Available online at www.sciencedirect.com
ScienceDirect
Control Engineering Practice 15 (2007) 1416–1426
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

Automatic model generation and PLC-code implementation for interlocking policies in industrial robot cells

H. Flordal[a,*], M. Fabian[a], K. Åkesson[a], D. Spensieri[b]

[a]Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Göteborg, Sweden
[b]Fraunhofer-Chalmers Research Centre for Industrial Mathematics, Chalmers Science Park, SE-412 88 Göteborg, Sweden

Received 2 March 2005; accepted 1 November 2006
Available online 2 March 2007

Abstract

In industrial production lines, for example in the automotive industry, cells with multiple industrial robots are common. In such cells, each robot has to avoid running into static obstacles and when the robots work together in a shared space they must also avoid colliding with each other. Typically, the latter is enforced by manually implementing interlocks in programmable logic controllers (PLCs). This is a tedious, error-prone task that is a bottleneck in the development of production lines. The PLC-code being man-made also greatly complicates the maintenance and reconfiguration of such production lines. However, in industry today, a lot of development of robot cells is made offline in 3D simulation environments which enables the use of computers also for deciding and implementing the necessary coordination. This paper presents a method that makes use of information in a robot simulation environment in order to automatically extract finite state models. These models can be used to generate supervisors for ensuring that the deadlock situations that may arise as a consequence of the introduced interlocks are avoided. It is also possible to optimize the work cycle time for the cell. Finally, PLC-code to supervise the production cell can be automatically generated from the deadlock-free and possibly optimized system model. This approach results in a high flexibility in that the coordination function can be quickly reimplemented whenever necessary. A prototype implementation has been developed making use of a commercial 3D robot simulation tool, and a software tool for supervisor synthesis and code generation. The approach is general and should be possible to implement in most offline robot simulation tools.

Keywords: Robot programming; Model generation; Programmable logic controller programming; Deadlock avoidance; Discrete event systems; Supervisory control

1. Introduction

Today's trend of mass customization and the ambition to shorten the time-to-market result in a demand for more flexible manufacturing systems, faster development of new production lines and faster adaptation of old production lines for producing new products. The automotive industry is a very illustrative example with mass production of long series of car models being replaced by customized products in short series.

Production in the *body-in-white* shops of the automotive industry is typically centered around an assembly line, where the car body moves from cell to cell. Within a cell multiple robots weld parts such as wheel-houses and the roof, to the stationary car body. As the robots sometimes work inside each others' workspaces there is a risk that the robots collide. Naturally, this must always be avoided.

To prevent robot collisions, the access to shared spatial volumes, here called *zones*, must be coordinated. Only one robot may occupy a zone at a time, to guarantee freedom of collision. However, this coordination may instead introduce *blocking* problems, where a number of robots get into a circular wait, indefinitely waiting for each other to release a currently occupied zone. Naturally, this must also always be avoided, hence the coordination problem is by no means trivial.

In industry today, this coordination is often implemented manually, either offline in a 3D simulation environment or online on the factory floor. The robots' interaction is

*Corresponding author. Tel.: +46 31 772 1803; fax: +46 31 772 1782.
E-mail address: flordal@chalmers.se (H. Flordal).

0967-0661/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2006.11.001

examined and whenever a collision is imminent in the concurrent execution of the robot programs, *interlocks* are introduced manually to prevent collisions. This is a tedious and error prone process that typically results in a highly inflexible coordination function which among other things make reconfiguration of the cell problematic. Additionally, the manually specified work cycle has no guarantees for being optimal, neither with respect to blocking nor various production "costs" such as cycle-time.

Typically, the zone coordination is implemented in a PLC (programmable logic controller). The PLC supervises the robots' sharing of the zones, and hinders a robot from claiming (and entering) a zone if this violates the constraint of only one robot at a time in a zone, or if approving the claim inevitably leads to blocking. Deciding whether the claiming of a certain zone leads to blocking requires a model of the robots' future desired zone-claiming. In contemporary industrial production system development methods such a model is typically non-existent, or at least highly implicit.

This paper presents a structured methodology for developing the coordinating supervision, mainly for the type of production used in body-in-white production in the automotive industry. The focus here is on automatic modeling on the cell level, making use of path planning and PLC-code generation methods as well as the supervisory control framework and optimization. The presented method is an offline approach based on extracting information from 3D simulation models of a production cell and using this information to automatically generate high level models of the system. These models in turn are employed to generate a non-blocking coordination model from which PLC-code for the real production cell can be automatically implemented. This coordination model can also be used for cycle-time optimization.

Such an automatic, offline approach results in a high flexibility in that the coordination function can be quickly reimplemented whenever the need arises; when a new car model is introduced or additional robots are added into the cell, for instance.

Earlier research in industrial robot coordination includes the path planning problem, described in Canny (1988), Latombe (1999) and Bohlin (2002). An introduction to multiple robot coordination and programming can be found in Tsai (1991). An introduction to the PLC programming standard can be found in Lewis (1995). A survey on earlier research on formal methods for PLC programming can be found in Frey and Litz (2000). Moon, Lee, and Kim (2001) describe an approach similar to the one presented here. However, the approach of Moon et al. (2001) involves manually modelling the system, something that is explicitly avoided with the approach described here, the whole idea being to increase flexibility while at the same time reducing the development time. Furthermore, the focus of Moon et al. (2001) is on the performance analysis, whereas the focus here is on development of the coordinating function. Some related work on performance optimization of work cycles for multiple robot coordination was performed in Green and Jonsson (2003) where the work cycle of a real world welding cell was examined and optimized in Stark and Göthberg (2003) where a similar zone interlocking approach is presented. Both these works rely on variants of the method presented here.

The paper is organized as follows: in Section 2 the problem setting is formulated and in Section 3, the main concepts and modeling abstractions are presented. Then the automatic model generation is described in Section 4, followed in Sections 5 and 6 by employment of the models for generation the PLC-code for coordination. In Section 7 there is a discussion around the implementation and finally, conclusions and suggestions for future work are presented in Section 8.

2. Problem

The setting of the problem in this paper is a cell with a number of robots, each with a number of *targets* to visit. The targets may be places where the robot should perform work, e.g. weld or grip something, or places the robot should visit in order to avoid stationary obstacles.

The *task* of each robot is to, starting from its home position, visit all its targets and then return to the home position. Unlike in Flordal, Fabian, and Åkesson (2004), no specific ordering of the targets is assumed. However, as is shown below, such constraints can be added at will.

To avoid collisions among robots, parts of the 3D workspace will be identified as *mutual exclusion zones* (henceforth *zones*). These zones can be accessed by at most one robot at a time, and hence the zone-sharing must be coordinated. At the same time, all robots must always be able to finish their tasks. The crux is that *blocking* may arise as a consequence of the zone-sharing coordination. Blocking prevents one or more robots from finishing their tasks and must therefore be avoided.

The main goal here is to provide a systematic and largely automatic method to develop a blocking-free model for the zone-sharing coordination, given the targets and a 3D robot-simulation environment. The robot simulation software includes a model of the physical cell with the robots. For the method to be fully automatic the simulation software must have means to be driven from some other external software. The methodology presented in this paper has been implemented using RobotStudio, see Robotstudio manual (2003); a robot simulation environment from ABB, being driven by `Supremica`, a general tool for verification and synthesis of DESs developed at Chalmers University of Technology, see Åkesson, Fabian, Flordal, and Vahidi (2003).

From the resulting blocking-free model of the zone-coordination, PLC-code that guarantees that the robots will not collide and that the system will not block can be automatically generated. In addition, the *makespan*, that is, the total time of work, can be calculated and the optimal sequence of tasks can be selected (if the simulation software