INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On the Validation of Robotics Control Systems Part II: Analysis of real-time closed-loop control tasks

Daniel SIMON , Eduardo CASTILLO and Paul FREEDMAN

N° 2720

Novembre 1995

PROGRAMME 4

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

On the Validation of Robotics Control Systems Part II: Analysis of real-time closed-loop control tasks

Daniel SIMON *, Eduardo CASTILLO ** and Paul FREEDMAN ***

Programme 4 — Robotique, image et vision
Projet Icare

Rapport de recherche n° 2720 — Novembre 1995 — 24 pages

Abstract: In the framework of the ORCCAD system, periodic and multi-rate control laws are implemented in terms of a set of computing tasks to be executed under a real-time operating system. Simulations and experiments demonstrate that partially synchronizing such tasks can improve the practical performance of the implementation. In this paper, we examine the consequences of introducing such synchronisation in terms of two temporal problems which may occur and how they may be detected using Petri net modeling. We conclude with some guidelines about how to add such synchronisation to design deadlock free and efficient real-time periodic control laws.

Key-words: Real-time, multitask, synchronization, Petri net modeling, verification

(Résumé : tsvp)

*INRIA Sophia-Antipolis, e-mail: dsimon@sophia.inria.fr
**ITESM, j.Oviedo No 10, 76130-Queretaro,MEXICO, e-mail:ecastill@itesmcq1.qro.itesm.mx. This author was supported by Conacyt during PhD studies at Inria
***CRIM, 1801 Av. McGill College, Bureau 800, H3A 2N4 MONTREAL, Québec, CANADA, e-mail: freedman@dartagnan.crim.ca

Sur la validation de systèmes de commande en robotique 2ème partie : Analyse de tâches de commande temps-réel en boucle fermée

Résumé : Dans le système ORCCAD les lois de commande temps-réel et multicadence sont implantées sous la forme d'un réseau de tâches communicantes plus ou moins fortement synchronisées, sous le contrôle d'un système d'exploitation temps-réel. Des simulations et des expériences sur site réel ont montré qu 'une synchronisation partielle judicieuse de ces tâches permet d'améliorer l'efficacité de ces lois de commande. Nous montrons que l'introduction de synchronisations peut cependant introduire des problèmes de type interblocages ou incoherences de spécification. Une modélisation à l'aide de réseaux de Petri permet de vérifier formellement l'absence d'interblocage dans le réseaux de tâches synchronisées et de détecter l'absence d'un certain type d'incoherence temporelle. Finalement, des conditions suffisantes permettant de construire ces lois de commande efficaces et exemptes d'interblocages est donnée.

Mots-clé : Temps-réel, multitâche, synchronisation, réseaux de Petri, vérification

Contents

1 Introduction

ORCCAD is a set of design, programming and verification tools aimed to ease the development of efficient and safe robotic tasks and applications ([29]). It is based on a bottom-up approach, where a basic assumption is that many complex robotic actions can be stated and efficiently solved using Automatic Control theory, e.g. the Task function approach ([24]). Therefore, the first items to be designed and verified are the so-called Robot-Tasks (RT), i.e. closed-loop control laws encapsulated in a reactive logical behavior. These RTs are then composed to build more complex actions and full robotic missions as explained in a companion paper ([9]).

Usually, the design of control laws for non-linear systems like robots is made in continuous time and allows for checking mainly qualitative properties like stability. At run time, these control laws are generally implemented as multitasks programs controlled by a real time operating system on a single or multiprocessors target. Multitasks programs permits modular programming, software reusability and design of multirate controllers. This last feature is useful when the plant exhibits subparts with different dynamics and when feedforward paths are used to update some parameters of the controller. Also, it can useful to optimize computing resources, e.g. for untethered underwater vehicles where both on-board space and energy are limited ([20]).

Actual control laws must meet end-users' requirements like the maximum value of tracking errors, time of response and perturbation rejection. Besides the algorithm in use, these quantitative performance indexes strongly depends on the actual implementation and in particular on sampling rates and computing latencies. Unfortunately, non-linear systems control theory do not provide tools to analyze or synthetize such sampled control laws with respect to output performance indexes. On the other hand, research on real-time operating systems deals with important issues like schedulability, fault tolerance or liveness but do not measure the impact of the organization of the controller on the controlled process. In fact, rather few such work is reported.

In [6], Chen et al make a first attempt to analyze the impact of synchronization between concurrent control tasks on the tracking accuracy of an industrial manipulator. In a further report ([1]) Armstrong computes the average latency in a two tasks controller and concludes that the way they must be synchronized depends on the ratio between their respective durations. In [15] Khosla reports experiments on a direct driven arm while varying the sampling rate of single loop controllers and shows that higher sampling rates allows for using higher gains with respect to stability. Whitcomb et al ([31]) computes the cross latency matrix of a multivariable control system for a robot arm implemented on a network of Transputers where both asynchronous and handshaking communications are available between the processors. In a more recent paper ([27]), Shin et al refers to delay and loss problems according to the respective values of the sampling rate and of the latency and computes upper bounds for the values of the gains of the controller of a robot arm.

In this paper we report some results about the liveness of sets of synchronized real-time tasks in the framework of the ORCCAD system. In the next section we define the main entities of the lower level of our system, i.e. tasks and synchronization protocols together with the temporal properties we want to check. In section 3 we design a Petri net model of the network of communicating tasks and show in section 4 how it can be used to verify deadlocks freedom. Extending the model in section 5 allows to check for some kind of temporal incoherence. A synthesis method to build deadlock free, partially synchronized control laws is given in section 6. Further possible improvements are outlined in the conclusion.

2 From control laws to multitasks programs

2.1 Robot-tasks and Module-tasks

The *Robot-Task* (RT) in ORCCAD is the minimal "granularity" seen by the end-user at the application level, and the maximum granularity considered by the control systems engineer at the control level. It characterizes in a structured way continuous time closed loop control laws, along with their temporal features related to implementation and the management of associated events. More formally, a RT is the entire parametrized specification of:

- an elementary servo-control task, i.e. the activation of a control scheme *structurally invariant* along the task duration;
- a logical behavior associated with a set of signals (events) which may occur before or during the task execution.

Structure in the description thereafter makes it possible to "translate" the continuous-time specification into a description taking into account implementation aspects including *temporal properties*, i.e. discretization of the time, durations of computations, communication and synchronization between the involved processes. This is done by defining each RT in terms of communicating realtime computing "tasks" called *Module-Tasks* (MT) which each implement an elementary part of the control law.

Most of the MTs are periodic tasks: some perform the calculations involved in the computation of the control algorithm, e.g. the task Jacobian matrix or the control torque. Others, called *observers*, monitor conditions and are therefore used to handle preconditions, postconditions and exceptions. The non-periodic *reactive* behavior of the RT is handled by a special MT called the Robot-task Automaton (RTA) which may be awakened by signals coming from the RT itself through the outputs of the observers and is used to link the RT to the input/output signals associated with the Robot-Procedure and Application level ([9]).

Since we expect that in most cases, the MTs will be distributed over a multiprocessor target architecture, ORCCAD makes available various message passing mechanisms over typed ports. Moreover, in order to ease the automatic code generation from the graphical HMI, the structure of the periodic MTs is as shown in Figure 1. Such a structure clearly separates calculations, related to control algorithms issues, and communications, related to implementation aspects and calls to the underlying operating system.

The temporal attributes of a MT are:

- its (nominal or worst case) duration d_i, which mainly depends upon the algorithm, the programming language used to encode it (generally C or C++), and the target microprocessor. It is often simply estimated for simulation purposes.
- its activation period τ_i (the reciprocal of sampling rate measured in Hz).
- the set of input and output ports of the MT and their associated communication protocols.
- their priority, allowing them to be scheduled at run-time by the operating system

Note that the ports of each MT (except the RT Automaton) are read or written in the order they are declared.