Hierarchical Interface-based Supervisory Control: AIP Example

R.J. Leduc,[1,2], M. Lawford[2], and W.M. Wonham[1]

[1]Dept. of Electrical and Computer Engineering, University of Toronto
[2]Dept. of Computing and Software, McMaster University
email: leduc@control.toronto.edu, lawford@mcmaster.ca, wonham@control.toronto.edu

Abstract

In this paper we present the application of the *Hierarchical Interface-based Supervisory Control* theory to a model of the Atelier Inter-établissement de Productique (AIP), a large (7×10^{21} possible states), highly automated, manufacturing system. We first describe the model of the AIP system, then our interfaced based supervisor design, and finally we apply the local checks described in [10] to show that the system is globally nonblocking and controllable. This example demonstrates that Hierarchical Interface-based Supervisory Control can be applied to interesting systems of realistic complexity that were previously far beyond the means of previous monolithic, modular, or hierarchical supervisor design techniques.

1 Introduction

In the area of Discrete-Event Systems (DES), two common tasks are to verify that a composite system, based on a cartesian product of subsystems, is (i) nonblocking and (ii) controllable. The main obstacle to performing these tasks is the combinatorial explosion of the product state space. Although many methods have been developed to deal with this problem (modular control [1, 16, 20], decentralised control [12, 17, 21], model aggregation methods [2, 5, 19, 23], and multi-level hierarchy [4, 7, 8, 13, 14, 18]), large-scale systems are still problematic, particularly for verification of nonblocking.

In [10, 11], we presented the parallel case of a method called *Hierarchical Interface-based Supervisory Control (HISC)* that decomposes a system into a *high level subsystem* which communicates with $n \geq 1$ parallel *low level subsystems* through separate interfaces, which restrict the interaction of the subsystems. The method uses interfaces designed as DES, and a set of local consistency properties that can be used to verify if a discrete-event system is globally nonblocking and controllable. As each of these consistency properties can be verified using a single subsystem, the complete system model never needs to be constructed, thus offering significant savings in computational effort. This paper will use the terminology and results of the companion paper [10], but for reasons of brevity will not restate them here.

It is worth noting that Zhang et al. have developed algorithms that use Integer Decision Diagrams to verify centralized DES systems on the order of 10^{23} states [22]. These results are complementary to the hierarchical method illustrated in this paper, as their approach can be used to verify many of the required conditions, allowing HISC to scale to even larger systems.

To demonstrate the utility of our method, we now present its application to a large manufacturing system, the Atelier Inter-établissement de Productique (AIP). Section 2 provides an overview of the AIP system while Section 3 describes the desired closed loop behavior of the system and associated assumptions about the plant. We describe the modular decomposition (in the sense of [15]) of the system and provide the details of the component interfaces and supervisor design in Section 4. We close with Section 5 discussing the results of applying the method to show that the AIP system is nonblocking and controllable.

2 Overview of the AIP

In this section, we introduce the automated manufacturing system of the Atelier Inter-établissement de Productique (AIP), as described in [3] and [6]. The AIP, shown in Figure 1, is a highly automated manufacturing system consisting of a central loop (CL) and four external loops (EL), three assembly stations(AS), an input/output (I/O) station, and four inter-loop transfer units (TU). The I/O station is where the pallets enter and leave the system. Pallets can be of type 1 or of type 2, and it is assumed that the type of the pallet entering is random.

2.1 Assembly Stations

The structure of the assembly stations is shown in Figure 2. Each station consists of a robot to perform assembly tasks, an extractor to transfer the pallet from the conveyor loop to the robot, sensors to determine the location of the extractor, and a raising platform to present the pallet to the robot. The station also contains pallet sensors to detect a pallet at the pallet gate, the pallet stop, and to detect when a pallet has left the station. Finally, the assembly station contains a read/write (R/W) device to read and write to the pallet's electronic label. The pallet label contains information about the pallet type, error status, and assembly status (which tasks have been performed).

Whereas the assembly stations contain the same basic components, they differ with respect to functionality. Station 1 is capable of performing two separate tasks denoted task1A and task1B, while station 2 can perform tasks task2A and task2B. Station 3 can perform all four of these tasks as well as functioning as a repair station allowing an operator to repair a damaged pallet. The assembly stations also differ with respect to reliability. Stations 1 and 2 can break down and must be repaired, while station 3 is of higher quality and is assumed never to break down. Station 3 is used to substitute for the other stations when they are down.

2.2 Transport Units

The structure of the four identical transport units is shown in Figure 3. The transport units are used to transfer pallets between the central loop, and the external loops. Each one consists of a transport drawer which physically conveys the pallet between the two loops, plus sensors to determine the drawer's location. At each loop, the unit contains a pallet gate and a pallet stop, to control access to the unit from the given loop. The unit also contains multiple pallet sensors to detect when a pallet is at a gate, drawer, or has left the unit. Also, each unit contains a R/W device located before the central loop gate.

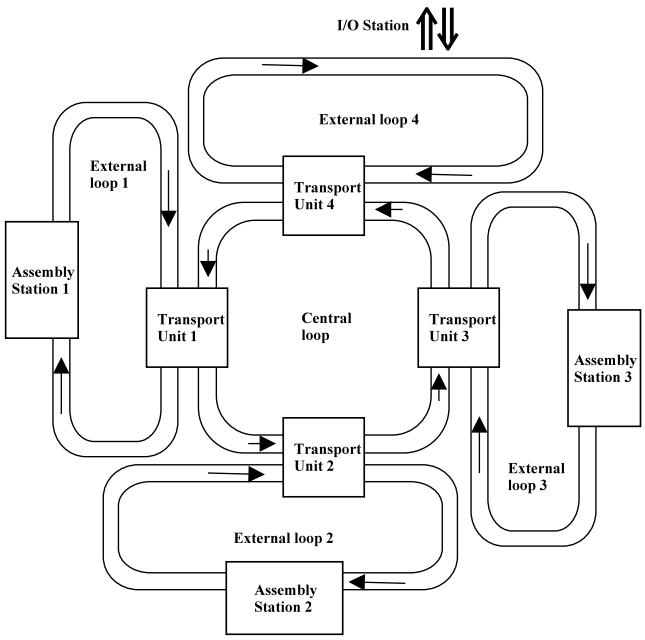

Figure 1: The Atelier Inter-établissement de Productique

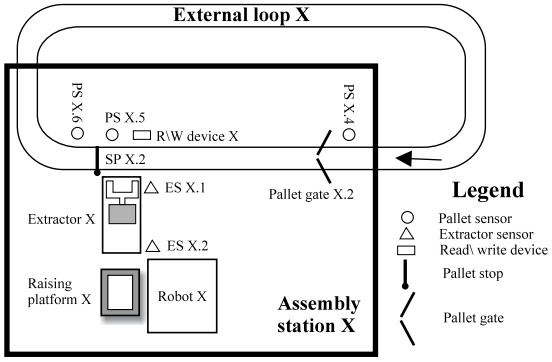

Figure 2: Assembly Station of External Loop $X = 1, 2, 3$.

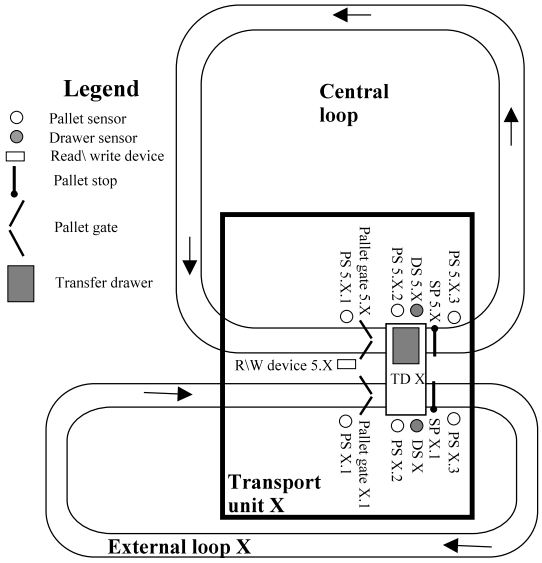

Figure 3: Transport Unit for External Loop $X = 1, 2, 3, 4$

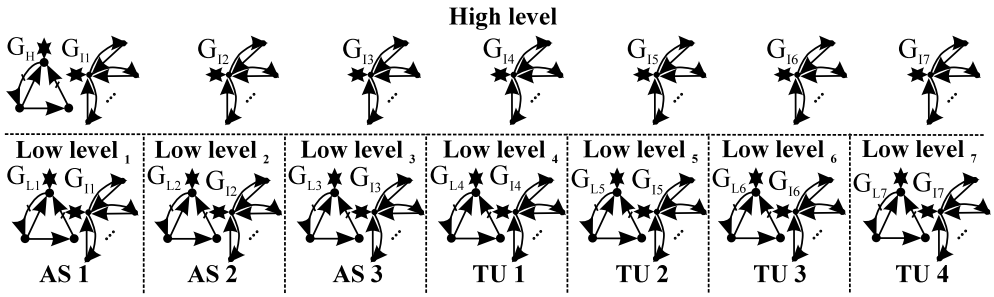

Figure 4: Structure of Parallel System

3 Control Specifications

For this example, we adopt the control specifications and assumptions used in [3] and [6] and restated below. To this we add **Specification 7** to make the assembly stations more interesting.
Assumptions: We assume that (1) the system is initially empty, and (2) two types of pallets are randomly introduced to the system, subjected to assembly operations, and then leave.
Specifications:

1. **Routing:** Pallets follow a certain route based on their type. A type 1 pallet must go first to AS 1, then AS 2 before leaving the system. Type 2 pallets go first to AS 2, then AS 1 before leaving the system. A pallet is not allowed to leave the system until all four assembly tasks have been successfully performed on it.

2. **Maximum capacity of external loops 1 and 2:** The maximum allowed number of pallets in either loop at a given time is one.

3. **Ordering of pallet exit from system:** The pallets must exit the system in the following order: type 1, type 2, type 1, ...

4. **Assembly errors:** When a robot makes an assembly error, the pallet is marked damaged and routed to AS 3 for maintenance. After maintenance, the pallet is returned to the original assembly station to undergo the assembly operation again.

5. **Assembly station breakdown:** The robots of external loops 1 and 2 are susceptible to breakdowns. When a station is down, pallets are routed to assembly station 3 which is capable of performing all tasks of the other two stations. When the failed station is repaired, all pallets not already in external loop 3 are rerouted to the original station.

6. **Maximum capacity of assembly stations:** To avoid collisions, only one pallet is allowed in a given station at a time.

7. **Assembly task ordering:** Assembly tasks are performed in a different order for pallets of different types. For pallets of type 1, task1A is performed before task1B, and task2A is performed before task2B. For pallets of type 2, task1B is performed before task1A, and task2B is performed before task2A.

4 System Structure

To cast the AIP into a parallel interface system, we break the system down into a *high level,* and seven *low levels* corresponding to the three assembly stations and four transport units, as shown in Figure 4. Each subsystem is described in the following sections. The models and supervisors developed for this example are based on the automata presented in [3] and [6]. We have altered them to fit our setting, and we have extended them to fill in the missing details of several events that were defined as "macro events" in the cited references. The authors note that while all the supervisors that follow were created manually, standard synthesis techniques could be applied to generate each of the local supervisors from formal specifications. As this example contains 181 DES, we are not

able here to describe the design in complete detail. We refer the reader to [9] for a complete description of the system.

In the diagrams to follow, *uncontrollable events* are shown in italics; all other events are *controllable.* Also, initial states can be recognised by a thick outline, and marked states are filled.

4.1 The High Level

The *high level subsystem* contains 15 DES, and keeps track of the breakdown status of assembly stations 1 and 2, as well as enforces the maximum capacity of external loops 1 and 2. This subsystem controls the operation of all transport units and all assembly stations, as well as tracking the pallets' progress around the manufacturing system.

As an example of the *high level subsystem's* behaviour, we discuss supervisor *ManageTU1,* shown in Figure 7. This supervisor controls the transfer of pallets between the central loop and external loop 1. It permits pallets on the central loop to pass through transport unit 1 (to be liberated) without being transferred to the external loop. Pallets are liberated if the attached external loop is at maximum capacity, assembly station 1 is down, or TU 1 determines that the pallet is not to be transferred.

4.2 Low Levels AS 1 and AS 2

We now describe the *low level subsystems* that represent assembly stations 1 and 2. As they are identical, we will describe them collectively as *low level subsystem* k, where $k = \text{AS } 1, \text{AS } 2$.

Subsystem k contains 17 DES and provides the functionality specified in its interface, shown in Figure 5. An assembly station accepts the pallet at its gate, and then presents it to the robot for assembly. It then releases the pallet, and reports on the success of the assembly operation. If the robot breaks down, this is reported through the interface, and the pallet is released. Subsystem k then waits for a repair command to return the robot to operation.

Supervisor *HndlPallet.AS1,* shown in Figure 10, provides an example of *low level subsystem* AS 1's behaviour. *HndlPallet.AS1* handles the task of processing a pallet once it reaches the extractor. It reads the pallet's label, presents the pallet to the robot, and has the robot perform the appropriate tasks on the pallet. The supervisor then allows the pallet to leave the assembly station and reports on the success of the processing operation by updating the pallet's label, and notifies the *high level subsystem* through their common interface.

4.3 Low Level AS 3

Low level subsystem AS 3 contains 27 DES and provides the functionality specified in its interface, shown in Figure 6. This subsystem describes the behaviour of assembly station 3, which is very similar to stations 1 and 2. The main differences are that station 3 can repair damaged pallets, is assumed not to breakdown, and can substitute for either AS1 or AS2 when they are down.

Supervisor *DoRobotTasks.AS3,* shown in Figure 11, provides an example of *low level subsystem* AS 3's behaviour. *DoRobotTasks.AS3* controls the operation of assembly station 3's robot. The supervisor makes sure that the assembly tasks are performed in the correct order for a given type of pallet, and then it reports on the success of the assembly

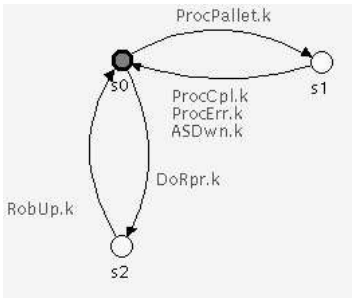

Figure 5: Interface to *low level k*.

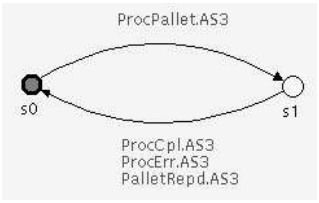

Figure 6: Interface to *low level* AS 3.

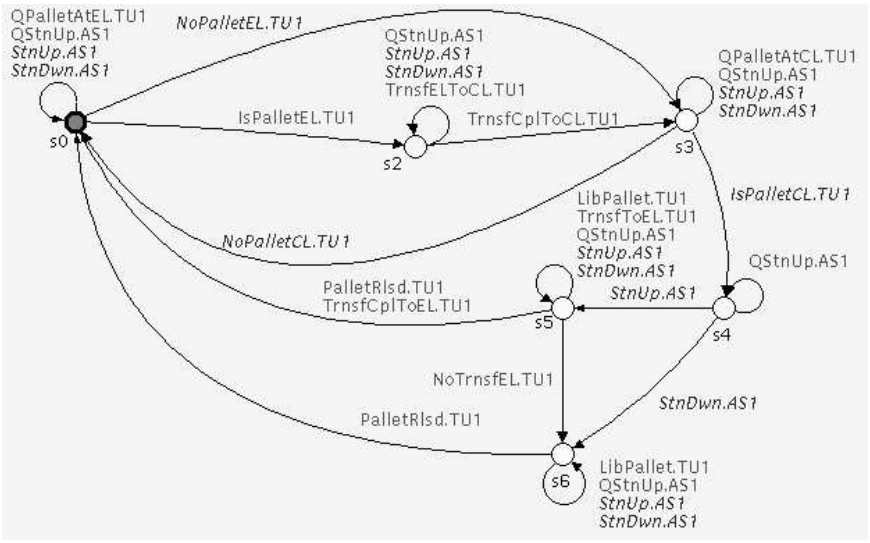

Figure 7: Supervisor ManageTU1.

operation. As AS 3 only performs assembly tasks when another station is down, the robot can perform assembly tasks as if it were substituting for AS 1's robot or AS 2's robot.

4.4 Low Levels TU 1 and TU 2

We now describe the *low level subsystems* that represent transport units 1 and 2. As they are identical, we will describe them collectively as *low level subsystem* r, where $r = \text{TU } 1, \text{TU } 2$.

Subsystem r contains 25 DES and provides the functionality specified in its interface, shown in Figure 8. The transport units are used to transfer pallets between the central loop, and the external loops (ie. TU 1 transfers pallets between CL and EL 1). Subsystem **r** has two entry points for pallets, the central loop gate, and the external loop gate. If a pallet is at the EL gate, subsystem **r** transfers the pallet to the central loop. If a pallet is at the CL gate, subsystem **r** can be requested to liberate the pallet (allow it to pass through and continue on CL), or to transfer the pallet to the EL. When requested to transfer a pallet to the EL, subsystem **r** will only transfer the pallet if the pallet is undamaged and if the next assembly task required by the pallet is performed by the external loop's assembly station.

Supervisor *HndlTrnsfToEL.r,* shown in Figure 12, provides an example of *low level subsystem* r's behaviour. *HndlTrnsfToEL.r* handles transporting pallets from the central loop to the external loop. It only transfers pallets if they are undamaged, or if the next

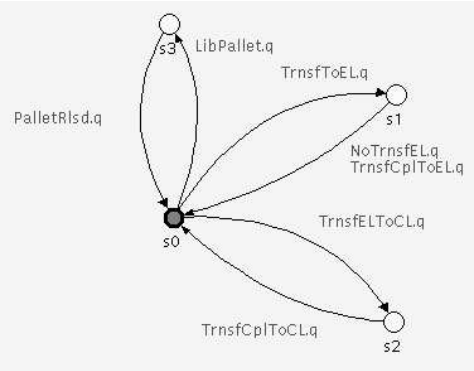

Figure 8: Interface to *low level* q = TU 1, TU 2, TU 4.

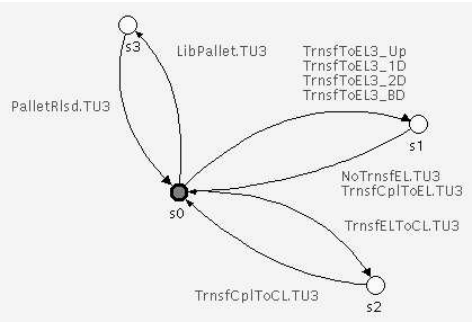

Figure 9: Interface to *low level* TU 3.

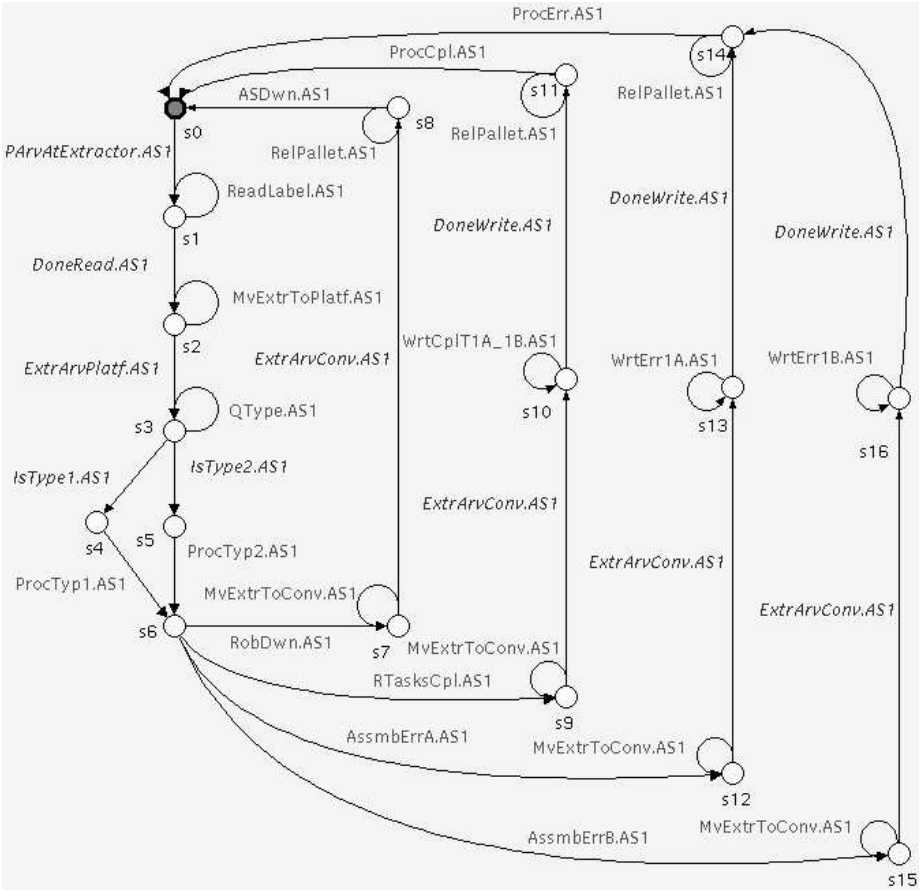

Figure 10: Supervisor HndlPallet.AS1

assembly task required by the pallet is performed by the external loop's assembly station.

4.5 Low Level TU 3

Low level subsystem TU 3 contains 29 DES and provides the functionality specified in its interface, shown in Figure 9. This subsystem describes the behaviour of transport unit 3, which is very similar to TU 1 and TU 2. Subsystem TU 3 differs in how it decides if a pallet should be transferred from the central loop to external loop 3. First, all damaged pallets are to be transferred to EL 3 for maintenance. Second, if an assembly station

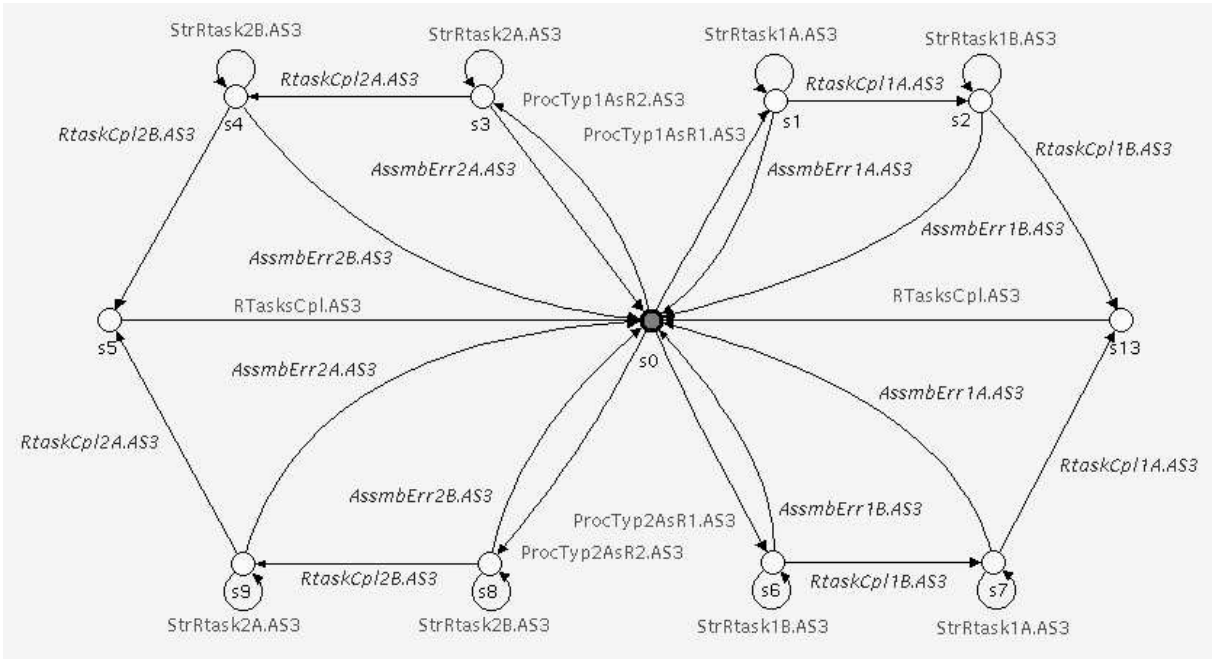

Figure 11: Supervisor DoRobotTasks.AS3

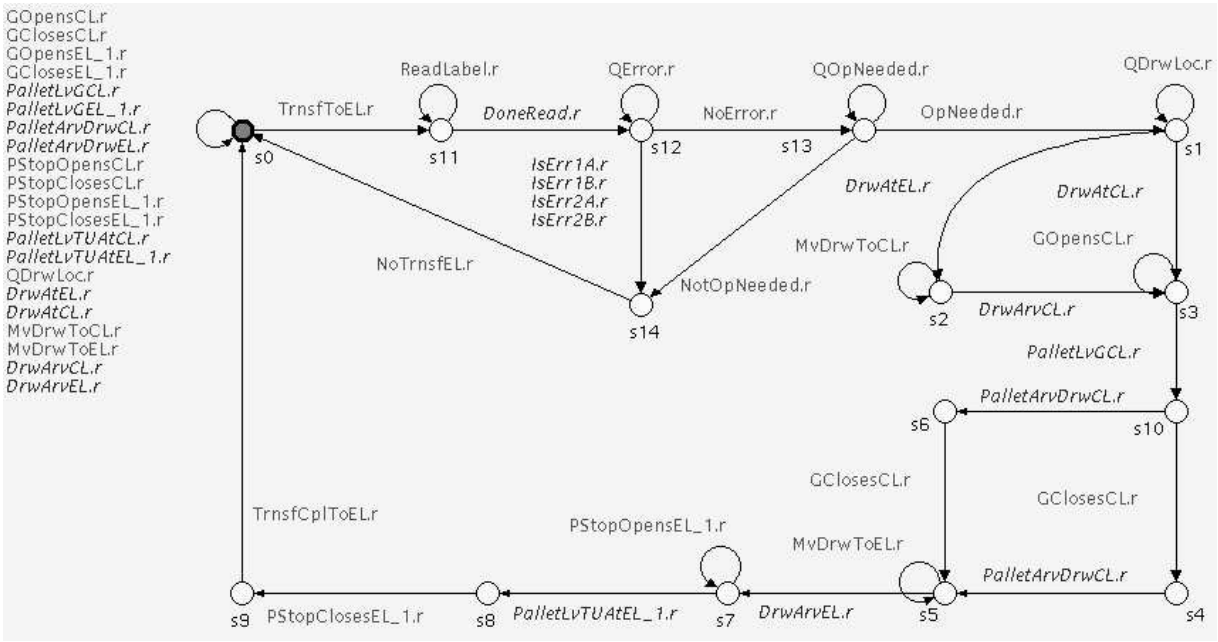

Figure 12: Supervisor HndlTrnsfToEL.r

is down and it performs the next pending task for the pallet, then the pallet is to be transferred. As subsystem TU 3 most know the breakdown status of assembly stations 1 and 2, this information is passed in explicitly as separate *request events* (see Figure 9, events from state s_0 to s_1).

4.6 Low Level TU 4

Low level subsystem TU 4 contains 19 DES and provides the functionality specified in its interface, shown in Figure 8. This subsystem describes the behaviour of transport unit 4, which is very similar to TU 1 and TU 2. Subsystem TU 4 differs in how it decides

if a pallet should be transferred from the central loop to external loop 4, which contains the I/O station (the I/O station is where pallets enter and leave the system). As pallets are required to leave the system in a particular order (ie. type 1, type 2, type 1, ...), subsystem TU 4 keeps track of the type of the last pallet to be transferred to EL 4 and will only transfer the current pallet if it is of the type required by the sequence, and if all required assembly tasks have been successfully performed on the pallet.

5 Conclusions

Analysing the system, we found it to be *interface consistent, level-wise nonblocking,* and *level-wise controllable.* We can thus conclude by **Theorems 1** and **2** of [10], that the *flat system* is nonblocking and that the system's *flat supervisor* is controllable for the *flat plant.* As explained in [10], a system's *flat supervisor* is the synchronous product of all interfaces and supervisors in the system, and a system's *flat plant* is the synchronous product of all plant models.

This example contains 181 DES in total, with an estimated closed-loop state space of 7×10^{21}. This estimate was calculated by determining the closed-loop state space of the *high level,* and each *low level* and then these values were multiplied together to create a worst case state estimate. The computation ran for 25 minutes, using 760MB of memory. The machine used was a 750MHz Athlon system, with 512MB of RAM, 2GB of swap, and running Redhat Linux 6.2. A standard nonblocking verification was also attempted on the monolithic system, but it quickly failed due to lack of memory.

In this paper we presented a large manufacturing example that uses the parallel case of the *Hierarchical Interface-based Supervisory Control* method. This example demonstrates that our method can be applied to interesting systems of realistic complexity. Because the interface conditions can be verified using only one subsystem at a time, we were able to quickly verify a large system that was previously far beyond our means. Also, since subsystems can be verified independently, requiring only the relevant interface DES, we were able to model and design each subsystem separately. This permitted parallel development, requiring no interaction once the interfaces were defined. Also, the resulting system is easier to understand as each subsystem can be understood independently. Finally, we have a high degree of re-usability as any individual subsystem can be replaced without requiring the other subsystems to be altered or re-verified.

References

[1] N. Alsop. *Formal Techniques for the Procedural Control of Industrial Processes.* PhD thesis, Dept of Chemical Engineering and Chemical Technology, Imperial College, London, 1996.

[2] R. Alur and T. Henzinger. Local liveness for compositional modelling of fair reactive systems. In *Proc. of 7th CAV*, LNCS, pages 166–179, 1995.

[3] B. Brandin and F. Charbonnier. The supervisory control of the automated manufacturing system of the AIP. In *Proc. Rensselaer's 1994 4th Intl. Conf. on Computer Integrated Manufacturing and Automation Technology*, pages 319–324, Troy, Oct 1994.

[4] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Trans. on Automatic Control*, 38(12):1803–1819, Dec 1993.

[5] P.E. Caines and Y.J. Wei. The hierarchical lattices of a finite machine. *Systems Control Letters*, 25:257–263, July 1995.

[6] F. Charbonnier. Commande par supervision des systèmes à événements discrets: application à un site expérimental l'Atelier Inter-établissement de Productique. Technical report, Laboratoire d'Automatique de Grenoble, Grenoble, France, 1994.

[7] E. W. Endsley, M. R. Lucas, and D. M. Tilbury. Modular design and verification of logic control for reconfigurable machining systems. Submitted to Discrete Event Dynamic Systems: Theory and Applications.

[8] P. Gohari. A linguistic framework for controlled hierarchical DES. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1998.

[9] R. Leduc, M. Lawford, and W.M. Wonham. Hierarchical interface based supervisory control: AIP example for parallel case. Technical Report SERG, McMaster University, Hamilton, ON, Canada, 2001. To appear.

[10] R. Leduc, W.M. Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Parallel case. Accepted to *39th Allerton Conf. on Comm., Contr., and Comp.*, 2001.

[11] R.J Leduc, W.M. Wonham, and M. Lawford. Hierarchical interface based supervisory control: Bi-level systems. Technical report, Systems Control Group, University of Toronto, Toronto, ON, Canada, 2001. To appear.

[12] F. Lin and W.M. Wonham. Decentralized control and coordination of discrete-event systems with partial observations. In *Proc. 27th IEEE Conf. Decision Contr.*, p. 1125–1130, Dec 1988.

[13] H. Liu, J. Park, and R. Miller. On hybrid synthesis for hierarchical structured Petri nets. Technical report, Dept. of Comp. Sci., University of Maryland, College Park, MD, 1996.

[14] C. Ma. A computational approach to top-down hierarchical supervisory control of DES. M.A.Sc. thesis, Dept. of Elec. and Comp. Eng., University of Toronto, Toronto, Ont, 1999.

[15] D. L. Parnas, P. C. Clements, and D. M. Weiss. The modular structure of complex systems. *IEEE Trans. on Software Engineering*, SE-11(3):259–66, March 1985.

[16] M.H. de Queiro and J.E.R. Cury. Modular supervisory control of large scale discrete event systems. In *Proc. of WODES 2000*, pages 103–110, Ghent, Belgium, Aug 2000.

[17] K. Rudie and W.M. Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Trans. on Automatic Control*, 37(11):1692–1708, Nov 1992.

[18] B. Wang. Top-down design for RW supervisory control theory. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1995.

[19] K.C. Wong. *Discrete-Event Control Architecture: An Algebraic Approach.* PhD thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1994.

[20] W.M. Wonham. *Notes on Control of Discrete-Event Systems.* Dept. of Electrical and Computer Engineering, University of Toronto, 1999. http://odin.control.toronto.edu/DES/.

[21] T. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. In *Proc. of WODES 2000*, p. 111–118, Ghent, Belgium, Aug 2000.

[22] Z.H. Zhang and W.M. Wonham. STCT: an efficient algorithm for supervisory control design. In *Proc. of SCODES 2001*, INRIA, Paris, July 2001.

[23] H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. on Automatic Control*, 35(10):1125–1134, Oct 1990.