Journal of Systems Architecture 63 (2016) 48–60

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Definitions of predictability for Cyber Physical Systems

Beilei Sun, Xi Li*, Bo Wan, Chao Wang, Xuehai Zhou, Xianglan Chen

School of Computer Science, University of Science and Technology of China, Room 421, Electronic Building 3rd, West Campus USTC, Hefei 215123, China

ARTICLE INFO

Article history:
Received 4 July 2015
Revised 26 November 2015
Accepted 25 January 2016
Available online 6 February 2016

Keywords:
Cyber Physical Systems
Predictability
I/O behavior predictability
Time predictability
Order predictability

ABSTRACT

With the recent proliferation of different types of Cyber Physical Systems (CPS), it is critically important to investigate the predictability of such systems. Along with functional correctness of the components, these systems must also ensure that timing and delay constraints of components are properly for the entire system to behave in a predictable manner in presence of various kinds of uncertainties. While the functional correctness of the CPS components has been investigated in the past, very little is available about the timing issues. The objective of this paper is to conduct an investigation of key issues involved to ensure the predictability of the system, introduce rigorous definitions of performance parameters, and propose metrics for their evaluation and analyze their suitability to be used in the presence of uncertainties in which CPS operate. The results are expected to provide greater insight into the time critical behavior of CPS components.

1. Introduction

Cyber–Physical Systems (CPS) are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, with feedback loops where physical processes affect computations and vice versa [38,40]. Predictability is crucial for CPS testing and verification, which are required by time-critical applications such as traffic control, automotive safety, and health care systems [20,38,40,41,45]. To improve the predictability of concurrent and preemptive CPS, the first and foremost work is to figure out the properties that should be predictable in CPS and precisely define the concept of predictability.

The predictability problem resounds throughout the embedded systems community, particularly throughout the real-time community. Stankovic and Ramamritham discuss the need for predictability, with respect to the time requirements of different kinds of systems [52]. Thiele and Wilhelm define the time predictability by considering the difference between the real Worst-Case-Execution-Time (WCET) and the WCET bound, which is often called the pessimism of the analysis, as a measure for predictability [53]. Kirner and Puschner define predictability as the interval between the Best Case Execution Time (BCET) and the WCET, where a smaller interval means better predictability [28]. Grund et al. investigate and propose a template for the predictability definitions, and define the predictability as the quotient of the minimum execution time over the maximum execution time with all the possible inputs [21]. The state-of-the-art predictability definitions can be classified into four categories: 1) interval between BCET and WCET [28]; 2) quotient of BCET over WCET [21]; 3) quotient of WCET over WCET bound [53]; 4) quotient of BCET over WCET bound [53]. Almost all the aforementioned definitions of predictability are based on WCET and BCET, which are estimated by assuming the uninterrupted execution of a single task [4,21,28,53,56]. However, the assumption is not true for most of CPS due to the concurrent and preemptive nature of CPS [38]. In addition, CPS must be robust in presence of unexpected conditions, and adaptable to subsystem failures. Therefore, the predictability of CPS has more rigorous semantic requirements than the above definitions [40].

The objective of this paper is to investigate issues involved to ensure the time predictability of CPS, introduce rigorous definitions of performance parameters, and propose metrics for their evaluation and analyze their suitability to be used in the presence of uncertainties in which Cyber Physical Systems operate. Through analyzing the behaviors of I/O and function task of typical programming models and characteristics of CPS, we argue that both the function tasks and I/O must have time predictability and order predictability. The I/O behaviors must be predictable in terms of their ready time and the inputs ready order, where the ready time refers to the interval between the time when the value in the I/O source changes and the time when the value in the I/O port is modified correspondingly. The predictability of function tasks are discussed and defined according to the concurrent and preemptive characteristics of CPS. It includes the time predictability of individual task and tasks set. And the execution order predictability of tasks in a scheduling set must be predictable as well. We investigate the sources of uncertainties of different properties that must be

* Corresponding author. Tel.: +86 0512 62888062
E-mail address: llxx@ustc.edu.cn (X. Li).

http://dx.doi.org/10.1016/j.sysarc.2016.01.007
1383-7621/

Table 1
A comparison between the existing predictability definitions and our definitions.

	Existing definitions	Our definitions			
Behaviors	Function task	Function task		I/O	
Properties	Execution time	Response time	Execution order	I/O ready time	Inputs ready order
Sources of uncertainties	Inputs, Hardware states	Inputs, Hardware states, Concurrency, Preemption, Scheduling, Runtime system, Mutex hazards		Sensor, Network, Memory delay	Inputs ready time, processing order
Metrics	Task execution time variability	Task response time variability	Execution order distance variability	Inputs ready time variability	Inputs buffer costs variability

predictable, and propose definitions for them based on the uncertainties and the template introduced in [21]. Table 1 shows the differences between the existing definitions and our definitions. The major contributions of our work are as follows:

(1) We give a comprehensive catalog of possible unpredictability and summarize properties that must be predictable. We emphasize the importance of I/O predictability by analyzing their behaviors in different programming models, and we analyze the function task predictability requirements in the concurrent and preemptive environments.
(2) A careful investigation of the uncertainty sources of different properties is made. By analyzing the major uncertainties, we propose metrics and formal predictability definitions for those properties. We present methods to calculate parameters that are used in the definitions.
(3) To the best of our knowledge, this paper is the first to stress the importance of order predictability in addition to only time predictability reported in the literature. We propose new ideas to determine the order (i.e., the inputs ready order and the tasks execution order), and present methods to calculate their predictability.

Our definitions can serve as metrics for measuring and comparing the predictability of CPS. CPS that have greater predictability tend to have more reliable testing and verification results, as well as time safety. When applied during CPS design, the metrics help the designer in selecting suitable programming models, runtime systems and hardware platforms to satisfy safety as well as energy efficiency requirements. The order predictability can also help to improve the CPS design, i.e., a predictable inputs ready order is critical for the predictability of the input-determined real-time systems, and the task execution order is important to make the scheduling algorithm predictable and avoid some scheduling anomalies and the mutex hazards as well. Finally, since our predictability definitions capture major sources of uncertainties, they can be used to find out the bottleneck of the predictability. The designer can improve the system's predictability effectively by optimizing the bottleneck.

The rest of this paper is organized as follows: Section 2 discusses the requirements of the predictability of CPS and summarizes the behaviors and properties that must to be predictable. We investigate the sources of uncertainties of each predictable property, and present formal definitions in Section 3. Section 4 shows the related works, and we conclude our work and show the future work in Section 5.

2. Predictability requirements of CPS

In this section, we illustrate that both the function task behaviors and the I/O behaviors must be predictable by analyzing their logical and real time performance in different real-time programming models at first. Then, from the characteristics of CPS standpoint, we argue that both the time property and the order property of the different behaviors must be predictable.

According to Klein and Ralya [31], **Input** is defined as reading data from one or more sources of input; **Output** is defined as writing the results of the computation to one or more sinks, which may be devices and/or main memory; **function task** is defined as the process to compute output values, which are functions of the gathered input values. These definitions are used throughout this paper.

2.1. Behaviors need to be predictable

Most of the real-time programs are input-determined programs, i.e., if, for all sequences of input values and times, the program produces, in all runs, unique sequences of output values and times [29]. Therefore the I/O behaviors are critical to improve the predictability of the system. However, the I/O behaviors of most of the real-time programs are unpredictable in either the logical time (specified by programming models) or the real time (performed during run time). We use the example shown in Fig. 1 to analyze the I/O behaviors of a task, which is realized in different programming models and executed in preemptive environments. The top half of Fig. 1 shows the logical time of three real-time programming models (summarized in [16]). The bottom half shows the execution of a task in different real time periods, during which the task can be preempted by other tasks.

The *asynchronous model* bounds execution times using deadlines. A system built on the asynchronous model is a set of programs that consist of a finite number of tasks, which execute concurrently under the supervision of specific mechanisms such as real-time operating systems [16]. The execution of asynchronous program is correct if the tasks finish writing outputs before deadline. However, since no I/O behaviors are specified in the asynchronous model, the I/O behaviors are highly unpredictable in both the logical time and the real time. As shown in Fig. 1(a), the time to process I/O is not specified in the programming model. In the first real time period, the input and output happen at the second time instant and the seventh time instant, respectively, while in the second real time period, they are processed at the first time instant and the eighth time instant, respectively. In the *synchronous model*, the logical temporal reference is completely determined by the successive reactions of the system happening on the occurrences of observed events [8,23]. Due to the synchronous assumption of the model, the I/O behavior is predictable in the logical time. In the real time, the input is processed at fixed time instant, such as PRET-C [2]. However, since the synchronous program is correct as long as the output completes before the next input, the output behavior is still not predictable in the real time. As shown in Fig. 1(b), the logical time to read input is specified in the programming model. Therefore, the input is read at beginning of each real time period. However, the output is written at the seventh time instant and the 8*th* time instant in different real time periods, respectively. The *timed model* (e.g., the Logical Execution Model (LET) [30]) describes the system with logical duration information, which is specified by the designer [16,30]. Since the time information is priori fixed, e.g., the times to read input and to write output are fixed in the LET model, the I/O behaviors of the LET model are predictable and composable both in the real time and the logical time [29,30]. As shown in Fig. 1(c), the logical