C1} = T_{C2}$ is the periodicity of reception of data by Pi on Ci.

The channels $B1$ and $B2$: The data packets arriving on these channels contain queries regarding the health status of the processor P2. Once P2 is initialised, it receives a data packet on Bi ($i = 1, 2$) from the external environment. The interrupt handler for Bi transfers the data to buffer Buf_{Bi}. The time taken for handling the interrupt is at most δ seconds which is the delay incurred due to the processor handling other interrupt(s).

P2 extracts the information from the data packet received and sends out an acknowledgement. All these are done in τ_{Bi} seconds such that $\delta + \tau_{Bi} < T_{Bi}$ where $2 * T_{Bi}$ is the periodicity of arrival of data on Bi.

The subsystem P2 alternately receives from and transmits to the external environment through Bi in the manner described above.

The channel $B3$: The data packets received on this dedicated input channel of processor P2 contain information, I_3.

Once P2 is initialised, it receives a data packet on B3 from the external environment. The interrupt handler for B3 transfers the data to buffer Buf_{B3}. The time taken for handling the interrupt is at most δ seconds which is the delay incurred due to the processor handling other interrupt(s).

P2 extracts the information, I_3 from the contents of buffer Buf_{B3}, processes it along with the contents of Buf_{out2} (which contains the information extracted by P2 from the data arriving on channel C2), and puts the result, I_{out}, in the buffer Buf_{P2}. All these are done in τ_{B3} seconds such that $\delta + \tau_{B3} < T_{B3}$ where T_{B3} is the periodicity of arrival of data on B3.

2. DURATION CALCULUS: A BRIEF SUMMARY

In this section, we give a brief summary of Duration Calculus which will be used as the formalism to specify the IPS in this paper. For more details, readers are referred to [4].

Time in DC is the set R^+ of non-negative real numbers. For $t, t' \in R^+, t \leq t', [t, t']$ denotes the time interval from t to t'.

We assume a finite set E of Boolean variables called primitive states. E includes the Boolean constants 0 and 1 denoting *false* and *true* respectively. States, denoted by P, Q, P_1, Q_1, etc., consist of Boolean expressions over E. A primitive state P is interpreted as a function $I(P) : R^+ \rightarrow \{0, 1\}$. $I(P)(t) = 1$ means that state P is present at time instant t, and $I(P)(t) = 0$ means that state P is not present at time instant t. We assume that a state has finite variability in a finite time interval. A composite state is interpreted as a function which is defined by the interpretations for the primitive states and Boolean operators.

For an arbitrary state P, its duration is denoted by $\int P$. Given an interpretation I of states and an interval, duration $\int P$ is interpreted as the accumulated length of time within the interval at which P is present. So for an arbitrary interval $[t, t']$, the interpretation $I(\int P)([t, t'])$ is defined as $\int_t^{t'} I(P)(t)dt$. Therefore, $\int 1$ always gives the length of the intervals and is denoted by ℓ.

The set of primitive duration terms consists of variables over the set R^+ of non-negative real numbers and durations of states. In

this paper, a duration term is defined either as a primitive term or as a linear combination of primitive terms.

A primitive duration formula is an expression formed from terms by using the usual relational operations on the reals, such as equality = and inequality $<$. A duration formula is either a primitive formula or an expression formed from formulas by using the logical operators $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, and the chop ; (see below) and quantifiers $\forall$, $\exists$ applied to variables ranging over R^+.

A duration formula D is satisfied by an interpretation I in an interval $[t', t'']$ just when it evaluates to true for that interpretation over that time interval. This is written as

$$I, [t', t''] \models D,$$

where I assigns every primitive state a finitely variable function from R^+ to {0,1}, and $[t', t'']$ decides the observation window. So the satisfaction relation has nothing to do with the values of the primitive state assigned by I outside the observation window $[t', t'']$. That is, for interpretations I and I', if

$$I(P)(t) = I'(P)(t),\ t' \leq t \leq t''$$

holds for all primitive states in D, then we can prove

$$I, [t', t''] \models D \text{ iff } I', [t', t''] \models D.$$

Given an interpretation I, the chop-formula $D_1; D_2$ is true for $[t', t'']$ if there exists a t such that $t' \leq t \leq t''$ and D_1 and D_2 are true for $[t', t]$ and $[t, t'']$ respectively.

We give now shorthands for some duration formulas which are often used. For an arbitrary state P, $\lceil P \rceil$ stands for $(\int P = \ell) \wedge (\ell > 0)$. This means that P holds everywhere in a non-point interval. We use $\lceil\ \rceil$ to denote the predicate which is true only for point intervals. Modalities $\Diamond, \Box$ are defined as: $\Diamond D = true; D; true,\ \Box D = \neg\Diamond\neg D$. This means that $\Diamond D$ is true for an interval iff D holds for some subinterval of it, and $\Box D$ is true for an interval iff D holds for all subintervals of it.

In this paper, we will use the following abbreviations as well.

$$\Diamond D \quad \widehat{=} \quad (D; true) \qquad \boxdot D \quad \widehat{=} \quad \neg\Diamond\neg D$$

$\Diamond D$ holds for an interval $[a, b]$ if and only if D holds for some prefix $[a, c]$ ($c \leq b$) of the interval, and $\boxdot D$ holds for an interval if and only if D holds for any prefix of the interval.

DC has a set of axioms about states and rules which is sound and (relatively) complete. The readers are referred to [4] for the proof system of DC.

3. FORMAL SPECIFICATION

In order to write down a formal specification of the IPS, we segregate the $fixed$ properties and the $tunable$ properties of the IPS. The fixed properties refer to the inherent behaviour of the system viz. the effects of the hardware components used. We refer to these $hard$ features of the system as the $behavior$ of the IPS. The tunable properties, as the name implies, refer to the behaviour of the software which can be *tuned* to meet the requirements most effectively. Hence, the reference to the characteristics of the software processes as the scheduling algorithm. Therefore, it is expected that the behaviour of IPS that controlled by the scheduler will meet the *requirements* of the IPS.

Our purpose is to write the DC formulas

1. $\Im^{env}$, to capture the behaviour of the IPS,
2. $\Im^{req}$, to specify the requirements, and
3. $\Im^{sch}$, to formalise the working of the scheduling algorithm

that are valid for any time interval of the form $[0, t]$.

Specification of the Behaviour of IPS

The behaviour of the IPS is modelled by defining the states of its subsystems, P1 and P2.

Definition of States: Let us define the sets $Ch_{P1} = \{A1, A2, A3, A4, C1\}$ and $Ch_{P2} = \{B1, B2, B3, C2\}$. For each channel $i \in Ch_m$ and $m \in \{P1, P2\}$, we introduce two state variables viz.

1. $intr_i^m \in Time \rightarrow \{0, 1\}$ to model the interrupt caused by the arrival of new data on channel i in subsystem m. For any time t, $intr_i^m(t) = 1$ means that at time t an interrupt has been raised by the arrival of data on channel i and the processor (m) has been requested for servicing it, but the data has not been captured into buffer Buf_i^m. Therefore, if the data on channel i has arrived, $intr_i^m(t) = 0$ means that the interrupt on channel i has been serviced and the data has been flushed into Buf_i^m. It is to be noted that Buf_i^m can hold only one data packet at a time. Thus, when a new data packet is flushed into Buf_i^m, the old data packet is overwritten.
2. $proc_i^m \in Time \rightarrow \{0, 1\}$ to model the processing of the contents of buffer Buf_i^m by the processor m. For any time t, $proc_i^m(t) = 1$ means that the data in Buf_i^m is being processed by the processor, and $proc_i^m(t) = 0$ means that the data in Buf_i^m is not being processed by the processor. Note that the precessing of the data can be preempted (it is the scheduler who decides the data in the buffer of which channel is processed by the processor).

Definition of the Periodic Intervals: For any channel i of subsystem m, the periodic interval is the time interval starting at $\alpha_i^m + k * T_i^m$ and ending at $\alpha_i^m + (k+1) * T_i^m$, where α_i^m is the arrival time of the first data packet, $k = 0, 1, 2, \ldots$, and T_i^m is the periodicity of arrival of data. Thus, a periodic interval is a time interval between two consecutive arrivals of data on a channel.

For any channel i of subsystem m, the interval $[0, t]$ is always expressed by

$$\begin{array}{l}(\ell \leq \alpha_i^m) \vee (\ell = \alpha_i^m \bmod T_i^m\ ;\ \ell = T_i^m) \\ \vee\ (\ell = \alpha_i^m \bmod T_i^m\ ;\ 0 < \ell < T_i^m)\end{array} \tag{1}$$

That is, for any t either there is no arrival of data before t, or else, the suffix of the interval $[0, t]$ from the last data arrival before t is a prefix of a periodic interval.

Properties of the IPS: A data packet on channel $i \in Ch_m \wedge m \in \{P1, P2\}$ raises an interrupt on processor m at the beginning of the periodic interval T_i^m. After δ (where $0 \leq \delta < T_i^m$) time units, the processor completes the handling of the interrupt by flushing the data packet into the buffer Buf_i^m. Therefore, every k^{th} periodic interval satisfies the constraint [1] :

$$\lceil intr_i^m \rceil\ ;\ \lceil \neg\, intr_i^m \rceil \tag{2}$$

[1] Note that when $\neg$, $\Box$, and $\Diamond$ occur in formulas, they have higher precedence than the binary connectives and the modality ;.

and since that it takes at most δ time units to handle an interrupt, every time interval satisfies the condition :

$$\Box \lceil intr_i^m \rceil \Rightarrow \ell \leq \delta \quad (3)$$

When a data packet arrives on channel $i \in Ch_m$ of processor $m \in \{P1, P2\}$, it can be processed only if it has been flushed into the buffer Buf_i^m. So the processing of Buf_i^m can take place only after the interrupt caused by the data packet has been handled by processor m. Therefore, for any time interval

$$\Box \lceil intr_i^m \rceil \Rightarrow \lceil \neg\, proc_i^m \rceil \quad (4)$$

The processor $m \in \{P1, P2\}$ handles the processing of the data in the buffers $Buf_i^m,\ i \in Ch_m$ with the help of the scheduler implemented in the system. However, at a time, the processor can process the data of at most one buffer. So, for $j \in Ch_m \wedge j \neq i$,

$$\Box \lceil proc_i^m \rceil \Rightarrow \lceil \neg\, proc_j^m \rceil \quad (5)$$

It was mentioned earlier that P1 and P2 are identical systems, and that channel C (referred to as C1 and C2 in the context) is the common link between the two systems. It can be seen that T_{C1} and T_{C2} are each equal to $T_{P1} + T_{P2}$. Therefore, $T_{C1} = T_{C2}$. From the description, it is evident that $\alpha_{C2} = \alpha_{C1} + T_{P1}$. This being a property of the IPS, we have

$$(T_{C1} = T_{C2}) \wedge (\alpha_{C2} = \alpha_{C1} + T_{P1}) \quad (6)$$

Combining the properties of the IPS, we get the specification of the behaviour of IPS during the time interval $[0, t]$ as in Fig. 3.

Specification of the Requirements of IPS

Let

$$\begin{array}{ll} Bi\_ch_{P1} = \{A1, A2\}, & Bi\_ch_{P2} = \{B1, B2\} \\ Uni\_ch_{P1} = \{A3, A4\}, & Uni\_ch_{P2} = \{B3\} \\ Int\_ch_{P1} = \{C1\}, & Int\_ch_{P2} = \{C2\} \end{array}$$

denote the bi-directional, uni-directional, and inter-link channels of the subsystems P1 and P2.

As described earlier, there is a task associated with each channel in a subsystem. The execution of the task should be completed before the next data packet arrives on the associated channel. In other words, the execution of the task should be completed within the periodic interval associated with the channel. The periodic interval has already been defined in Section 1.

For the bi-directional channels, the processing of data should be completed before half the duration of the periodic interval T_i^m is over. This means that every periodic interval of channel $i \in Bi\_ch_m \wedge m \in \{P1, P2\}$ should satisfy

$$\int proc_i^m = \tau_i^m\ ;\ \ell = 0.5 * T_i^m \quad (8)$$

For the uni-directional channel $i \in Uni\_ch_m \wedge m \in \{P1, P2\}$, every periodic interval should satisfy

$$\int proc_i^m = \tau_i^m \quad (9)$$

For the inter-link channel $i \in Int\_ch_m \wedge m \in \{P1, P2\}$, every periodic interval should satisfy

$$\int proc_i^m = \tau_i^m\ ;\ \ell = T_i^m - T_m \quad (10)$$

By putting together the requirement for the periodic intervals for different channels (8),(9),(10) and taking into account the representation of the intervals of the form $[0, t]$ via periodic intervals (1), we get the specifications of the requirements to be met by the IPS for the time interval $[0, t]$ as in Figure 4

Specification of the Scheduler in the IPS

The IPS is a hard real-time system integrating two independent subsystems whose behaviour and requirements have been specified in the earlier sections. The existence of an EDF scheduler as an integral part of the operating system within each subsystem is also assumed. Each subsystem has a single processing unit for carrying out the task of processing the data arriving on its multiple I/O channels. The periodicity of arrival of data packets, the computation time required for processing the data in the buffer, and the deadline for completion of the computation task vary with each channel within a subsystem.

Consider channel i of processor m. A data packet arrives at time $\alpha_i^m + k * T_i^m$ time units where $k = 0, 1, 2, \ldots$. It takes at most δ time units for the newly arrived data to be flushed into buffer Buf_i^m. So, in the worst case, the task becomes ready only after $\alpha_i^m + k * T_i^m + \delta$ time units. Therefore, in the worst case, the periodic interval for the task starts from $\alpha_i^m + k * T_i^m + \delta$ and ends at $\alpha_i^m + (k+1) * T_i^m + \delta$ time units.

After a task becomes ready, it requires a computation time of τ_i^m time units to process the data in Buf_i^m. The accumulated run time of the task should not exceed its deadline D_i^m which is less than its period T_i^m. The deadline for a task varies with the channel associated with the task as:

$$\text{for } i \in Bi\_ch_m, m \in \{P1, P2\} \quad D_i^m = 0.5 * T_i^m \quad (12)$$

$$\text{for } i \in Uni\_ch_m, m \in \{P1, P2\} \quad D_i^m = T_i^m \quad (13)$$

$$\text{for } i \in Int\_ch_m, m \in \{P1, P2\} \quad D_i^m = T_m \quad (14)$$

The Scheduling Policy: The tasks to be scheduled of course should satisfy

$$\tau_i^m \ \leq\ D_i^m \ \leq\ T_i^m - \delta \quad (15)$$

and also by an offset of $\alpha_i^m + \delta$. Leung and Whitehead [5] have defined a deadline monotonic priority assignment that caters for tasks with the time constraint (15). Using the results of Leung and Whitehead, Audsley [2] established schedulability tests for periodic tasks with multiple release times.

The tasks are executed in a preemptive manner: at any instant, the task which has the nearest deadline for completion of its computation is allocated processor time. In literature, this scheduling mechanism is referred to as the earliest deadline first (EDF) scheduler.

Audsley's tests guarantee the deadlines of periodic tasks which satisfy the timing constraint (15).

For a given set of tasks, ($i \in$) Ch_m in a processing unit m, the deadline monotonic scheduling is feasible if and only if

$$\sum_i \frac{\tau_i^m + \sum_{j \wedge D_j^m \leq D_i^m \wedge j \neq i} \lceil \frac{D_i^m}{T_j^m} \rceil * \tau_j^m}{D_i^m} \leq 1 \quad (16)$$

$$
\Im^{env} = \bigwedge_{\substack{i \in Ch_m, \\ m \in \{P1,P2\}}} \left(\begin{array}{c} (\Box \, [\![intr_i^m]\!] \Rightarrow \ell \leq \delta) \wedge \\ (\Box \, [\![intr_i^m]\!] \Rightarrow [\![\neg \, proc_i^m]\!]) \wedge \\ ((j \in Ch_m \wedge j \neq i) \Rightarrow (\Box \, [\![proc_i^m]\!] \Rightarrow [\![\neg \, proc_j^m]\!])) \wedge \\ \left(\begin{array}{c} (\ell \leq \alpha_i^m) \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} \ell = T_i^m \\ \wedge \\ [\![intr_i^m]\!] \; ; \; [\![\neg \, intr_i^m]\!] \end{array} \right)) \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} 0 < \ell < T_i^m \\ \wedge \\ [\![intr_i^m]\!] \; ; \; ([\![\neg \, intr_i^m]\!] \; \vee \; \lceil \, \rceil) \end{array} \right)) \end{array} \right) \end{array} \right) \quad (7)
$$

$$
\wedge
$$

$$
(T_{C1}^{P1} = T_{C2}^{P2}) \wedge (\alpha_{C2} = \alpha_{C1} + T_{P1})
$$

Figure 3. Specification of the properties of the IPS

$$
\Im^{req} = \begin{array}{l} \bigwedge_{i \in Bi\_ch_m \wedge m \in \{P1,P2\}} \left(\begin{array}{c} (\ell \leq \alpha_i^m) \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} (\ell = T_i^m) \wedge \\ (\int proc_i^m = \tau_i^m \; ; \; \ell = 0.5 * T_i^m) \end{array} \right)) \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} \left(\begin{array}{c} (0.5 * T_i^m \leq l < T_i^m) \wedge \\ (\int proc_i^m = \tau_i^m \; ; \; \ell \geq 0) \end{array} \right) \\ \vee \\ \left(\begin{array}{c} (0 < \ell < 0.5 * T_i^m) \wedge \\ (\int proc_i^m \leq \tau_i^m) \end{array} \right) \end{array} \right)) \end{array} \right) \\ \wedge \\ \bigwedge_{i \in Uni\_ch_m \wedge m \in \{P1,P2\}} \left(\begin{array}{c} \ell \leq \alpha_i^m \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} \ell = T_i^m \wedge \\ (\int proc_i^m = \tau_i^m) \end{array} \right)) \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} 0 < \ell < T_i^m \wedge \\ \int proc_i^m \leq \tau_i^m \end{array} \right)) \end{array} \right) \\ \wedge \\ \bigwedge_{\substack{i \in Int\_ch_m \\ \wedge \, m \in \{P1,P2\}}} \left(\begin{array}{c} \ell \leq \alpha_i^m \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} \ell = T_i^m \wedge \\ (\int proc_i^m = \tau_i^m \; ; \; \ell = T_i^m - T_m) \end{array} \right)) \vee \\ (\ell = \alpha_i^m \bmod T_i^m \; ; \left(\begin{array}{c} \left(\begin{array}{c} T_m \leq l < T_i^m \wedge \\ (\int proc_i^m = \tau_i^m \; ; \; \ell \geq 0) \end{array} \right) \\ \vee \\ \left(\begin{array}{c} 0 < \ell < T_m \wedge \\ \int proc_i^m \leq \tau_i^m \end{array} \right) \end{array} \right)) \end{array} \right) \end{array} \quad (11)
$$

Figure 4. Specification of the requirements of IPS

τ_i^m, D_i^m, and T_i^m are reasonably assumed to be integral multiples of machine cycles.

Properties of the EDF Scheduler: To aid the formal verification of the IPS, the behaviour of the scheduler is described in DC which enables us to abstract the accumulated run time of tasks. The main strategy of the EDF scheduler is that in any time interval, one of the most urgent tasks (i.e. one with the nearest deadline) will occupy the processor. Less urgent ones should be kept waiting or be preempted by more urgent ones.

At time t, the length of time elapsed is $\ell = \alpha_i^m + n * T_i^m + \theta_i^m$ where $0 \leq \theta_i^m < T_i^m$ and $n = 0, 1, 2, \ldots$. The remaining time, $rem\_time_i^m$, with regard to the currently approaching deadline depends on the value of θ_i^m and is computed as follows:

$$rem\_time_i^m = D_i^m - \theta_i^m \text{ if } 0 \leq \theta_i^m < D_i^m \text{ or}$$
$$rem\_time_i^m = D_i^m + T_i^m - \theta_i^m \text{ if } D_i^m \leq \theta_i^m < T_i^m$$

So, for any two tasks $proc_i^m$ *and* $proc_j^m$ where $i, j \in Ch_m \wedge i \neq j \wedge m \in \{P1, P2\}$, if $rem\_time_i^m < rem\_time_j^m$, then $proc_i^m$ is more urgent than $proc_j^m$ because t is closer to the deadline of $proc_i^m$ than the deadline of $proc_j^m$. This fact is represented by the DC formula $Urg^m(i,j)$ which holds for the interval $[0,t]$ iff at time t $proc_i^m$ is *more urgent* than $proc_j^m$,

$$Urg^m(i,j) \widehat{=} \quad \left(\left\lfloor \frac{l-\alpha_i-D_i}{T_i} \right\rfloor + 1\right) * T_i + \alpha_i + D_i < \left(\left\lfloor \frac{l-\alpha_j-D_j}{T_j} \right\rfloor + 1\right) * T_j + \alpha_j + D_j$$

In order to express that $proc_i^m$ has not been finished at time t, we introduce a DC formula $under_i^m$,

$$under_i^m \widehat{=} \quad \ell \leq \alpha_i \vee (\ell = \alpha_i \bmod T_i^m; \left(\begin{array}{c} \ell \leq T_i^m \wedge \\ \int proc_i^m < \tau_j^m)) \end{array} \right).$$

So, if $under_i^m$ holds for the interval $[0,t]$, the processing of the last arrived data packet at channel i has not finished at time t.

Suppose $proc_i^m$ is currently running on processor m, then $proc_i^m$ must be the *most urgent* among all the other incomplete tasks on processor m before it is selected to run. In other words, at no time during the run can there be an incident of $proc_i^m$ being selected to run when another task has been found to be the most urgent. This is expressed as

$$\bigwedge_{j \in Ch_m} \neg\Diamond \left((Urg^m(j,i) \; ; \; \lceil proc_i^m \rceil) \wedge under_j^m \right) \qquad (17)$$

Assuming there is no overhead in the scheduling, an incomplete task implies that the processor is busy running some task and is not idle.

$$\boxdot under_i^m \Rightarrow \bigvee_{j \in Ch_m} true; \lceil proc_j^m \rceil \qquad (18)$$

Combining equation (17) and equation (18) for all the tasks in a processor, we get the specification of the EDF scheduler as

$$\Im^{sch} = \bigwedge_{\substack{i \in Ch_m \wedge \\ m \in \{P1,P2\}}} \mathcal{A}(i,m) \qquad (19)$$

where

$$\mathcal{A}(i,m) = \left(\begin{array}{c} \bigwedge_{j \in Ch_m} \neg\Diamond \left(\begin{array}{c} (Urg^m(j,i); \lceil proc_i^m \rceil) \\ \wedge \; under_j^m \end{array} \right) \\ \wedge \\ \boxdot under_i^m \Rightarrow \bigvee_{j \in Ch_m} (true; \lceil proc_j^m \rceil) \end{array} \right)$$

Formal Verification

If the specification of the behaviour of the IPS and that of the scheduler in the IPS hold in any time interval $[0,t]$, and the sufficient condition 16 is satisfied, then the requirements is met:

Theorem 1 *It is proved that*

$$\vdash \left(\begin{array}{c} \Im^{env} \wedge \Im^{sch} \wedge \\ \sum_i \frac{\tau_i^m + \sum_{j \wedge D_j^m \leq D_i^m \wedge j \neq i} \lceil \frac{D_i^m}{T_j^m} \rceil * \tau_j^m}{D_i^m} \leq 1 \end{array} \right) \Rightarrow \Im^{req}$$

The formal proof of the theorem using the proof system of Duration Calculus is similar to that of Liu Layland's theorem in [7], and is not presented here.

4. CONCLUSION

This paper sets out the formal specifications of the inherent behaviour of the IPS, the properties of the EDF scheduler, and the requirements of the implementation of the IPS that was taken up for as a case study in DC. By formalising the system that the first author was involved in the design, we have a clear understanding its implementation. Besides, we are able to verify the system formally. The next step of our work will be the formal proof of the correctness of the system using DC proof assistant in PVS.

References

[1] N. Audsley, A. Burns, *Real-Time System Scheduling*, Technical Report YCS 134, University of York, UK.

[2] Neil C. Audsley, *Deadline Monotonic Scheduling*, Technical Report YCS 146, University of York, UK, 9/90.

[3] Philip Chan, Dang Van Hung, *Duration Calculus Specification of Scheduling for Tasks with Shared Resources*, LNCS 1023, Springer-Verlag 1995, pp. 365–380.

[4] Michael R. Hansen, Zhou Chaochen, *Duration Calculus: Logical Foundations*, Formal Aspects of Computing, 9(3): 283-330, 1997.

[5] J. Leung, J. Whitehead, *On the Complexity of Fixed Priority Scheduling of Real-Time Tasks*, Performance Evaluation 2(4): 237-250, 1982.

[6] Zhiming Liu, Mathai Joseph, Tomasz Janowski, *Verification of Schedulability for Real-Time Programs*, Formal Aspects of Computing, 7(5): 510-532, 1995

[7] Zheng Yuhua, Zhou Chaochen, *A Formal Proof of the Deadline Driven Scheduler*, Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863, pp. 756-775, Springer-Verlag, 1994.