An Intuitive Formal Proof for Deadline Driven Scheduler

ZHAN Naijun

Laboratory of Computer Science and Technology, Institute of Software
The Chinese Academy of Sciences, Beijing 100080, P.R. China
E-mail: znj@ios.ac.cn
Received September 20, 1999; revised March 7, 2000.

Abstract This paper presents another formal proof for the correctness of the Deadline Driven Scheduler (DDS). This proof is given in terms of Duration Calculus which provides abstraction for random preemption of processor. Compared with other approaches, this proof relies on many intuitive facts. Therefore this proof is more intuitive, while it is still formal.

Keywords duration calculus, deadline driven scheduler, real-time

1 Introduction

DDS is used for scheduling multiple tasks on a single processor. It is proposed by Liu and Layland[1]. It assumes that all tasks raise periodic requests for processor time, and priorities are dynamically assigned to tasks according to the deadlines of their current requests. A task will be assigned the highest priority if the deadline of its current request is the nearest, and will be assigned the lowest priority if the deadline of its current request is the furthest. At any instant, the task with the highest priority and yet unfulfilled request will occupy the processor. So using DDS, full processor utilisation can be achieved by dynamically assigning priorities on the basis of their current deadlines.

[1] establishes a necessary and sufficient condition for DDS when it serves multiple periodic tasks running on a single processor. The condition is described in the following theorem.

Theorem (Liu/Layland). For a given set of m tasks, the deadline driven scheduling algorithm is feasible if and only if:

\[(C_1/T_1) + \cdots + (C_m/T_m) \leq 1 \quad (0 < C_i < T_i)\]

where \(C_i\) is the run time of the task \(p_i\), and \(T_i\) is the request period of \(p_i\) (assume \(T_i\) for any \(i \in \alpha\) to be positive integer).

[1] gives an informal proof for the theorem. The necessary part of the theorem is very clear, but its sufficient part is far from obvious. [2] gives a formal description and proof for DDS by using DC. This is the first formal proof for DDS in the literature. However, it applies heavily the induction rule of DC, and makes proof details less intuitive and harder to follow.

In this paper, we give a more intuitive but still formal proof for DDS in terms of DC. The main idea behind our proof is similar to [1]. We prove the sufficiency through three steps: firstly, to prove that if the processor has no idle time on an interval, then the requirement holds for it; secondly, to prove that, for a given interval, if the requirement holds for every proper prefix of the interval but not for itself, then the processor has no idle time on the interval; thirdly, to prove that if the processor has no idle time on any interval, then the requirement holds for it.
interval; finally, to prove that provided the requirement does not hold for an interval, then there must exist a prefix of the interval such that the requirement holds for all proper sub-prefixes of the prefix but not for the prefix itself. The last one contradicts the first two conclusions. So the sufficiency can be proved.

The remaining sections are organised as follows. Section 2 introduces preliminary knowledge for formal description and proof for DDS. Section 3 presents a formal specification for DDS. Section 4 gives a formal proof of Liu/Layland theorem. Lastly a brief conclusion is provided in Section 5.

2 Preliminaries

In this section, we shall give a brief review of DC and list some DC theorems which will be used later.

DC is an extension of real arithmetic and interval temporal logic to reason about real-time hybrid systems which is proposed by Zhou, Hoare and Ravn. Readers can refer to [5] for a detailed and comprehensive introduction to the duration calculus.

2.1 The Syntax and Semantics of DC

Let \( X, X_1, X_2, \ldots \) be propositional temporal letters, which are interpreted as Boolean-valued functions over time intervals. Let \( P, P_1, P_2, \ldots \) be state variables, which are interpreted as Boolean-valued functions over Time. Let \( x, y, z, \ldots \) be global variables, which can be interpreted as values of real numbers. Let \( f, f_1, \ldots \) be global function symbols, \( R, R_1, R_2, \ldots \) be global relation symbols. Their meanings are standard.

The terms of DC are defined inductively as follows:

\[
\theta ::= x | \ell | \int S[f(\theta_1, \ldots, \theta_n)]
\]

where \( f \) is an \( n \)-ary function symbol, \( \int S \) is the integral value of \( S \) in the interval, \( S \) is a state expression defined as:

\[
S ::= 0 \mid 1 \mid P \mid S_1 \lor S_2 \mid \neg S
\]

Terms are interpreted as real-valued functions over time intervals, where \( \int S \) over an interval \([b, e]\) is interpreted as the integral of Boolean-valued function \( S \) over \([b, e]\).

The atomic formulas of DC are defined as follows:

\[
\text{Atom} ::= \text{true} | X | R(\theta_1, \ldots, \theta_n)
\]

where \( R \) is an \( n \)-ary relation symbol, and \( \theta_1, \ldots, \theta_n \) are terms.

The formulas of DC are defined inductively as follows:

\[
\phi ::= \text{Atom} | \neg \phi \lor \psi \lor (\phi \land \psi) \lor \exists x. \phi
\]

Formulas are interpreted as truth functions over time intervals. Given an interpretation \( \mathcal{I} \) and an interval \([b, e]\), \( \mathcal{I}, [b, e] \models \phi \iff \mathcal{I}, [b, m] \models \phi \) and \( \mathcal{I}, [m, e] \models \psi \) for some \( m \in [b, e] \).

The following abbreviations will be used:

- \( \Diamond \phi \equiv \text{true} \land (\phi \land \text{true}) \) reads: "for some sub-interval: \( \phi \)"
- \( \Box \phi \equiv \neg (\neg \phi) \) reads: "for all sub-intervals: \( \phi \)"
- \( \Diamond_p \phi \equiv \phi \land \text{true} \) reads: "for some prefix: \( \phi \)"
- \( \Box_p \phi \equiv \neg \phi \land \text{true} \) reads: "for all prefixes: \( \phi \)"
- \( \Diamond_{kp} \phi \equiv \phi \land \ell > 0 \) reads: "for some proper prefix: \( \phi \)"
- \( \Box_{kp} \phi \equiv \neg \phi \land \ell = 0 \) reads: "for all proper prefixes: \( \phi \)"
- \( \Box \ell = 0 \) reads: "for all proper prefixes: \( \phi \)"

\[
[S] \equiv \int S = \ell \land \ell > 0
\]