

A. NOORULDEEN

**SUPERVISORY CONTROL FOR RECONFIGURABLE
MANUFACTURING SYSTEMS:
STRUCTURAL CHANGES AND RE-USABILITY OF
CONTROLLERS**

ANAS NOORULDEEN

ÇANKAYA UNIVERSITY

ANKARA, 2012

**SUPERVISORY CONTROL FOR RECONFIGURABLE
MANUFACTURING SYSTEMS: STRUCTURAL CHANGES AND
RE-USABILITY OF CONTROLLERS**

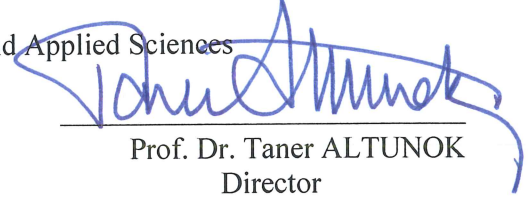
ANAS NOORULDEEN

SEPTEMBER, 2012


Title of the Thesis: Supervisory Control for Reconfigurable Manufacturing Systems: Structural Changes and Re-usability of Controllers

Submitted By: **Anas NOORULDEEN**


Approval of the Graduate School of Natural and Applied Sciences


Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.


Prof. Dr. Celal Zaim ÇİL
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.


Assoc. Prof. Dr. Klaus Werner SCHMIDT
Supervisor


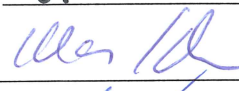

Examination Date : September 21, 2012

Examining Committee Members

Dr. Zeki Uğurata Kocabıykoğlu (Çankaya Univ.)

Assoc Prof. Dr. Klaus Werner SCHMIDT (Çankaya Univ.)

Mustafa Sancay Kırık (TÜBİTAK Uzay ODTÜ)

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Anas NOORULDEEN

Signature :



Date :

21.09.2012

ABSTRACT

SUPERVISORY CONTROL FOR RECONFIGURABLE MANUFACTURING SYSTEMS: STRUCTURAL CHANGES AND RE-USABILITY OF CONTROLLERS

NOORULDEEN, Anas

M.Sc., Department of Electronic and Communication Engineering

Supervisor: Assoc. Prof. Dr. Klaus Werner SCHMIDT

September 2012, 79 Pages

This thesis deals with reconfigurable manufacturing systems (RMS) and reconfigurable machine tools (RMT), that are designed to provide flexibility in both the variety of products and the configuration of the manufacturing system itself in order to quickly adjust to new products and production. In particular, the thesis investigates the aspect of reconfigurable manufacturing systems to change their structure during run-time. To this end, methods for the synthesis of controllers that support structural changes of the RMS are developed. In addition, the topic of re-usability of previously designed controllers is addressed. The methods are applied to a laboratory model of an RMS.

Keywords: Reconfigurable Manufacturing Systems, Reconfigurable Machine Tools, Discrete Event Systems, Controller Design, Structural Changes, Re-usability

ÖZ

YENİDEN YAPILANDIRILABİLİR ÜRETİM SİSTEMLERİ İÇİN
DENETLEYİCİ KONTROLU: YAPISAL DEĞİŞİKLİKLER VE
KONTROLCULERİN TEKRAR KULLANILABİLİRLİĞİ

NOORULDEEN, ANAS

Yüksek Lisans, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Tez Yöneticisi: Doç. Dr. Klaus Werner SCHMIDT

Eylül 2012, 79 Sayfa

Bu tez yeniden yapılandırılabilir üretim sistemleri (reconfigurable manufacturing systems - RMS) ve yeniden yapılandırılabilir makine araçları ile ilgilidir, yeni ürünlere ve üretime hızlı bir şekilde uyum sağlamak için ürünlerin çeşitliliğini ve üretim sistemlerinin esnekliğini her iki alanda da desteklemek için tasarımı edilmiştir. Özellikle, bu tez yeniden yapılandırılabilir üretim sistemlerinin yapısını çalışırken değiştirilmesini incelemektedir. Bu amaçla, kontrolcülerin sentez metodları yeniden yapılandırılabilir üretim sistemlerindeki yapısal değişiklikleri desteklemek için geliştirilmiştir. Ek olarak, önceden tasarlanmış olan kontrolcülerin yeniden kullanılabilirliği örneklenmiştir. Bu metodlar yeniden yapılandırılabilir üretim sistemlerine laboratuvar modeli olarak uygulanabilir.

Anahtar Kelimeler: Yeniden Yapılandırılabilir Üretim Sistemleri, Yeniden Yapılandırılabilir Makine Araçları, Ayrık Olay Sistemleri, Kontrolcu Tasarımı, Yapısal değişiklikleri, Tekrar Kullanılabilirlik

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Assoc. Prof. Dr. Klaus Werner Schmidt. Also, I am very grateful to him, for his leading role in the support, continued encouragement and guidance to achieve the correct end during the course of the study and thesis.

I also thank and appreciate the Scientific and Technological Research Council of Turkey (TÜBİTAK), in the provision of material and moral support represented by financial support (during my thesis) and dissemination of results in international conferences and indexed journals. Note, that this thesis was supported by TÜBİTAK [Carrier Award 110E185].

And also, I thank Çankaya University, and especially the Department of Mechatronics Engineering, as well as the Department of Electronic and Communication Engineering, for their support and for creating the appropriate requirements of laboratories and equipment necessary during periods of study and research.

Finally, all the love and respect to my beloved mother and dear father for their love, support and encouragement during my studies.

TABLE OF CONTENTS

STATEMENT OF NON-PLAGIARISM	iii
ABSTRACT	iv
ÖZ	v
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	vii
CHAPTERS	
INTRODUCTION	1
1 BASIC NOTATION	3
1.1 Discrete Event Systems	3
1.1.1 Formal Language	4
1.1.2 Automata	5
1.2 Supervisory Control	9
1.3 State Attraction	11
2 STRUCTURAL CHANGES OF RECONFIGURABLE MANUFACTURING SYSTEMS	13
2.1 Manufacturing System Overview	13
2.2 Reconfigurable Manufacturing Systems and Reconfigurable Machine Tools	14
2.3 Structural Changes of Reconfigurable Machine Tools	17
2.4 Supervisory Control of Reconfigurable Manufacturing Systems with Structural Changes	24
3 APPLICATION EXAMPLE	34
3.1 Modeling of Reconfigurable Machine Tools	34
3.1.1 Conveyor Belt	35

3.1.2	Machine Head	36
3.1.3	Machine Tools	37
3.1.4	Overall Model	37
3.1.5	Safety Specifications for the RMT	38
3.2	Supervisory Control of RMTs	42
3.3	Modeling of Reconfigurable Manufacturing Systems	48
3.3.1	Stack Feeder	49
3.3.2	Rotary Table	50
3.3.3	Exit Slides	51
3.4	Supervisory Control of RMS	52
3.5	Re-usability of Controllers	58
3.6	RMS Simulation	61
	CONCLUSIONS	63
	REFERENCES	64
	APPENDIX	66

LIST OF TABLES

TABLES

Table 1.1	Design procedure for the supervisory control problem.	11
Table 1.2	Design procedure for the supervisory control problem.	12

LIST OF FIGURES

FIGURES

Figure 1.1	Example finite state automaton G	7
Figure 2.1	Overview of a simple RMS with rotary table (RT) and a re- configurable machine tool (RMT)	16
Figure 2.2	Schematic of the example RMT.	18
Figure 2.3	Component models of the example RMT.	19
Figure 2.4	Configuration specification and supervisors.	19
Figure 2.5	Optimal supervisors for state attraction T^D ($x_0^D = 1$) and T^M ($x_0^M = 4$).	23
Figure 2.6	Supervisors \bar{S}^1 and \bar{S}^2	23
Figure 2.7	Supervisor \bar{S}	24
Figure 2.8	Plant automata for the small RMS example.	25
Figure 2.9	Configurations of the RMS.	26
Figure 2.10	Example for strong attraction under language specification.	27
Figure 2.11	Example solution supervisor for Problem 1.	28
Figure 2.12	Automaton C'_3 according to Algorithm 1.	31
Figure 2.13	Automaton R'_3 such that $L_m(R'_3) = SupC(K_3, G_3, \Sigma_u)$	32
Figure 2.14	State-feedback supervisor S_3 according to Theorem 2.	33
Figure 3.1	RMT with three different modes of operation.	35
Figure 3.2	Picture of a conveyor belt.	35
Figure 3.3	Conveyor belt model G_{cb}	36

Figure 3.4	Machine head model G_{mh}	36
Figure 3.5	Model of the turning operation of the machine head.	37
Figure 3.6	Models of the RMT operation.	38
Figure 3.7	Specification automata: C_{Spec1} , C_{Spec2} and C_{Spec3}	39
Figure 3.8	Specification automata C_{Spec4} and C_{Spec5}	40
Figure 3.9	Specification automaton C_{Spec6}	40
Figure 3.10	Specification automata C_{Spec7} , C_{Spec8} , C_{Spec9} and C_{Spec10}	41
Figure 3.11	High-level abstraction S_{RMT}^{high} of the RMT.	42
Figure 3.12	Specification per configurations of RMT.	43
Figure 3.13	Supervisor per configurations of RMT.	43
Figure 3.14	Reconfiguration supervisor \bar{S}^D	44
Figure 3.15	Reconfiguration supervisor \bar{S}^P	45
Figure 3.16	Reconfiguration supervisor \bar{S}^M	46
Figure 3.17	Abstaction of \bar{S} of RMT.	47
Figure 3.18	Reconfiguration supervisor S^{rec} , high plant of RMT $G_{RMTplant}$	47
Figure 3.19	Diagram of the example RMS.	48
Figure 3.20	Picture of the stack feeder.	49
Figure 3.21	An automaton model G_{sf} for the stack feeder.	50
Figure 3.22	High level stack feeder model G_{sf}	50
Figure 3.23	Picture of the rotary table.	50
Figure 3.24	Rotary table model G_{rb}	51
Figure 3.25	Picture of the exit slide.	52
Figure 3.26	Exit slide models G_{xs1} , G_{xs2} and G_{xs1}	52
Figure 3.27	RMS system plant G_{RMS}	53
Figure 3.28	Configurations specification of RMS.	54
Figure 3.29	Supervisor per configuration specification of RMS.	55
Figure 3.30	Components \bar{S}_{conf1} of the reconfiguration supervisor.	56
Figure 3.31	Components \bar{S}_{conf2} of the reconfiguration supervisor.	57
Figure 3.32	Language specifications for reconfiguration.	57

Figure 3.33	Supervisors for state attraction under language specification.	58
Figure 3.34	Specification and supervisor for milling configuration of RMS.	59
Figure 3.35	Components \overline{S}_{conf3} of the reconfiguration supervisor.	60
Figure 3.36	Coordinator of RMS.	61
Figure 3.37	Schematic of the RMS setup.	62
Figure 3.38	Simulation by controller synthesis.	62

INTRODUCTION

Future manufacturing systems are expected to produce more alternative products to meet the evolving needs of the market due to progress of academic research and industrial practice. Traditionally, manufacturing systems are realized as *dedicated manufacturing lines* or *flexible manufacturing systems* in order to either achieve a high product quality at high volumes and low cost in a dedicated manufacturing plant or to be able to produce a variety of different products on the same machines. The common factor for such types of a manufacturing systems is that, they used fixed structure to produce their product. For this reason, these manufacturing systems are not keeping pace with the advancement of manufacturing systems and purpose of the market need. Reconfigurable manufacturing systems (RMS) were introduced as a new paradigm in manufacturing [4, 5, 6, 10] to address this problem. The new perspective of RMS is to adapt and respond to changing market demands quickly and to effectively increase production capacity and functionality if required. As a result, the purpose of an RMS is to reduce lead-time for launching new products, the rapid manufacturing modification, the quick integration and easy adaptation to new functions and technologies.

To this end, RMSs must fulfill several requirements such as:

- the reconfiguration of components of the RMS, denoted as reconfigurable machine tools (RMT)
- the change of product specifications
- the change of production plans
- the modification of the manufacturing system layout.

The subject of this thesis is the development of methods for the control of RMS in order to fulfill the stated requirements. First, this thesis studies the applica-

tion of an existing method [13] for the control of RMTs to a practical example. This method is based on the supervisory control theory for discrete event systems (DES) by Ramadge and Wonham [12]. The RMT plant is modeled by a finite state automaton and each configuration of the RMT is described by a *reconfiguration specification* language. Each reconfiguration, that is, each change from one configuration to another configuration, is represented by a *reconfiguration event* that can occur at any time. Then, the method allows to fulfill the specification of the new configuration after a bounded reconfiguration delay, whenever a reconfiguration event occurs. In addition, the thesis develops a new method for the structural change of RMSs. It combines ideas from [13] and develops new ideas, that are specific to RMSs. As addition to [13], the new method requires that an additional specification has to be fulfilled while moving to a new configuration. In the thesis, this problem is defined as state attraction under language specification. The thesis states necessary and sufficient conditions for the solution of this problem and also presents an algorithm for the computation of a suitable controller. This algorithm is further applied to a laboratory model of an RMS. It is also shown in the thesis that controllers for reconfiguration can be re-used if a new configuration is added to a running RMS.

The thesis is organized as follows. Chapter 1 gives basic notation regarding discrete event systems (DES) in Section 1.1, regarding the supervisory control of DES in Section 1.2, and regarding state attraction for DES in Section 1.3. Chapter 2 presents the structural changes of reconfigurable manufacturing systems. It consists of an overview of manufacturing systems in Section 2.1, the description of reconfigurable manufacturing systems and reconfigurable machine tools in Section 2.2, a control method for structural changes of reconfigurable machine tools in Section 2.3, and the new method for supervisory control of RMS in Section 2.4. Chapter 3 shows a laboratory application example of an RMS. That is, the modeling of reconfigurable machine tools are presented in Section 3.1, the supervisory control of RMTs are described in Section 3.2, the modeling of reconfigurable manufacturing systems are presented in Section 3.3, the supervisory control of RMSs are elaborated in Section 3.4, and Section 3.5 discusses the re-usability of controller. Finally, Chapter 3.6 contains the conclusions.

CHAPTER I

BASIC NOTATION

1.1 Discrete Event Systems

The expression "discrete event system" (DES) was introduced in the early 1980s to model an increasingly important class of human-made dynamic systems with a discrete state space, and state transitions that are given by the asynchronous occurrence of discrete events. Such systems are encountered in a variety of fields and have been successfully employed in many areas, for example in manufacturing systems and communication networks [3, 18]. The operation of a DES is in principle governed by sequences of events, whereby the order of events is most relevant. The exact timing of events is nevertheless not important. A DES model can hence very well describe systems with activities such as turning some device "On ,Off", sending one or multiple message packets, or detecting if an object arrives at a certain location.

DES can be characterized by the following distinctive properties:

- A DES has a finite or countably infinite number of discrete states
- Each change of state occurs at a discrete time instant. State changes are called *transitions*
- Each transition is driven by the occurrence of an event
- A DES spends time only in states. That is, state transitions occur instantaneously (they do not need time).

A simple example of a DES is a light switch. The switch has two discrete states, which can be identified as "On" and "Off". There are further two possible events `switchOn` and `switchOff`. If the light switch is for example in state "Off", the event `switchOn` can occur, and the light switch performs an instantaneous transition to the state "On". Analogously, in state "On", a transition with event `switchOff` leads to state "Off".

In the established literature, the behavior of a DES is modeled by a formal language [3]. We next introduce the concept of a formal language.

1.1.1 Formal Language

The notion of language is introduced in discrete event systems for modeling the logical behavior of a DES. Let Σ be the finite set of events, also called *alphabet*. This set consists of all the events that can possibly happen in a given DES. A *string* (or trace) is a finite sequence of events from Σ . The length of a string s , denoted by $|s|$ is given by the number of events in s . The empty string, denoted by ϵ , is the string with zero length (i.e., $|s| = 0$). The set of all possible finite strings of events from a finite alphabet Σ (including ϵ) is denoted as Σ^* , whereby the \star -operation is called *Kleene Closure*.

Considering the light switch example, the alphabet is given by $\Sigma = \{\text{switchOn}, \text{switchOff}\}$. A possible sequence of events is $s = \text{switchOn } \text{switchOff } \text{switchOn}$ with length $|s| = 3$. The Kleene closure of Σ in this case is $\Sigma^* = \{\epsilon, \text{switchOn}, \text{switchOff}, \text{switchOn switchOn}, \text{switchOn switchOff}, \text{switchOff switchOn}, \text{switchOff switchOff}, \dots\}$. Note that not all strings in Σ^* must be possible in the DES.

The concatenation of two strings $s_1 \in \Sigma^*$ and $s_2 \in \Sigma^*$ is the string $s_1 s_2$ (that is, s_2 is attached to the end of s_1). If there exist strings $s_1, s_2 \in \Sigma^*$ such that $s = s_1 s_2$, then we write $s_1 \leq s$ and s_1 is called a prefix of s . A subset $L \subseteq \Sigma^*$ is called a *language* over Σ . An important language operation is the *prefix closure*. The prefix closure of L is written \overline{L} and contains all prefixes of strings of L . This

means

$$\overline{L} = \{s \in \Sigma^* \mid \exists t \in \Sigma^* \text{ such that } st \in L\}$$

A language L fulfilling $L = \overline{L}$ is called prefix closed.

Now, Let $\hat{\Sigma} \subseteq \Sigma$. The *natural projection* erases all events in $\Sigma \setminus \hat{\Sigma}$ (\setminus is the set difference) from strings $s \in \Sigma^*$. This function is defined as $p : \Sigma^* \rightarrow \hat{\Sigma}^*$ with

$$\begin{aligned} p(\epsilon) &= \epsilon \\ p(\sigma) &= \begin{cases} \sigma & \text{if } \sigma \in \hat{\Sigma} \\ \epsilon & \text{otherwise} \end{cases} \\ p(s\sigma) &= p(s)p(\sigma) \end{aligned}$$

For example, if we are only interested in the event `switchOn` of the light switch, we can project each string to the alphabet $\hat{\Sigma} = \{\text{switchOn}\}$ and make occurrences of `switchOff` invisible. For the string $s = \text{switchOn switchOff switchOn}$, the projection gives $p(s) = \text{switchOn switchOn}$.

1.1.2 Automata

A very widely used compact model for languages is the automaton. It allows to describe and study the structural behavior of DES. Usually, a DES is modeled by a finite state automaton. It is a 5-tuple of the form $G = (X, \Sigma, \delta, x_0, X_m)$, where

- X is a finite set of *states*;
- Σ is a finite set of *events*;
- $\delta : X \times \Sigma \rightarrow X$ is a *partial transition function*;
- $x_0 \in X$ is the *initial state*; It is state where the system begins the process.
- $X_m \subseteq X$ is the set of *marked states*; It is desired states that must be reached after finishing a task.

A finite state automaton $G = (X, \Sigma, \delta, x_0, X_m)$, also called generator, is characterized by two subset languages of Σ^* :

- *Closed language* $L(G)$:

$$L(G) = \{s \in \Sigma^* \mid \delta(x_0, s) \text{ exists}\}$$

- *Marked language* $L_m(G)$:

$$L_m(G) = \{s \in L(G) \mid \delta(x_0, s) \in X_m\}$$

The language $L(G)$ contains all event sequences that follow the transitions of G starting from the initial state. The language $L_m(G)$ contains all strings of events, starting from the initial state of G and leading to a marked state in X_m .

A finite state automaton G is said to be *nonblocking* if

$$\overline{L_m(G)} = L(G)$$

This property holds, when every string generated by G can be extended to a marked (desired) state in G . Finite state automata can be *cyclic* or *acyclic*. A cycle in a finite automaton is a sequence of states x_1, x_2, \dots, x_k (k is a natural number) such that $x_1 = x_k$ and for all $i = 1, \dots, k - 1$, there exists an event $\sigma_i \in \Sigma$ such that $\delta(x_i, \sigma_i) = x_{i+1}$. This means, it is possible to start at a state x_1 of G , follow the transitions in G and return back to x_1 . Then, an automaton G without cycles is called acyclic.

Now we introduce several relevant properties and operations for automata:

- *Accessible*

The automaton $G = (X, \Sigma, \delta, x_0, X_m)$ is *accessible*, if all states in X can be reached from the initial state x_0 . Formally, we write

$$\forall x \in X, \exists s \in \Sigma^* \text{ such that } \delta(x_0, s) = x$$

In addition, we write $Acc(G)$ for the automaton, where all states from G that are not reachable from x_0 are removed. Then, $Acc(G)$ is accessible.

- *Coaccessible*

The automaton $G = (X, \Sigma, \delta, x_0, X_m)$ is *coaccessible*, if it is possible to reach a marked state from any state in X . Formally, we write

$$\forall x \in X, \exists s \in \Sigma^* \text{ such that } \delta(x, s) \in X_m$$

It holds that a coaccessible automaton is nonblocking: $\overline{L_m(G)} = L(G)$. In addition, we write $CoAcc(G)$ for the automaton, where all states from G that are not coaccessible are removed. Then, $CoACC(G)$ is coaccessible.

- *Trim*

The automaton $G = (X, \Sigma, \delta, x_0, X_m)$ is *trim*, if it is accessible and coaccessible at the same time. We write

$$Trim(G) = Acc(CoAcc(G)) = CoAcc(Acc(G))$$

Next, we introduce the notion of *subautomaton*. Let $G = (X, \Sigma, \delta, x_0, X_m)$ and $G' = (X', \Sigma, \delta', x'_0, X'_m)$ be finite state automata. G' is a *subautomaton* of G , if $X' \subseteq X$, $x'_0 = x_0$ and for all $x \in X'$ and $\sigma \in \Sigma$, it holds that $\delta'(x, \sigma) \neq \emptyset \Rightarrow \delta'(x, \sigma) = \delta(x, \sigma)$ [8]. This means, G' is obtained from G by removing states and transitions. In this case, we write $G' \sqsubseteq G$ if G' is a subautomaton of G . G' is a *strict subautomaton* of G if additionally $\delta(x, \sigma) \in X' \Rightarrow \delta'(x, \sigma) = \delta(x, \sigma)$. This means that only states are removed from G to obtain a strict subautomaton G' . We use the example automaton $G = (X, \Sigma, \delta, x_0, X_m)$ in Fig. 1.1 to explain the notation introduced before.

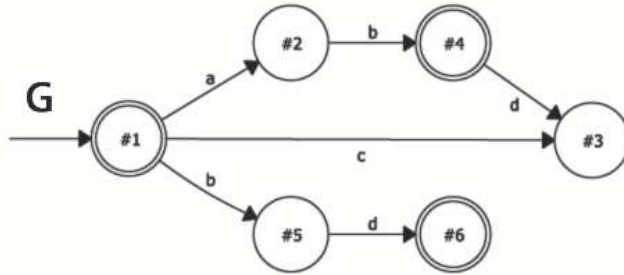


Figure 1.1: Example finite state automaton G .

G has 6 states with $X = \{1, 2, 3, 4, 5, 6\}$ and an alphabet with 4 events $\Sigma = \{a, b, c, d\}$. The transition relation δ is defined with $\delta(1, a) = 2$, $\delta(1, b) = 5$,

$\delta(1, c) = 3$, $\delta(2, b) = 4$, $\delta(4, d) = 3$, $\delta(5, d) = 6$. The initial state is $x_0 = 1$ and the set of marked states is $X_m = \{1, 4, 6\}$. We can further determine the closed language $L(G) = \{\epsilon, a, ab, abd, c, b, bd\}$ and the marked language $L_m(G) = \{\epsilon, ab, bd\}$. Moreover, the automaton G is accessible because all states in X are reachable from the initial state 1. Hence, $Acc(G) = G$. However, G is not coaccessible because there is no path from the state 3 to a marked state. Hence $CoAcc(G)$ is obtained by removing state 3 from G . Then, $CoAcc(G)$ is a strict subautomaton of G . It can also be seen that G is acyclic.

If a DES is modeled by more than one finite state automaton (for example, if the system has several components with their own automata model), the *synchronous composition* operation can be used to obtain a single automaton model of the DES. Assume two automata $G_1 = (X_1, \Sigma_1, \delta_1, x_{0,1}, X_{m,1})$ and $G_2 = (X_2, \Sigma_2, \delta_2, x_{0,2}, X_{m,2})$ are given. The synchronous composition is written as:

$G_{12} = (X_{12}, \Sigma_{12}, \delta_{12}, x_{0,12}, X_{m,12}) = G_1 || G_2$, and is defined such that

- $X_{12} = X_1 \times X_2$ (canonical product of states from X_1 and X_2)
- $\Sigma_{12} = \Sigma_1 \cup \Sigma_2$ (union of events in Σ_1 and Σ_2)
- $x_{0,12} = (x_{0,1}, x_{0,2})$
- $X_{m,12} = X_{m,1} \times X_{m,2}$
- the transition relation takes care that events in $\Sigma_1 \cap \Sigma_2$ that are shared by G_1 and G_2 are synchronized. For $(x_1, x_2) \in X_{12}$ and $\sigma \in \Sigma_{12}$:

$$\delta_{12}((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \wedge \delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)! \\ (\delta_1(x_1, \sigma), x_2) & \text{if } \sigma \in \Sigma_1 \setminus \Sigma_2 \wedge \delta_1(x_1, \sigma)! \\ (x_1, \delta_2(x_2, \sigma)) & \text{if } \sigma \in \Sigma_2 \setminus \Sigma_1 \wedge \delta_2(x_2, \sigma)! \end{cases}$$

That is, the important point of the synchronous composition is that it synchronizes the shared events (events that appear in both automata), whereas all other events can occur independent of each other.

Until now, we described how a DES can be modeled by formal languages and finite state automata, respectively. The main task of this thesis is the control of

such DES, that is the design of a controller such that the DES assumes a pre-specified desired behavior. In this thesis, two frameworks for the control of DES are needed. The supervisory control theory for DES is introduced in the next section and the control for state attraction is explained in Section 1.3.

1.2 Supervisory Control

The supervisory control theory for a DES was first established by Ramadge and Wonham [12]. It offers a formal framework to design and implement control for DES. The control of the system is executed by allowing or preventing specific events from occurring in the plant. Such control is performed by a controller (or supervisor) while taking into consideration the necessity to ensure the desired behavior of the system.

Assume $G = (X, \Sigma, \delta, x_0, X_m)$ is a finite state automaton DES representing the plant which must be controlled. Then, the alphabet of the plant is divided into two disjoint subsets as follows

$$\Sigma = \Sigma_c \cup \Sigma_u$$

Here, Σ_c is the set of controllable events. These events can be disabled by a controller (supervisor) any time. Examples for controllable events are actuator events such as turning on a motor or machine. Σ_u is the set of uncontrollable events. These events can never be disabled by a supervisor. Examples for uncontrollable events are sensor events such as detecting if a product arrives at a certain location.

A supervisor for a DES plant G can also be realized by a finite state automaton $S = (Q, \Sigma, \nu, q_0, Q_m)$. In that case, the closed loop (controlled) system is given by the synchronous composition of the plant G and the supervisor S : $G||S$. Then, the closed loop languages of $G||S$ are $L_m(G)||L_m(S)$ (marked language) and $L(G)||L(S)$ (closed language). As remarked before, a supervisor is never allowed to disable uncontrollable events. Because of this reason, it must hold for all $s \in L(G) \cap L(S)$ and $\sigma \in \Sigma_u$ with $s\sigma \in L(G)$ that also $s\sigma \in L(S)$. This means, if an uncontrollable event σ can occur in the plant after a string s , then it must

also occur in the supervisor after s . A supervisor S is called nonblocking if the automaton $G||S$ is nonblocking.

The design of the supervisor S is based on the desired behavior of the plant. This desired behavior is usually given in the form of another automaton $C = (Y, \Sigma, \beta, y_0, Y_m)$ and $K = L_m(C)$ is called the *specification* language for the control problem. The meaning of K is, that it contains all strings that are desirable. On the other hand, a supervisor should disable all strings in $L(G)$ that do not belong to K . The question if this task is possible is answered by the *controllability* condition. The specification K is *controllable* with respect to G and $\Sigma_u \subseteq \Sigma$ if it satisfies

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$$

In this expression, \overline{K} it is prefix-closure of K and $\overline{K}\Sigma_u$ represents the set of all strings that start with a prefix in \overline{K} concatenated with an uncontrollable event in Σ_u . In words, K is *controllable* with respect to G and Σ_u if, whenever a string $s \in \overline{K}$ that is allowed by the specification can be extended by an uncontrollable event in the plant ($s\sigma \in L(G)$) the extended string must again belong to the specification ($s\sigma \in \overline{K}$). This is necessary, because σ could not be disabled by a supervisor after s . It is a very basic result from supervisory control theory that there exists a nonblocking supervisor S such that $L_m(G||S) = K$ if and only if K is controllable with respect to G and Σ_u .

In case, the specification K cannot be realized by a supervisor (this means that K is not controllable with respect to G and Σ_u), the supervisory control theory suggests to find the largest possible sublanguage $K_{sub} \subseteq K$ such that K_{sub} is controllable with respect to G and Σ_u . The supremal element of all controllable sublanguages of K is given as:

$$SupC(K, G, \Sigma_u) = \bigcup \{K' \subseteq K \mid K' \text{ is controllable with respect to } G \text{ and } \Sigma_u\}$$

That is, $SupC(K, G, \Sigma_u)$ includes all sublanguages of K that are controllable with respect to G and Σ_u . If K_{sub} is not empty, there is a supervisor S such that $L_m(G||S) = K_{sub}$ that can be computed algorithmically [12] also with existing software tools [9]. This supervisor is nonblocking and is also called *maximally*

permissive, because it enables the largest possible sublanguage of K .

In summary, the supervisory control theory allows to algorithmically compute a supervisor S for a DES plant G such that a given language specification is fulfilled, which means that $L_m(G||S) \subseteq K$. The design problem is summarized in the following table.

Table 1.1: Design procedure for the supervisory control problem.

Given	Desired	Solution
Plant G Specification K Uncontrollable events Σ_u	Find supervisor S such that $L_m(G S) \subseteq K$	Compute $SupC(K, G, \Sigma_u)$ Use supervisor S with $L_m(G S) = SupC(K, G, \Sigma_u)$

It has to be noted that the supervisory control problem as described above is not the only control problem for DES that is relevant in this thesis. The next section describes the problem of state attraction, which is particularly useful for reconfiguration control of DES.

1.3 State Attraction

We consider a finite state automaton $G = (X, \Sigma, \delta, x_0, X_m)$ as DES plant and an uncontrollable event set Σ_u . We call a subset $X' \subseteq X$ an *invariant set* in G if there is no transition from any state in X' to a state outside X' . Formally, $\forall x \in X'$ and $\sigma \in \Sigma$ it must hold that $\delta(x, \sigma)! \Rightarrow \delta(x, \sigma) \in X'$ [1].

In addition, we call a set $X' \subseteq X$ a *weakly invariant set* if only transitions with controllable events leave X' , that is, $\forall x \in X'$ and $\sigma \in \Sigma_u$ it holds that $\delta(x, \sigma)! \Rightarrow \delta(x, \sigma) \in X'$ [13].

Based on these definitions, we can introduce the notion of *strong attraction* and *weak attraction* from [1, 2].

Definition 1 Let $A \subseteq X' \subseteq X$ and assume that A, X' are invariant sets in G . Then, A is denoted as a strong attractor for X' in G if

- the strict subautomaton of G with the state set $X' \setminus A$ is acyclic

- $\forall x \in X'$, there is $u \in \Sigma^*$ such that $\delta(x, u) \in A$

Moreover, A is denoted as a weak attractor for X' in G with Σ_u if there exists a state-feedback supervisor $S \sqsubseteq G$, such that A is a strong attractor for X' in S .

In words, A is a strong attractor for X' in G if, starting from any state of G , a finite number of transitions always leads to the invariant set A . Adding control to the attraction problem, weak attraction is defined such that there must be a supervisor S such that the closed loop $G||S$ becomes a strong attractor.

Regarding the computation of a supervisor for weak attraction, we refer to [1, 2]. It is shown, that there is a set $\Omega_G(A) \subseteq X$, that denotes the supremal subset of X such that A is a weak attractor for $\Omega_G(A)$ in G with Σ_u . The set $\Omega_G(A)$ can be computed with complexity $\mathcal{O}(|X| \cdot |\Sigma|)$, whereby $|X|$ and $|\Sigma|$ denote the number of states and events of G , respectively. In addition, a supervisor $S \sqsubseteq G$ such that A is a strong attractor for $\Omega_G(A)$ in S can be computed by the algorithm in [2] with complexity $\mathcal{O}(|X|^2)$.

In summary, the problem of state attraction requires to find a supervisor S such that the closed loop $G||S$ ensures reaching a desired set of states A after a bounded number of transitions. Again, the relevant design algorithms are available in software tools such as [9]. The following table explains the design procedure for state attraction.

Table 1.2: Design procedure for the supervisory control problem.

Given	Desired	Solution
Plant G Desired states A Uncontrollable events Σ_u	Find supervisor such that $G S$ reaches A after bounded number of transitions	Compute $\Omega_G(A)$ Use supervisor S such that the state set of $G S$ is equal to $\Omega_G(A)$

CHAPTER II

STRUCTURAL CHANGES OF RECONFIGURABLE MANUFACTURING SYSTEMS

In this chapter, we develop the main results of this thesis for the control of reconfigurable manufacturing systems. Based on background on general manufacturing systems, we study two main ideas: the reconfiguration of single machines by change of system structure and the reconfiguration of an entire manufacturing system by change of the control algorithm. Regarding the contribution of the thesis, the first idea is developed in previous work [13] however applied to a large example in this thesis. The second idea is developed together with application example in the scope of this thesis and also prepared as a journal paper [11].

The chapter is organized as follows. Section 2.1 gives an overview of manufacturing systems. Section 2.2 introduces *reconfigurable manufacturing systems* (RMS), and the *reconfigurable machine tool* (RMT) as the main component of RMS. In Section 3.3, we introduce a framework for modeling RMTs and RMSs supported by an application example. The structural changes of RMTs are considered in Section 2.3 using the model from Section 3.3. Section 2.4 presents a new method for the controller design of RMSs.

2.1 Manufacturing System Overview

Traditional manufacturing systems are represented by dedicated manufacturing lines (DML) or flexible manufacturing systems (FMS). DML are optimized to

produce a large quantity of a single product. As a result they are efficient in producing their dedicated product but it is very difficult to change the operation of a DML for producing different products. FMS are designed to produce different pre-defined products in varying quantities. That is, FMS are flexible in the sense that they use the same machines for different products. However, since the products are pre-defined when the FMS is installed, it is again difficult to change its operation. In summary, the current dominant manufacturing systems have the disadvantage that they have a fixed structure in order to produce their pre-defined product(s) and without any support for changes in the system [6, 10].

In the recent years, conventional manufacturers are suffering from a new situation on the market. It is characterized by the requirement of fast introduction of new products and quick changes in the product quantity at low cost, while maintaining the quality of manufacturing. The current manufacturing technologies do not keep pace with these advanced requirements of the market need since they are optimized for fixed operation.

In the late 1990s, the concept of *reconfigurable manufacturing systems* (RMS) was introduced by Koren et. al. as a solution to the stated problem. We next give a more detailed description of RMS.

2.2 Reconfigurable Manufacturing Systems and Reconfigurable Machine Tools

According to Koren et. al. [10],

Changing manufacturing environment characterized by aggressive competition on a global scale and rapid changes in process technology requires to create production systems that are themselves easily upgradable and into which new technologies and new functions can be readily integrated.

Based on this observation, several desired properties of RMS can be listed as follows:

- It must be possible to quickly change the operation of the system

- It must be possible to add new different products to the production during run-time
- It must be possible to vary the production volume and capacity
- It must be possible to rapidly modify the functionality of the system (for example by adding new machine components)
- It must be possible to easily integrate new functions and technologies
- The cost of the system must be reasonable.

In order to achieve the listed properties, it is necessary to modify or re-arrange the components of an RMS [10]. Components of an RMS are for example the machines, conveyors, sensors, machine tools but also the algorithms that control the operation of the RMS. This means, research on RMS requires mechanical design (machines and tools), electronic design (sensors and actuators) as well as software design (controllers and control algorithms).

Regarding the mechanical design, the main building block of an RMS is the *reconfigurable machine tool* (RMT). An RMT is designed to be able to perform different operations by changing its mechanical configuration. RMTs are supposed to have a modular structure such that components can easily and quickly be added or modified during system operation. The mechanical design of such RMTs is not the subject of this thesis. However the thesis will use the concept of RMTs for system modeling.

Regarding the electronic design, new sensor technologies such as for example RFID allow to attach product specific information to each individual product and manufacture the product based on this information. This thesis does not focus on the electronic design, however uses the features offered by advanced sensor solutions in the RMS modeling process.

Regarding the control of manufacturing systems, it first has to be mentioned that manufacturing systems are usually composed of many different manufacturing components. Each such component (for example a conveyor belt) has its own functionality that is controlled by a *local* controller and attached directly to the

manufacturing component. Such controller is simply responsible for the basic operation of the component, that is the same controller will be used in a DMS, FMS or RMS. The interesting control problem for RMS is given by:

- the need to change the basic mode of operation of RMTs in order to manufacture different products
- the need to coordinate the operation of different manufacturing components of an RMS.

The stated requirements can only be fulfilled if it is possible to change the control program during system operation, which is the main subject of this thesis. We next present a small RMS example in order to illustrate the previously presented ideas. The RMS is shown in Figure 2.1. It consists of an RMT and a so-called rotary table. The task of the rotary table is to transport products from the input direction toward the RMT. This is represented by the arrows `in` and `toRMT`. In addition, the rotary table receives products from the RMT (`toRT`) and can then move a product either up (`out1`) or down (`out2`). The RMT has two modes of operation. It either receives a product from the rotary table (`toRMT`), processes in mode `op1` and delivers the product back to the RT (`toRT`) or it receives a product, processes in mode `op2` and delivers back to the rotary table. In the

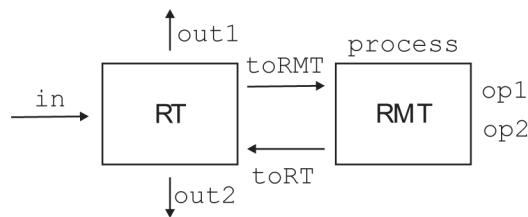


Figure 2.1: Overview of a simple RMS with rotary table (RT) and a reconfigurable machine tool (RMT)

example, the basic property of the RMT is, that can operate in two different modes. A reconfiguration of the RMT is needed to change the mode of operation. In addition, the overall system can be considered as an RMS. Just imagine that different products are produced depending on the mode of operation. Then, it might be desired to deliver different products to different locations. For example,

we could have configuration 1, where the product is processed in mode `op1` of the RMT and has to be delivered to `out1`. In configuration 2, the product has to be processed in mode `op2` of the RMT and delivered to `out2`. Then, two basic questions regarding the operation of the RMS can be asked:

1. How to perform the reconfiguration of the RMT in order to change from `op1` to `op2` and vice versa?
2. How to perform the reconfiguration of the whole RMS to change from configuration 1 to configuration 2 and vice versa if required?

Regarding question 1, we study the control of structural changes of RMTs for reconfiguration in Section 2.3. Here, we use the combination of a state attraction problem as in Section 1.3 (to control the system while changing configuration) and a classical supervisory control problem as in Section 1.2 (to determine the control for each fixed configuration). Considering question 2, we study the change of the desired system operation for RMSs in Section 2.4.

2.3 Structural Changes of Reconfigurable Machine Tools

We describe the reconfiguration problem for RMTs as introduced in [13]. In principle, it is desired to design a supervisor that can perform fast reconfiguration of the RMT structure. Furthermore, we point out that the resulting supervisor enables a modular realization, which makes it possible to add a new configuration quite easily.

As we mentioned, the described method is based on a DES model of the RMT plant. The behavior of an RMT is modeled by a finite state automaton $G = (X, \Sigma, \delta, x_0, X_m)$ and a set of uncontrollable events Σ_u . Different configurations of the RMT are represented by a set of *configurations* \mathcal{C} , and for each configuration $\rho \in \mathcal{C}$, a *configuration specification* $K^\rho \subseteq \Sigma^*$ and a *configuration start state* $x_{st,\rho} \subseteq X$ are introduced. The meaning of K^ρ and $x_{st,\rho}$ for some configuration $\rho \in \mathcal{C}$ is that the realization of configuration ρ requires to fulfill the specification K^ρ starting from state $x_{st,\rho}$ of the plant G .

The basic setup is illustrated by the following simplified example of a machine that can assume two different configurations D and M, that is $\mathcal{C} = \{D, M\}$. A schematic of the RMT is shown in Fig. 2.2. In configuration D, the RMT operates as a drill. A product can enter the RMT, which is associated with the event **in**. Then, the machine head (MH) moves down (event **toOp**) and the RMT performs the drilling task (**process**). Afterwards, the machine head moves back to its rest position (**toRest**), and the product can leave the RMT (**toRT** – it is assumed that the neighboring component is a rotary table). In configuration M the RMT operates as a mill. In this configuration, the basic operation is analogous to configuration D, whereby the machine head is now turned to a different position in order to perform milling.

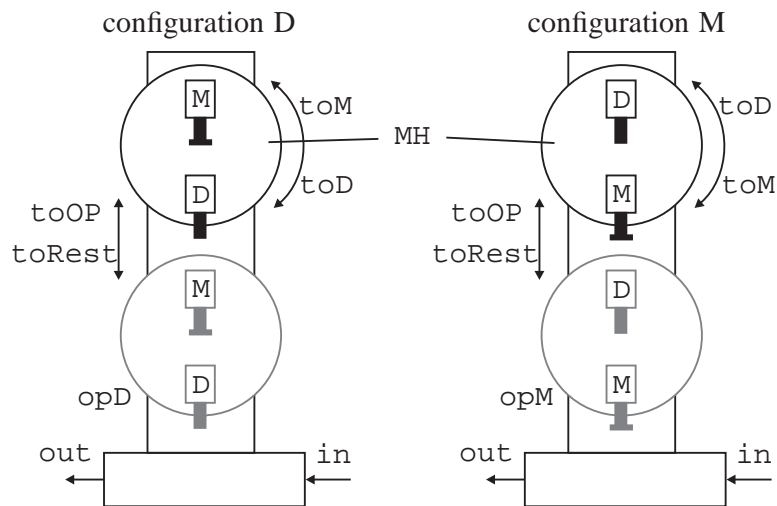


Figure 2.2: Schematic of the example RMT.

The components of the uncontrolled plant model of the RMT are given in Fig. 2.3. G_{IO} describes the input/output of products (**toRMT**, **toRT**), G_V describes the vertical motion of the machine head (**toOp**, **toRest**), G_R describes the rotation of the machine head (**op2**, **op1**) and G_M and G_D model the milling and drilling operation, respectively (**process**). In addition, Fig. 2.3 shows the basic RMT behavior.

The configurations D and M now look as follows. Configuration D should start from state 1 of G and requires that **process** is executed whenever a product

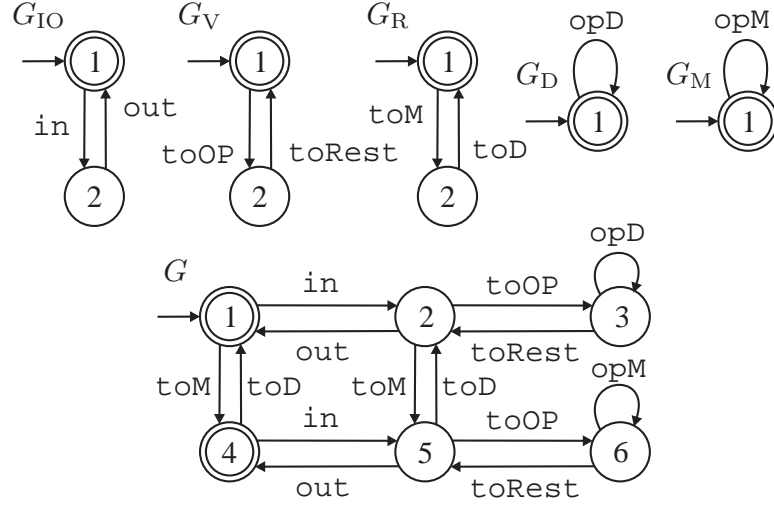


Figure 2.3: Component models of the example RMT.

arrives. Likewise, configuration M should start from state 4 of G such that opM happens after product arrival. The configuration specifications K^D and K^M can be written down in the form of automata C^D and C^M in Fig. 2.4 such that $L_m(C^D) = K^D$ and $L_m(C^M) = K^M$. If supervisors for the two configurations

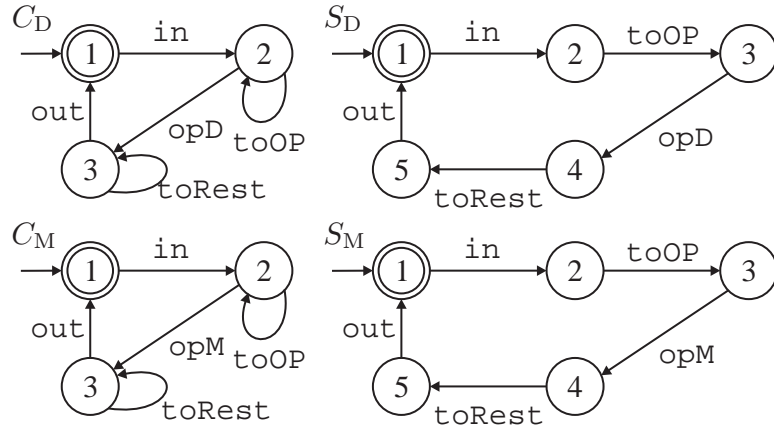


Figure 2.4: Configuration specification and supervisors.

should be computed, we need to fulfill the specification starting from the specified start state. Because of this reason, the plant automaton for configuration $\rho \in \mathcal{C}$ with the initial state $x_{\text{st},\rho}$ is written as $G^\rho = (X, \Sigma, \delta, x_{\text{st},\rho}, X_m)$. Using this plant with modified start state, it is possible to compute the maximally permissive

configuration supervisor S^ρ that achieves the specification for configuration $\rho \in \mathcal{C}$ such that $L_m(S^\rho) = \text{SupC}(K^\rho, L(G^\rho), \Sigma_u)$.

Figure 2.4 also shows the configuration supervisors S^D and S^M . Note that these supervisors are computed using the plants G^D (initial state 1) and G^M (initial state 4), respectively.

From the described procedure up to now, it is seen that the operation of each individual configuration and the computation of the corresponding supervisor is clear. Only a classical supervisory control problem has to be solved for this purpose. However, it is not enough to run the supervisors for the separate configurations. It also has to be determined how to change from one configuration to the other configuration whenever such reconfiguration is requested. As an important point, it has to be considered that the new configuration might not be achieved immediately but only after a reconfiguration delay. In the method of [13], the described reconfiguration task is formulated and solved as a state attraction problem. We next describe the solution, since it is most relevant for the application example in this thesis.

In addition to the previous notation, a *reconfiguration start event* ρ_{st} and a *reconfiguration finish event* ρ_{fin} for each $\rho \in \mathcal{C}$ is introduced. The new alphabets $\Sigma_{\text{st}} = \bigcup_{\rho \in \mathcal{C}} \{\rho_{\text{st}}\}$, $\Sigma_{\text{fin}} = \bigcup_{\rho \in \mathcal{C}} \{\rho_{\text{fin}}\}$, $\Sigma^{\text{rec}} = \Sigma \cup \Sigma_{\text{st}} \cup \Sigma_{\text{fin}}$ and $\Sigma_u^{\text{rec}} = \Sigma_u \cup \Sigma_{\text{st}}$ are used, and all new events are added to the plant automaton G in the form of selfloops. That is, an extended plant $G^{\text{rec}} = (X, \Sigma^{\text{rec}}, \delta^{\text{rec}}, x_0, X_m)$ is defined such that for each $x \in X$ and $\sigma \in \Sigma$

$$\delta(x, \sigma) = x' \Rightarrow \delta^{\text{rec}}(x, \sigma) = x'$$

and for each $x \in X$ and $\sigma \in \Sigma_{\text{st}} \cup \Sigma_{\text{fin}}$

$$\delta^{\text{rec}}(x, \sigma) = x$$

The reconfiguration start events are used to indicate whenever a change of configuration is requested. Since it is desired to allow such request at any time, these events are uncontrollable. The reconfiguration finish events are introduced to show when the transition period between two configurations is finished and are

considered as controllable events. Finally, we introduce the set of plant states X^ρ that are reachable in each configuration as follows:

$$X^\rho = \{x \in X \mid \exists s \in \Sigma^* \text{ s.t. } x = \delta(x_{\text{st},\rho}, s) \text{ and } s \in L(S^\rho)\}$$

Using the above notation and recalling the notion of weak attraction in Section 1.3, Theorem 1 states necessary and sufficient conditions for the existence of a solution to the described reconfiguration problem.

Theorem 1 *A solution supervisor S^{rec} for the reconfiguration problem exists if and only if for each $\rho \in \mathcal{C}$ and $\rho' \in \mathcal{C}$ it holds that $X^\rho \subseteq \Omega_G(\{x_{\text{st},\rho'}\})$.*

In this theorem, $\Omega_G(\{x_{\text{st},\rho'}\})$ represents the set of states that are attracted to the initial state of configuration ρ' . That is, the theorem means that, whenever the RMT follows a configuration ρ (which means the RMT is in one of the states of X^ρ , then it must be possible to move to the initial state $x_{\text{st},\rho'}$ of a new configuration ρ' in a bounded number of steps (which is true if any state in X^ρ is attracted by $x_{\text{st},\rho'}$).

[13] not only gives this existence result but also a procedure to construct a suitable reconfiguration supervisor S^{rec} . Since this supervisor is used to perform structural changes of RMTs in this thesis, the solution algorithm is presented now. It is suggested to compute the reconfiguration supervisor as the synchronous composition of one supervisor $\bar{S}^\rho = (\bar{Q}^\rho, \Sigma^{\text{rec}}, \bar{\nu}^\rho, \bar{q}_0^\rho, \bar{Q}_m^\rho)$ for each configuration $\rho \in \mathcal{C}$ and one coordinating supervisor $\bar{S} = (\bar{Q}, \Sigma^{\text{rec}}, \bar{\nu}, \bar{q}_0, \bar{Q}_m)$, that is,

$$S^{\text{rec}} = \bar{S} \parallel (\parallel_{\rho \in \mathcal{C}} \bar{S}^\rho) \quad (2.1)$$

For each $\rho \in \mathcal{C}$, the supervisor \bar{S}^ρ is used to enforce configuration ρ if it is active and to switch to a *waiting state* Wait^ρ otherwise. It is defined by

$$\bar{Q}^\rho = Q^\rho \cup \{\text{Wait}^\rho\} \text{ and } \bar{Q}_m^\rho = Q_m^\rho \cup \{\text{Wait}^\rho\},$$

for each $q \in Q^\rho$ and $\sigma \in \Sigma$

$$\nu^\rho(q, \sigma) = q' \Rightarrow \bar{\nu}^\rho(q, \sigma) = q',$$

for each $q \in Q^\rho$ and $\rho' \in \mathcal{C}$

$$\bar{\nu}^\rho(q, \rho'_{\text{st}}) = \text{Wait}^\rho,$$

for each $\sigma \in \Sigma^{\text{rec}} \setminus \{\rho_{\text{fin}}\}$

$$\bar{\nu}^\rho(\text{Wait}^\rho, \sigma) = \text{Wait}^\rho$$

$$\bar{\nu}^\rho(\text{Wait}^\rho, \rho_{\text{fin}}) = q_0^\rho.$$

Finally, we set $\bar{q}_0^\rho = q_0^\rho$ if $\rho = \rho_0$ and $\bar{q}_0^\rho = \text{Wait}^\rho$ otherwise.

The supervisor \bar{S} is used to follow the actual state of the plant G and to govern the change between different configurations. Let $T^\rho = (W^\rho, \Sigma, \omega^\rho, -, -)$ be the minimally restrictive optimal supervisor such that $\{x_{\text{st},\rho}\}$ is a strong attractor for $W^\rho = \Omega_G(\{x_{\text{st},\rho}\})$ in T^ρ . Then, we define the state set of \bar{S} such that it contains the set X and a copy of each state in W^ρ for all $\rho \in \mathcal{C}$:

$$\bar{Q} = X \cup \left(\bigcup_{\rho \in \mathcal{C}} \{x^\rho \mid x \in W^\rho\} \right) \text{ and } \bar{Q}_{\text{m}} = X_{\text{m}},$$

for each $x \in X$ and $\sigma \in \Sigma$

$$\delta(x, \sigma) = x' \Rightarrow \bar{\nu}(x, \sigma) = x',$$

for each $\rho \in \mathcal{C}$ and $x \in X^\rho$

$$\bar{\nu}(x, \rho_{\text{st}}) = x^\rho,$$

for each $x \in X$, $\rho \in \mathcal{C}$ and $\rho' \in \mathcal{C}$

$$x^\rho \in \bar{Q} \text{ and } x^{\rho'} \in \bar{Q} \Rightarrow \bar{\nu}(x^{\rho'}, \rho_{\text{st}}) = x^\rho,$$

for each $\rho \in \mathcal{C}$

$$\bar{\nu}(x_{\text{st},\rho}^\rho, \rho_{\text{fin}}) = x_{\text{st},\rho}.$$

Now, we illustrate the supervisor construction by continuing the previous example. To this end, we first evaluate the minimally restrictive optimal supervisors T^{D} and T^{M} as depicted in Fig. 2.5.

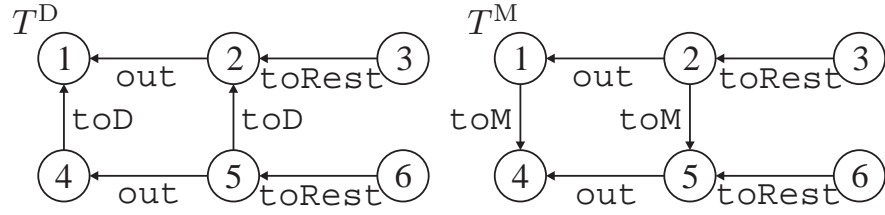


Figure 2.5: Optimal supervisors for state attraction T^D ($x_0^D = 1$) and T^M ($x_0^M = 4$).

Using the result in Figure 2.5, the components of the supervisor S^{rec} are constructed as shown in Fig. 2.6 and 2.7. It should be noted that the transitions with reconfiguration start events between the states in T^D and T^M are not shown in Fig. 2.7 for the sake of clarity. According to (2.1), the overall reconfiguration supervisor is given by $S^{\text{rec}} = \bar{S} \parallel \bar{S}^D \parallel \bar{S}^M$. That is, the resulting supervisor can be represented by the synchronous composition of modular supervisors that either perform the task of operating one configuration or switching between configurations.

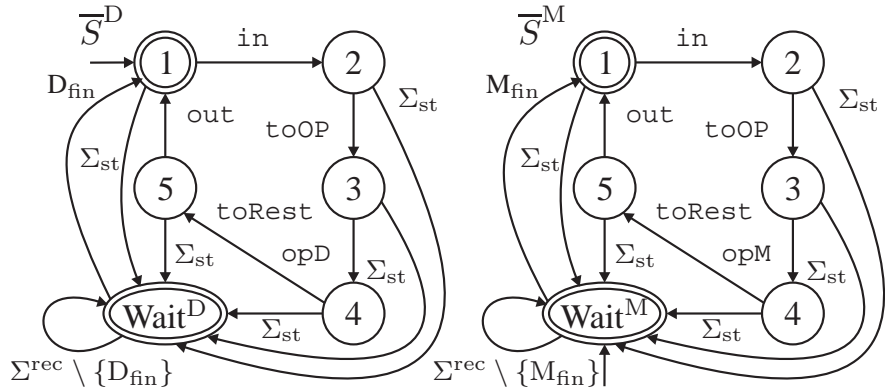


Figure 2.6: Supervisors \bar{S}^1 and \bar{S}^2

We now give a brief summary of the reconfiguration supervisor functionality. The operation of S^{rec} is as follows. If a configuration $\rho \in \mathcal{C}$ is active, the component \bar{S}^ρ follows the configuration supervisor S^ρ and switches to the waiting state Wait^ρ as soon as the configuration becomes inactive (when a reconfiguration is requested). During reconfiguration, all supervisors \bar{S}^ρ for $\rho \in \mathcal{C}$ are in their waiting state. The task of the supervisor \bar{S} is to follow the current plant state if some configuration

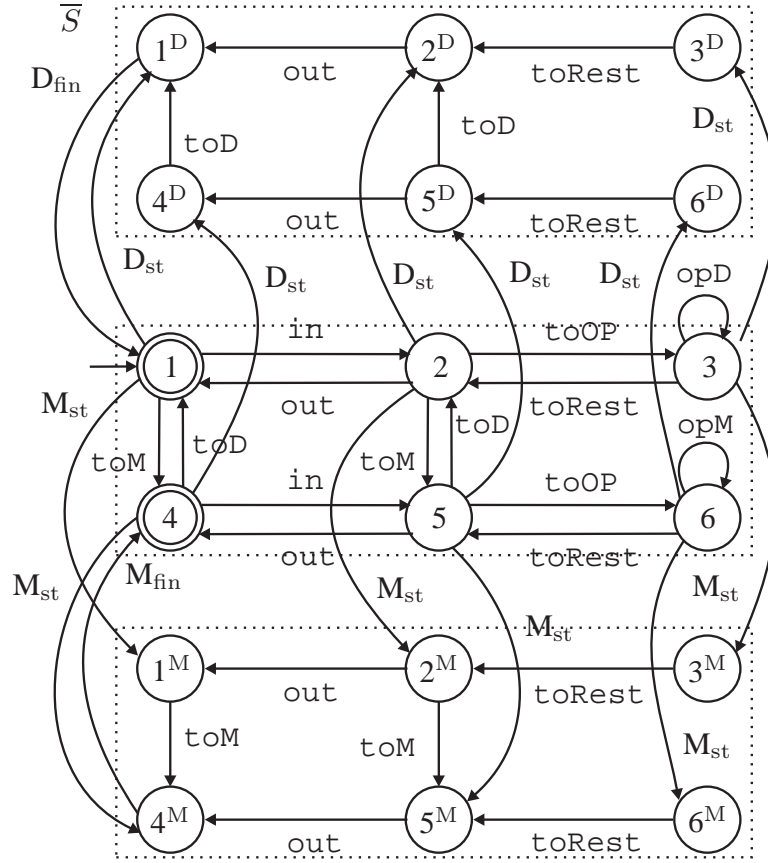


Figure 2.7: Supervisor \bar{S} .

is active. In case of reconfiguration to $\rho \in \mathcal{C}$, \bar{S} switches to the optimal supervisor for state attraction T^ρ that moves the plant state to the initial state for the new configuration as fast as possible. As soon as this initial state is reached, the new configuration becomes active, and \bar{S} again follows the current plant state. Hence, the components of S^{rec} are constructed to clearly separate the task of operating a certain configuration and the task of reconfiguration.

2.4 Supervisory Control of Reconfigurable Manufacturing Systems with Structural Changes

The previous section describes how the reconfiguration of RMTs can be performed by supervisory control. Since an RMS could in principle be considered as an RMT with many different components, it would be natural to assume that the method

for RMTs in the previous section can also be applied to RMSs. We show in this section, that this assumption is generally not true. Because of this reason we develop a new method for the reconfiguration of RMSs. The identification of the reconfiguration problem for RMSs and the solution procedure are a contribution of this thesis, whereas the detailed mathematical formulation and proofs were prepared in the scope of the journal manuscript [11]. We now explain the reconfiguration problem for RMSs supported by the small example from Section 2.2 in Figure 2.1.

The uncontrolled plant operation of the RMS components is modeled by the automata as shown in Fig. 2.8. G_{RT} describes the entry and exit of products from and to RT, which happens with the events ($out1, out2, in1,$ and $toRMT$). G_{RMT} shows different processing operations for products of the RMT as follows. After a product passes from RT to RMT, it is processed ($process$) and returns back to RT with event $toRT$. Reconfiguration with operations ($op1$ or $op2$ are possible in the idle state and after processing. This means, the RMT performs processing according to its current configuration. The overall uncontrolled behavior of the plant $G = G_{RT} || G_{RMT}$ is also shown in Fig. 2.8.

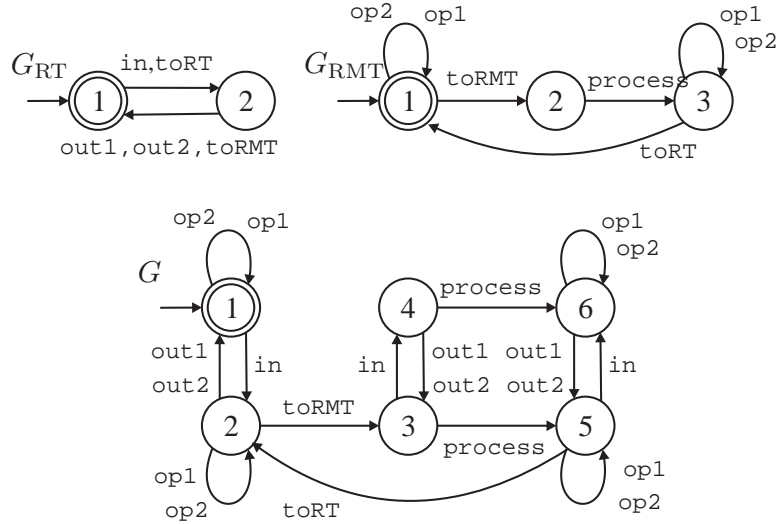


Figure 2.8: Plant automata for the small RMS example.

As was previously explained in Section 2.2, we suppose two types of operation of the RMT to process two types of products. One of the operations is labeled

as configuration 1 – processing by RMT, whereby the last reconfiguration event was op1. The other operation is labeled as configuration 2 – processing by RMT, whereby the last reconfiguration event was op2. Fig. 2.9 described the two different operations.

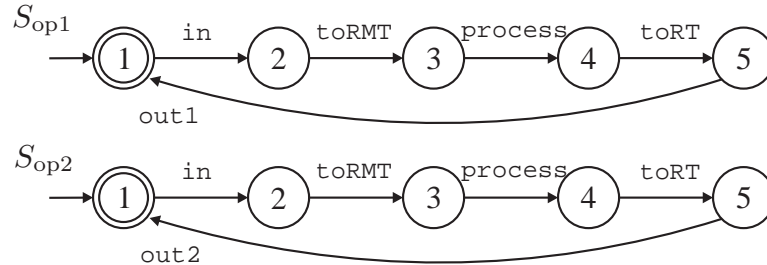


Figure 2.9: Configurations of the RMS.

Similar to the reconfiguration problem for RMTs, the operation of the different configurations is clear and the main question is how to perform the reconfiguration. In Section 2.3, the reconfiguration for RMTs could be solved by moving the state of the system to a state, where the new configuration could start its operation. According to Theorem 1, this problem corresponds to a state attraction problem. We now show that the situation is different in the case of RMSs. Consider again the example plant model in Figure 2.8. When reconfiguring from configuration 1 to configuration 2, we would want to move the system state to the initial state as fast as possible in order to start configuration 2. This is still in line with the previously discussed reconfiguration for RMTs. However, at the same time, we would want that the event op2 occurs, such that the RMT changes its structure for the new mode of operation. This is a new requirement of the design problem in the form of a language specification, that has to be fulfilled while moving to the initial state. Together, the reconfiguration needs:

1. to move to a desired subset on the state space of the DES plant with a bounded number of event occurrences as fast as possible
2. to fulfill the desired language specification while moving to the desired subset.

Problem 1 is again a state attraction problem, whereas problem 2 is a classical

supervisory control problem. That is, for the first time in the literature, we need to solve the combination of a state attraction problem (reconfiguration must start from initial state of new configuration) and a classical supervisory control problem (reconfigure the structure of the RMT by the occurrence of **op1** or **op2**).

We formulated the requirements for the problem solution by the concepts of *strong attraction under language specification* in the following definition.

Definition 2 Let $G = (X, \Sigma, \delta, x_0, X_2)$ be an automaton, $A \subseteq X$ and assume that A is an invariant set in G . In addition, let $K \subseteq \Sigma^*$ be a specification language and $x \in X$ be a state. Write $G_x = (X_x, \Sigma, \delta, x, X_{2,x})$ for the automaton G , where the initial state is replaced by x , X_x is the set of states reachable from x and $X_{2,x} = A \cap X_x$. Then, A is denoted as a strong attractor for x in G with K if

1. the strict subautomaton of G_x with the state set $X_x \setminus X_{m,x}$ is acyclic
2. $\forall x \in X_x$, there is a $u \in \Sigma^*$ such that $\delta(x, u) \in X_{m,x}$
3. $\forall u \in \Sigma^*$ such that $\delta(x, u) \in X_{m,x}$ and $\delta(x, u') \notin X_{m,x}$ for $u' < u$, it holds that $u \in K$.

This means, starting from x , we reach the set $X_{m,x} \subseteq A$ in a finite number of steps and fulfill the specification K when reaching A . For example, consider Fig. 2.10. The state set $A_{S_3} = \{7\}$ is a strong attractor for state 1 in S_3 with the specification $K = L_m(C)$: S_3 is acyclic in $\{1, 2, 3, 4, 5, 6\}$ and on every path from state 1 to state 7, the event **op1** occurs once, whereas **op2** does not happen.

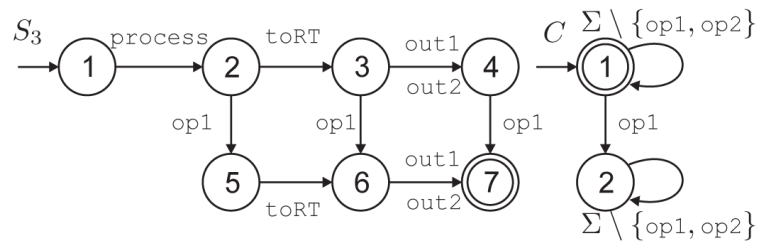


Figure 2.10: Example for strong attraction under language specification.

Now, we plan to apply the supervisory control for satisfying the conditions in Definition 2. Suppose that $S_x = (Q_x, \Sigma, \nu_x, q_{0,x}, Q_{m,x})$ is a supervisor for G_x and Σ_u . We also introduce the closed-loop automaton $R_x = (Z_x, \Sigma, \alpha_x, z_{0,x}, Z_{m,x}) = G_x || S_x$ of the system and the invariant state set $A_{R_x} = \{(x', q') \in Z_x | x' \in A \text{ and } q' \in Q_{m,x}\} \subseteq Z_x$. If R_x reaches a state in the set A_{R_x} , then, at the same time G_x reaches a state in the invariant set A . We now formulate the supervisory control problem using these notions.

Problem 1 *Assume that G_x , A and K are given as in Definition 2. Let Σ_u be a set of uncontrollable events. We want to find a supervisor S_x for G_x and Σ_u such that A_{R_x} is a strong attractor for Z_x in the closed-loop automaton R_x with K .*

Problem 1 captures what we want to achieve in the motivating example. It requires that the invariant set A is reached after a bounded number of transitions (which happens if A_{R_x} is reached), whereby the specification K is fulfilled. That is, S_x ensures reaching invariant set while satisfying K . S_3 in Fig. 2.10 achieves this task for state $x = 3$ of the plant G in Fig. 2.8 with the invariant set $A = \{1\} \subseteq X$, the specification $K = L_m(C)$ in Fig. 2.10 and the set of uncontrollable events $\Sigma_u = \{\text{process}\}$. The closed-loop automaton $R_3 = G_3 || S_3$ is shown in Fig. 2.11.

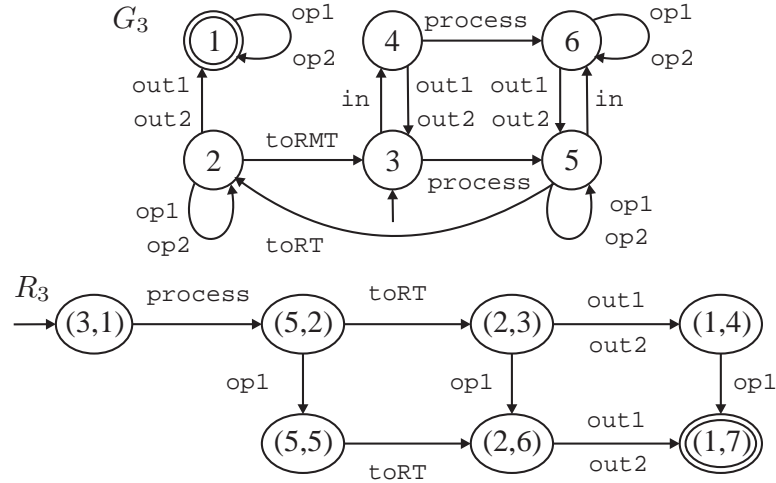


Figure 2.11: Example solution supervisor for Problem 1.

We would like to use the solution in [1, 2, 7] to compute the supervisor in Problem 1. However this is not possible. The state attraction solution assumes that the

closed-loop system R_x has the same state space as the plant G_x . This means that the supervisor S_x should be a subautomaton of G_x . It can be seen for example in Figure 2.11 that this need not be the case. Hence, a new algorithm for state attraction under language specification is needed.

As we mentioned, Problem 1 requires the combination of a supervisory control problem (in order to fulfill K) and a state attraction problem (in order to move to A). Accordingly, we solve Problem 1 in two steps. First, we develop an algorithm to construct a supervisor, that only fulfills 2 & 3 in Definition 2. Then, we remove potential cycles in this supervisor such that also 1) in Definition 2 is fulfilled.

We develop our algorithm for an arbitrary plant state $x \in X$. In order to address the first step of the problem solution, we define the language K_x of all strings that lead from x to A and such that the shortest prefix of each string in K_x fulfills K :

$$K_x := \{s \in L(G_x) \mid \delta(x, s) \in A \text{ and } s \in K \text{ but } \delta(x, s') \notin A \text{ or } s' \notin K \text{ for all } s' < s\} \cdot \Sigma^*$$

K_x contains all shortest possible strings that lead to the invariant set A and at the same time fulfill K . It follows that Problem 1 cannot have a solution if K_x is empty. This means that the solution of Problem 1 – if it exists – must enable a subset of K_x that is controllable for G_x and Σ_u . In particular, this implies that $SupC(K_x, G_x, \Sigma_u) \neq \emptyset$. We can formulate this observation in the following lemma. Note that the proof of this Lemma is found in [11].

Lemma 1 *Let G_x, A, Σ_u, K_x be given as defined above. Assume that the supervisor S_x solves Problem 1. Then, it holds that $SupC(K_x, G_x, \Sigma_u) \neq \emptyset$.*

Lemma 1 confirms that Problem 1 can have a solution only if $SupC(K_x, G_x, \Sigma_u) \neq \emptyset$.

Since the condition in Lemma 1 is necessary for the existence of a solution for Problem 1, it would be nice to have an algorithm that checks the condition. To this end, we first represent K_x by a finite state automaton. We assume that the automaton $C = (Y, \Sigma, \beta, y_0, Y_2)$ is chosen such $L_m(C) = K$. Then, we apply a modified synchronous composition operation to G_x and C . This operation is

defined in Algorithm 1. In principle, it follows that classical synchronous composition operation, however terminates if a state is found both in A and Y_m starting from the initial state (x, y_0) . The result of the computation is an automaton $C'_x = (Y'_x, \Sigma, \beta'_x, y'_{0,x}, Y'_{m,x})$ such that $L_m(C'_x) = K_x$.

Algorithm 1 **Input:** G_x, A, C ; **Output:** C'_x

1. *Initialize: $Wait = \{(x, y_0)\}$, $Done = \emptyset$; $Y'_x = \{(x, y_0)\}$, $y'_{0,x} = (x, y_0)$, β'_x is empty*
2. *Take some state (x', y') from $Wait$ and insert in $Done$*
3. *Compute the set \mathcal{S} of one-step successor states of (x', y') according to the classical synchronous composition*
4. *Insert the states in \mathcal{S} and the transitions from (x', y') to states in \mathcal{S} into the result generator C'_x*
5. *Set each state $(x'', y'') \in \mathcal{S}$ such that $x'' \in A$ and $y'' \in Y_2$ marked in C'_x ; introduce a selfloop with all events in Σ for all such states.*
6. *Insert all states in $\mathcal{S} \setminus (Done \cup Y'_{m,x})$ into $Wait$*
7. *If $Wait \neq \emptyset$, go back to 2*
8. *Terminate with the result C'_x .*

That is, Algorithm 1 starts from the initial state of the parallel composition $G_x || C$ and terminates at all states that correspond to strings which lead to the invariant set A and at the same time fulfill the specification K . All such terminal states are marked in C'_x , and after reaching such terminal state, all strings in Σ^* are allowed. It is readily observed that $L_m(C'_x) = K_x$.

After applying Algorithm 1 to the plant G in Fig. 2.8 with the starting state $x = 3$, the invariant set $A = \{1\}$ and the specification $K = L_m(C)$ (Figure 2.10), we get automaton C'_3 in Fig. 2.12. We see that every path from the initial state $(3,1)$ can be extended to the marked (terminate) state and each path that leads to the marked state fulfills the specification K , That is, any string from the initial

state (3,1) to (1,2) contains the event op1 . (1,2) is the only marked state of C'_x and receives a selfloop with all events in Σ .

Using G_x and C'_x , the condition in Lemma 1 can be easily verified by computing $\text{SupC}(L_m(C'_x), G_x, \Sigma_u)$. Considering that C'_x has a state space that is a subset of the state space of $G_x||C$, the computational complexity for this verification is $\mathcal{O}(|X|^M \cdot |Y|^M)$.

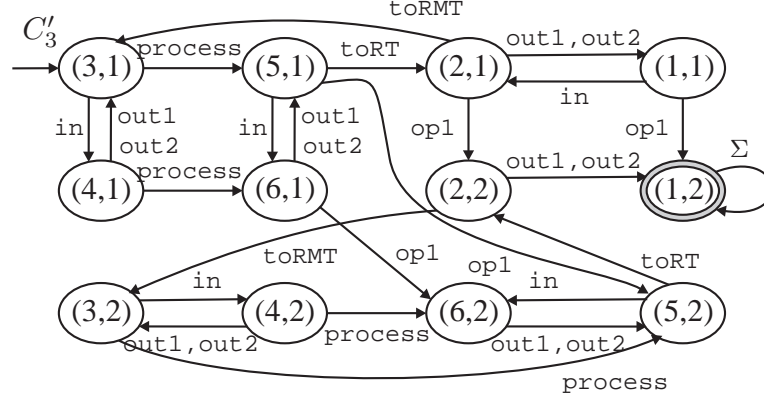


Figure 2.12: Automaton C'_3 according to Algorithm 1.

Since we want to solve Problem 1, we have to assume that the condition in Lemma 1 is fulfilled, that is, there is a supervisor S'_x such that $L_m(G_x||S'_x) = \text{SupC}(K_x, G_x, \Sigma_u)$. If we have such supervisors, two properties of the closed loop $G_x||S'_x$ can be concluded as shown in the following lemma. The proof of the lemma is given in [11].

Lemma 2 *Assume that S'_x is a supervisor such that $L_m(G_x||S'_x) = \text{SupC}(K_x, G_x, \Sigma_u)$ and write $R'_x = (Z'_x, \Sigma, \alpha'_x, z'_{0,x}, Z'_{m,x}) := G_x||S'_x$. Then, it holds that*

1. $s \in L_m(R'_x) \Rightarrow \delta(x, s) \in A$
2. $s \in L_m(R'_x)$ and $s' \notin L_m(R'_x)$ for all $s' < s \Rightarrow s \in K$.

That is, any string that leads to a marked state in the closed loop R'_x leads to the invariant set A in G_x . In addition, any string that first enters the marked states of R'_x also belongs to K . Using Lemma 2, we can directly conclude that the

conditions 2 & 3 in Definition 2 are already fulfilled by R'_x . Condition 2) follows from the fact that $R'_x = G_x || S'_x$ is nonblocking by construction. Then, for any state $z \in Z'_x$, there is a string $u \in \Sigma^*$ such that $\alpha'_x(z, u) \in A_{R'_x} = Z'_{m,x}$. Condition 3 holds because of 2 in Lemma 2.

We now look at our example again to illustrate the previous statements. R'_3 in Fig. 2.13 is found using C'_3 in Fig. 2.12. It can be seen that all strings that lead to the marked state $(1,2)$ in R'_3 lead to the invariant set $A = \{1\}$ in G_3 . In addition, the specification K is fulfilled for all strings that lead to $(1,2)$. This confirms also for the example that conditions 2 & 3 in Definition 2 are already fulfilled. Unfortunately, it can also be observed that R'_3 still contains cycles outside the attractive set $A_{R'_3} = \{(1,2)\}$. This contradicts the first condition in Definition 2. To resolve this problem, we now find a state-feedback supervisor $S_x = (Q_x, \Sigma, \nu_x, q_{0,x}, Q_{m,x}) \sqsubseteq R'_x$ such that $Z'_{m,x}$ is a strong attractor for $Q_x = \Omega_{R'_x}(Z'_{m,x}) \subseteq Z'_x$ in S_x using the algorithms in [1, 7]. It is then sufficient to check if the initial state $(x, q'_{0,x})$ of R'_x belongs to Q_x .

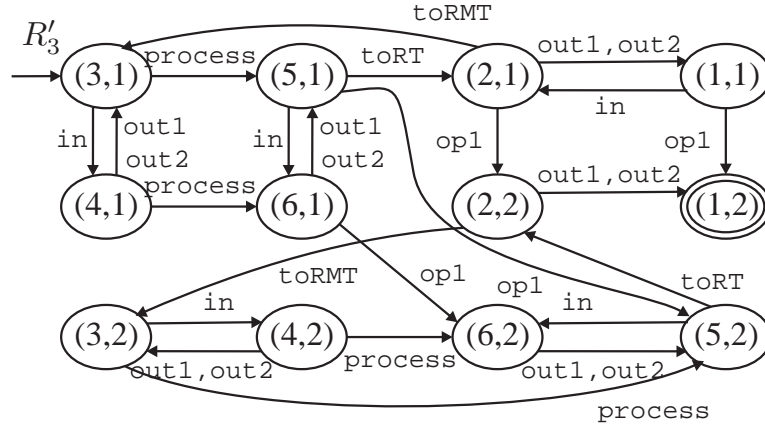


Figure 2.13: Automaton R'_3 such that $L_m(R'_3) = SupC(K_3, G_3, \Sigma_u)$.

Theorem 2 *Let G be an automaton, $A \subseteq X$ an invariant set in G , $x \in X$ a state, and Σ_u a set of uncontrollable events. In addition, let $K \subseteq \Sigma^*$ be a specification language with its recognizer C such that $L_m(C) = K$. Assume that C'_x is constructed according to Algorithm 1 and S'_x is a supervisor such that $L_m(R'_x) = SupC(L_m(C'_x), G_x, \Sigma_u)$ for $R'_x = G_x || S'_x$. Problem 1 has a solution if and only if $z'_{0,x} = (x, q'_{0,x}) \in Q_x = \Omega_{R'_x}(Z'_{m,x})$. In that case, any state-feedback supervisor*

$S_x \sqsubseteq R'_x$ such that $Q_x = \Omega_{R'_x}(Z'_{m,x})$ and $Z'_{m,x}$ is a strong attractor for Q_x in S_x can be used.

Hence, the basic procedure to compute a suitable supervisor for the solution of Problem 1 is to first determine the automaton C'_x as described in Algorithm 1. Then, the classical supervisory control theory is used to find a supervisor S'_x . The last step is the solution of a state attraction problem for the resulting closed loop $R_x = G_x || S'_x$. The first two steps are done with a computational complexity of $\mathcal{O}(|X|^2 \cdot |Y|^2)$ (see previous discussion). The last step has a computational complexity of $\mathcal{O}(|X|^2 \cdot |Y|^2)$ as can be seen from Section 1.3. Hence, the overall complexity is $\mathcal{O}(|X|^2 \cdot |Y|^2)$. It is very important to note that the result in Theorem 2 and the related Algorithm 1 are entirely new.

It is now possible to apply the results in Theorem 2 to our RMS example. We need to find the supremal set $\Omega_{R'_3}(Z'_{2,3})$ for the automaton R'_3 in Fig. 2.13 and the invariant set $Z'_{2,3} = \{(1,2)\}$. It turns out that all states are attractable, that is, $\Omega_{R'_3}(Z'_{2,3}) = Z'_3$. As well, we find a state-feedback supervisor S_3 according to 2, that moves the plant state to the invariant set while fulfilling the specification K . It is shown in Fig. 2.14 (see shaded states). The final supervisor for implementation in Problem 1 is equal to S_3 in Fig. 2.11 that was found by intuition before.

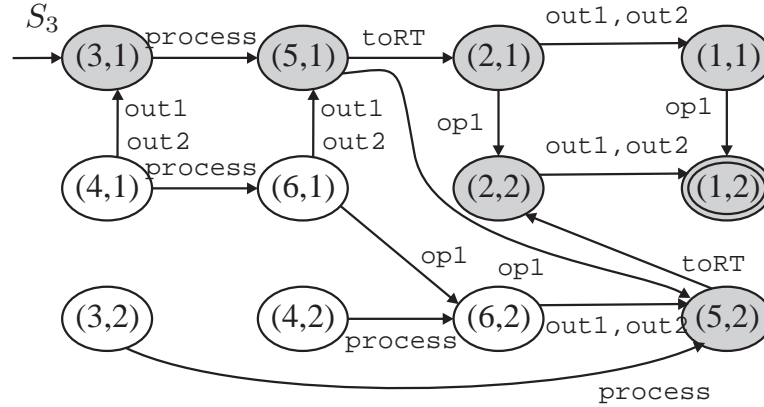


Figure 2.14: State-feedback supervisor S_3 according to Theorem 2.

CHAPTER III

APPLICATION EXAMPLE

In this section, we apply the methods introduced in Section 2 to a laboratory model of a RMS. We first develop a detailed model of a RMT in Section 3.1. This model is then used to apply the supervisory control solution presented in Section 2.3. Next, we extend the RMT model in order to form a RMS in Section 3.3. Finally, we use the method in Section 2.4 for the controller design of the studied RMS.

3.1 Modeling of Reconfigurable Machine Tools

DES as introduced in Section 1.1 are suitable for modeling elementary operations and the interaction behavior of technical machine processes such as reconfigurable manufacturing tools (*RMTs*). We now model the RMT in Figure 3.1. It is part of a laboratory model that was acquired in the scope of a TÜBİTAK project at Çankaya University [14].

Looking at the figure, the main components of our example RMT are:

- a conveyor belt for transporting products
- a machine head that can move up and down and turn to different positions
- three machine tools that can perform different operations.

We next develop finite automata models of the RMT components, that are then



Figure 3.1: RMT with three different modes of operation.

assembled to form a model of the overall RMT.

3.1.1 Conveyor Belt

The conveyor belt is used to transport products to and from the RMT. Fig. 3.2 shows a picture of such conveyor belt.



Figure 3.2: Picture of a conveyor belt.

The conveyor belt is actuated by a motor that is supposed to move a product from left to right or stop. The related events are `sf-rmt_SW` (start the motor)¹ and `rmt_boff` (stop). Since these events can be directly influenced by an operator (it is possible to switch on/off the motor on demand), these events are considered

¹ The name of the event is given under the assumption that there is a component with the name SF next to the RMT (see Section 3.3)

as controllable events. In addition, there is a sensor in the middle of the conveyor belt to detect the presence/absence of a product. It is assumed that the conveyor belt is stopped whenever the product reaches the sensor or leaves the conveyor belt. The automaton model G_{cb} is depicted in Fig. 3.3.



Figure 3.3: Conveyor belt model G_{cb} .

3.1.2 Machine Head

The machine head has two main functionalities. On the one hand, it can move vertically and stop at an upper and a lower position. Both positions are detected by sensors. On the other hand, it can turn to three different positions, which characterize the three different configurations of the RMT. Again, there are sensors that indicate the different positions.

Considering the vertical motion, the machine head is initially in the upper position. If the event rmt_pm+ happens, a motor is switched on, and the machine head moves toward the lower position. As soon, as the lower sensor is reached, the event rmt_poff happens and the motor is stopped. After that, the event rmt_pm- can happen to move back to the upper position until the upper sensor is reached. Then, again rmt_poff stops the motion of the machine head. The automata model G_{mh} of the machine head is shown in Figure 3.4.



Figure 3.4: Machine head model G_{mh} .

Considering the rotation of the machine head, each position of the machine head

corresponds to one of its operations. "drilling", "polishing" and "milling". We assume that the machine head is initially in the drilling position (D). From there, the machine head can turn to the polishing position (`rmt_D-P_SW`) or to the milling position (`rmt_D-M_SW`) until it stops at the respective position with `rmt_toff`. Similar changes of position are possible from the polishing and milling position. The automaton model G_t of the turning action of the machine head is shown in Figure 3.5.

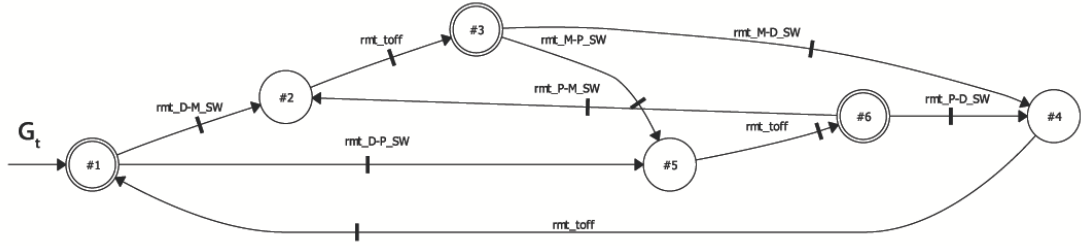


Figure 3.5: Model of the turning operation of the machine head.

3.1.3 Machine Tools

RMTs are designed to perform different operations in different configurations. The RMT in Figure 3.1 is designed to have three different configurations that are represented by the operations of "drilling!", "polishing" and "milling". We model the behavior of each operation by a finite state automaton. In principle, each of the operations starts with an initiating event. For example, the drilling operation starts with `rmt_don`. If the operation is finished, it acknowledges its completion. For example, the event `rmt_dack` happens after drilling is finished. After that, the drill is switched off with `rmt_dtoff`. The corresponding automaton G_d is shown in Figure 3.6. The operation of polishing and milling are analogous.

3.1.4 Overall Model

Now that the models of each components of RMTs have been derived, we get the overall model of the RMT from the parallel composition of the component models:

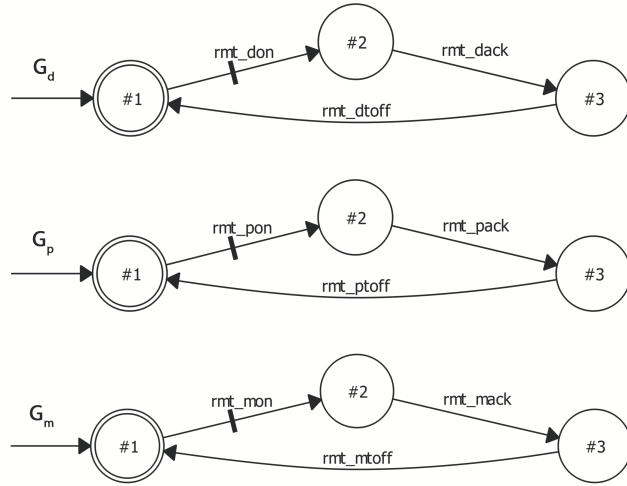


Figure 3.6: Models of the RMT operation.

$$G_{RMT} = G_{op} || G_{mh} || G_t || G_{cb}.$$

The resulting automation G_{RMT} is not shown since it has 768 states.

3.1.5 Safety Specifications for the RMT

The next step in the control of the RMT is the realization of so-called safety specifications. These specifications restrict the behavior of the RMT such that certain unwanted event sequences will never happen. We now explain the required restrictions.

- Spec1: If the conveyor belt is moving, the machine head must not move down and the machine must not operate. Otherwise products can be damaged. The automaton C_{Spec1} for Spec1 is shown in Fig. 3.7.
- Spec2:
 - If the machine head is in the upper position and does not move, then the conveyor belt is allowed to move but the machine must not operate. Operation of the machine is only possible if the machine head is in the

lower position. This part of the specification is represented by state 1 of C_{Spec2} in Figure 3.7.

- If machine head is moving, then the conveyor belt must not move and machine head must not operate. This part of the specification is represented by state 2.
 - If the machine head is in the lower position and not moving, then the machine is allowed to operate (state 3), and afterwards, the machine head is moving to upper position and stops (state 1).
- Spec3: The machine head is only allowed to move down if a product is present on the conveyor belt. The automaton C_{Spec3} is shown in Fig. 3.7.

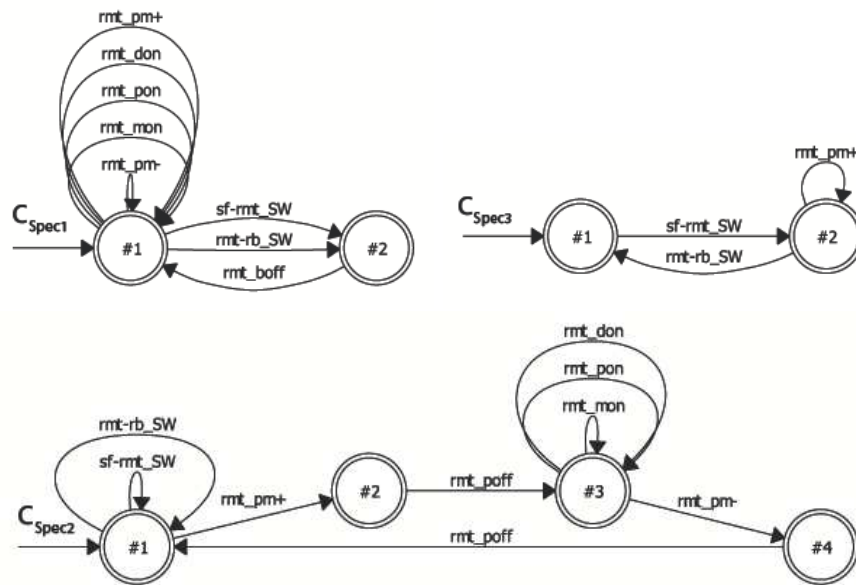


Figure 3.7: Specification automata: C_{Spec1} , C_{Spec2} and C_{Spec3} .

- Spec4: If some machine tool is operating, then the machine head must not move. Otherwise the product will be damaged. Figure 3.8 shows the automaton C_{Spec4} .
- Spec5: A machine tool is only allowed to operate if the machine head is in the lower position. Figure 3.8 shows the automaton C_{Spec5} .

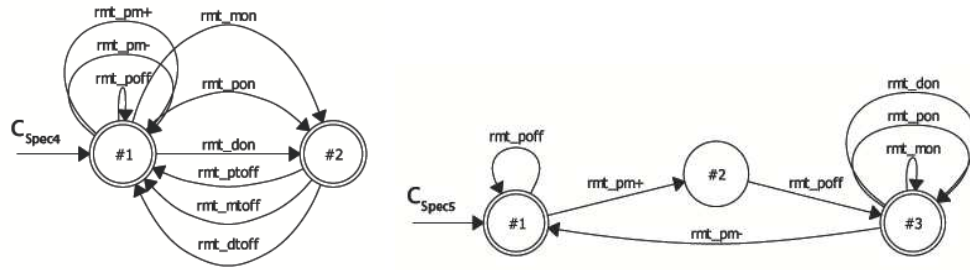


Figure 3.8: Specification automata C_{Spec4} and C_{Spec5} .

- Spec6: Each machine tool is only allowed to operate if the machine head is turned to the correct position. This means
 - the drilling operation is only allowed if the machine head is in drilling position
 - the polishing operation is only allowed if the machine head is in the polishing position
 - the milling operation is only allowed if the machine head is in the milling position

The automaton C_{Spec6} for Spec6 is shown in Fig. 3.9.

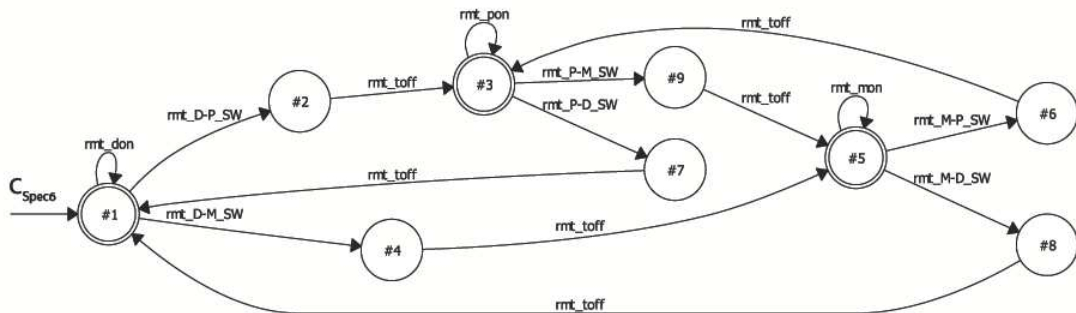


Figure 3.9: Specification automaton C_{Spec6} .

- Spec7: If the machine head is moving, then the machine head must not turn. The automaton C_{Spec7} for Spec7 is shown in Fig. 3.10.
- Spec8: If the machine head is turning, then the machine head must not move. The automaton C_{Spec8} for Spec8. is shown in Fig. 3.10.

- Spec9: If the machine head is turning, then the conveyor belt of the RMT must not move. The automaton C_{Spec9} for Spec9 is shown in Fig. 3.10.
- Spec10: If the conveyor belt of the RMTs is moving, then the machine head must not turn. The automaton C_{Spec10} for Spec10 is shown in Fig. 3.10.

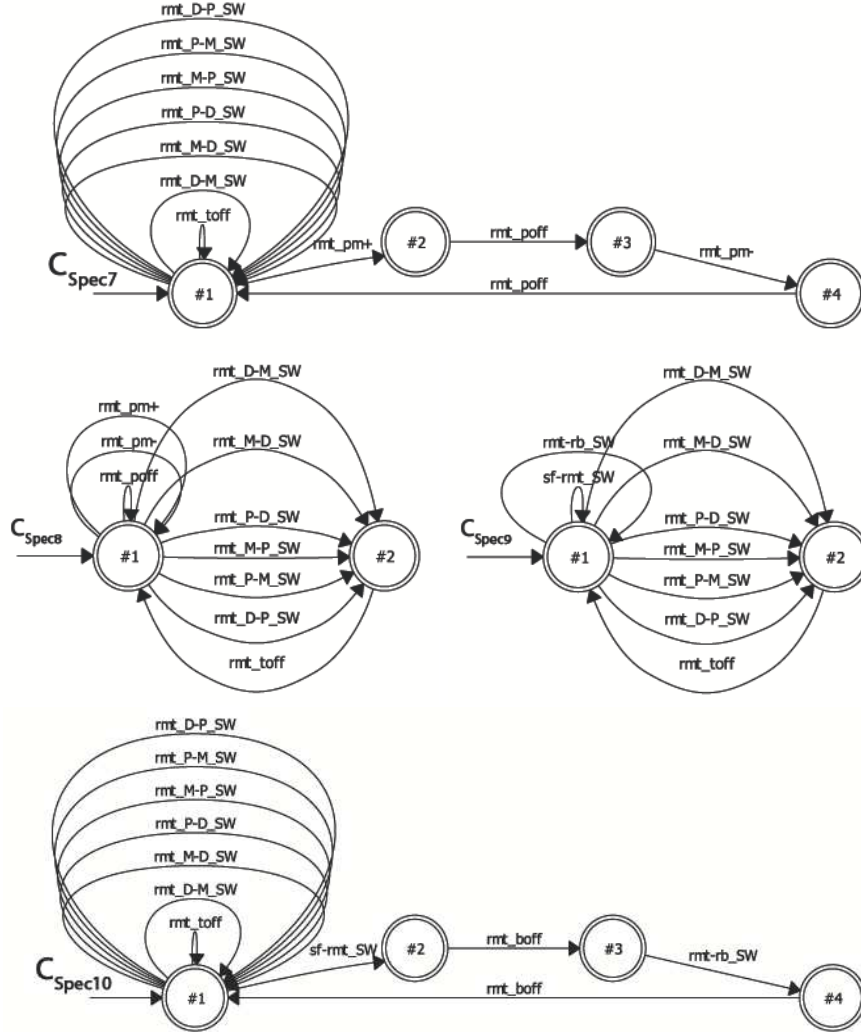


Figure 3.10: Specification automata C_{Spec7} , C_{Spec8} , C_{Spec9} and C_{Spec10} .

Together, we get the overall specification as the parallel composition of all specification automata:

$$C_{Spec} = C_{Spec1} || C_{Spec2} || C_{Spec3} || C_{Spec4} || C_{Spec5} || C_{Spec6} || C_{Spec7} || C_{Spec8} || C_{Spec9} || C_{Spec10}.$$

Based on the classical supervisory control theory [12], we can now compute a supervisor S_{RMT} for the overall RMT by applying the function SupConNB (see Section 1.2): $SupC(C_{Spec}, G_{RMT}, \Sigma_u)$. By definition, the supervisor S_{RMT} is non-blocking and fulfills the specification C_{Spec} . Considering that the plant model G_{RMT} already has 768 states and the specification C_{Spec} has 30 states, it turns out that the supervisor S_{RMT} has 30 states. Hence the automaton S_{RMT} is too large to be shown. However, according to the hierarchical supervisory control theory as presented in [15, 17], it is possible to compute a hierarchical abstraction by applying the natural projection to S_{RMT} . The abstraction has a smaller number of states and can be used for the next steps in the reconfiguration controller design. It is shown in Figure 3.11.

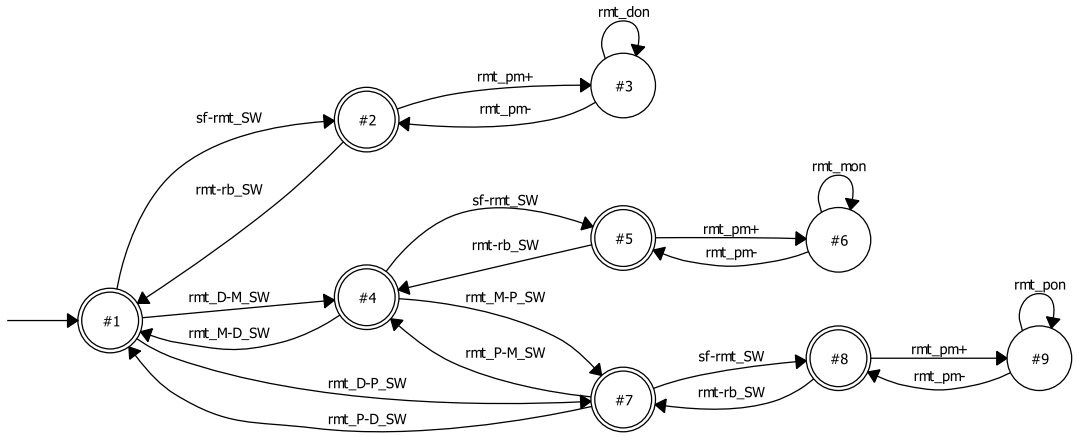


Figure 3.11: High-level abstraction S_{RMT}^{high} of the RMT.

3.2 Supervisory Control of RMTs

We now apply the controller design methods for RMTs as described in Section 2.3 to the RMT example considered in the previous section. That is, we use the abstracted closed-loop system shown in Figure 3.11 as the plant G for the reconfiguration controller computation. From the previous discussion, we already know that the RMT can operate in three different configurations – drilling (D), polishing (P) and milling (M). Hence, our set of *configurations* is $\mathcal{C} = \{D, P, M\}$.

We first determine the *configuration supervisor* S^ρ for each configuration $\rho \in$

\mathcal{C} . Considering drilling, it is desired to start from state 1 of G . Whenever a product enters the system (sf-rmt_SW), the drilling operation should be performed (rmt_don) and the product should exit the system (rmt-rb_SW). This operation is specified in the automaton C^D in Figure 3.12. Polishing should start in state 7 of G , where the machine head is turned to the polishing configuration. Then, polishing (rmt_pon) should be performed after arrival (sf-rmt_SW) and before departure (rmt-rb_SW) of a produce. The same idea is applied for the milling operation (rmt_mon). It starts in state 4 of G . All specification automata C^D (drilling), C^P (polishing) and C^M (milling) are shown in Figure 3.12.

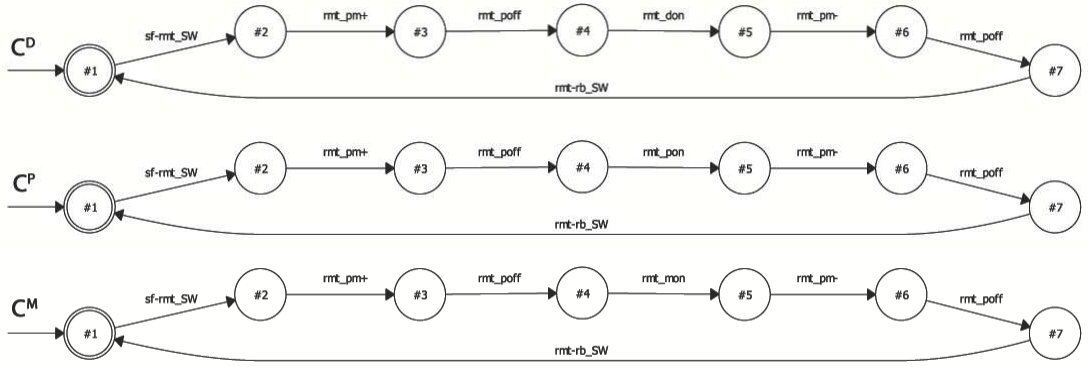


Figure 3.12: Specification per configurations of RMT.

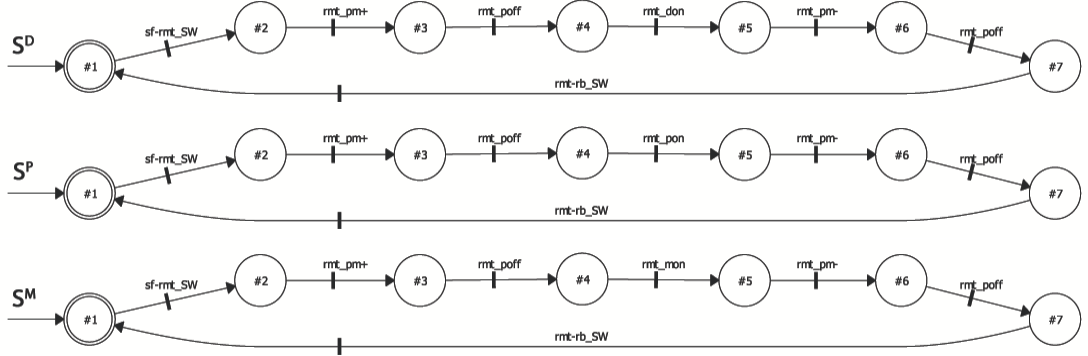


Figure 3.13: Supervisor per configurations of RMT.

As described in Section 2.3, the supervisor S^ρ for each $\rho \in \mathcal{C}$ is computed as $L_m(S^\rho) = \text{Sup}C(K^\rho, L(G^\rho), \Sigma_u)$. Here, the specification $K^\rho = L_m(C^\rho)$ and the plant G^ρ starting from the initial state of configuration ρ are used. For $\rho = D$, we use G^D (initial state 1) and $K^D = L_m(C^D)$, for $\rho = P$, we use G^P (initial

state 7) with $K^P = L_m(C^P)$, and for $\rho = M$, we use G^M (initial state 4) with $K^M = L_m(C^M)$. The resulting supervisors for the three configurations of our RMT are depicted in Fig. 3.13.

Until now, we introduced the elementary operations for each component of the RMT with its controller supervisor for separate configuration but not for re-configuration between the different configurations. Based on these previous notions we use the algorithm in Section 2.3 to construct supervisors for re-configuration (change the configuration from one configuration to another configuration). We introduce new events as a *reconfiguration start event* ρ_{st} , such that $\Sigma_{st} = \bigcup_{\rho \in C} \{\rho_{st}\}$, where $\Sigma_{st} = \{D_st, P_st, M_st\}$, also we enter the *reconfiguration finish event* ρ_{fin} such that $\Sigma_{fin} = \bigcup_{\rho \in C} \{\rho_{fin}\}$, where $\Sigma_{fin} = \{D_fin, P_fin, M_fin\}$.

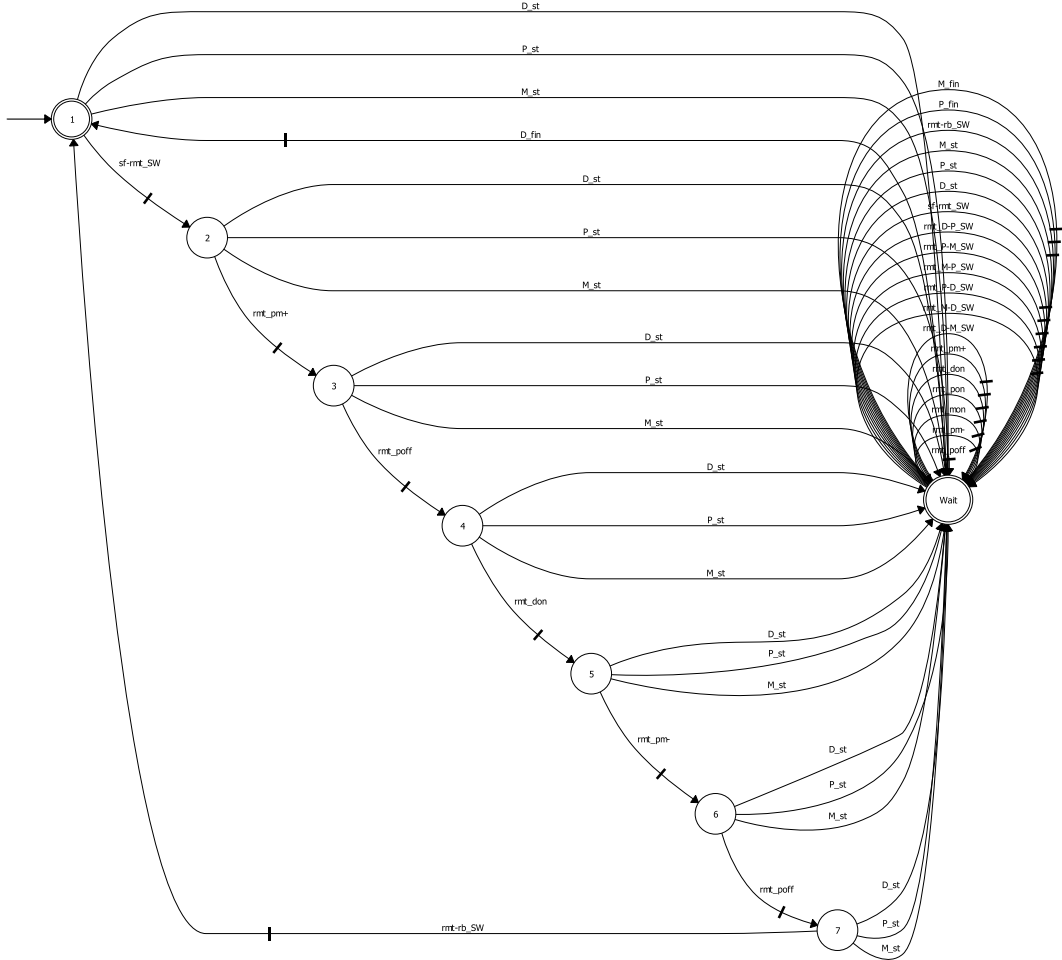


Figure 3.14: Reconfiguration supervisor \overline{S}^D .

Now, we can directly apply the algorithm in Section 2.3 to compute the reconfiguration supervisor of the RMT $S^{\text{rec}} = \overline{S} \parallel \overline{S}^D \parallel \overline{S}^P \parallel \overline{S}^M$. The supervisors per configuration \overline{S}^ρ for $\rho \in \mathcal{C}$ are shown in Fig. 3.14, 3.15 and 3.16. They are constructed such that, if a configuration $\rho \in \mathcal{C}$ (for $\rho = D$) is active, the component \overline{S}^D follows the configuration supervisor S^D as long as it is active and switches to the waiting state (Wait) as soon as the configuration becomes inactive and a different configuration becomes active. During reconfiguration, all other supervisors \overline{S}^P and \overline{S}^M for $P, M \in \mathcal{C}$ are in their waiting state. The behavior of \overline{S}^P and \overline{S}^M is analogous.

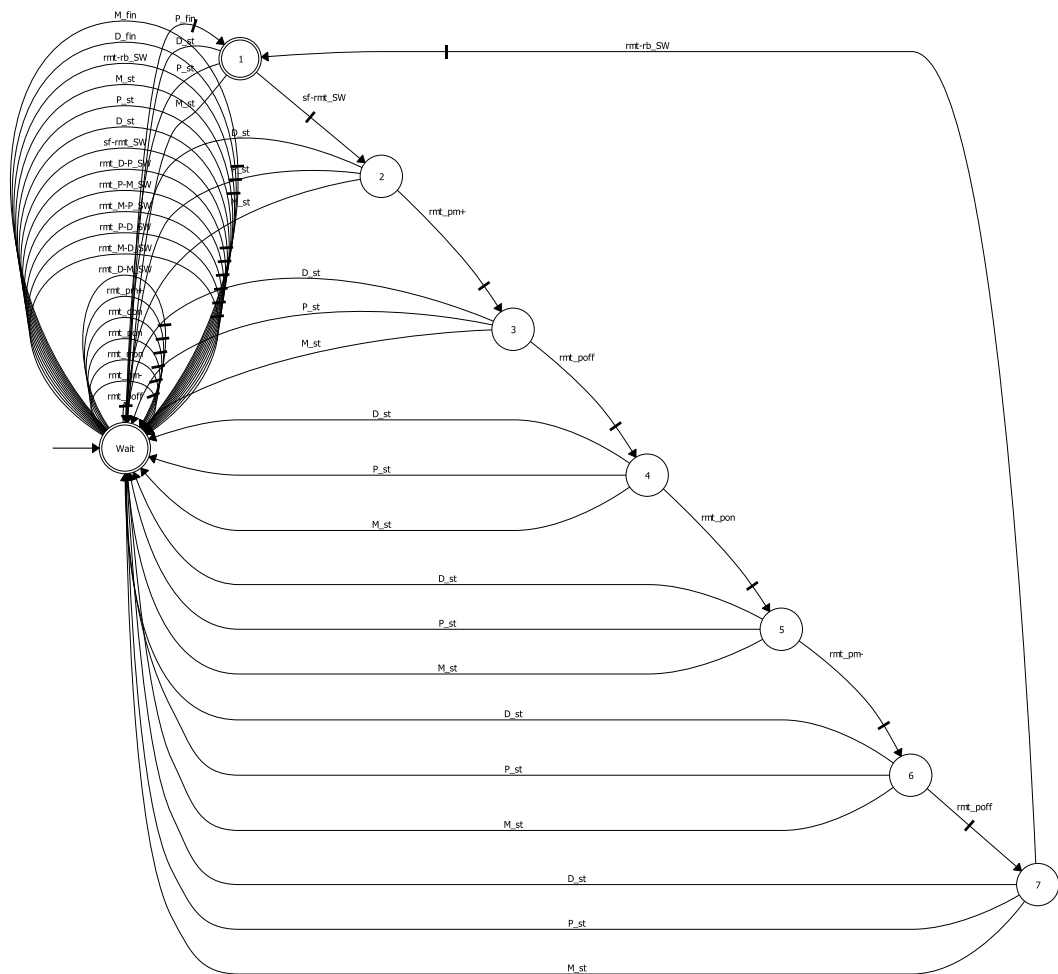


Figure 3.15: Reconfiguration supervisor \overline{S}^P .

In addition, there is one coordinator supervisor \overline{S} . The task of the supervisor \overline{S} is to follow the current plant state if some configuration is active. In case

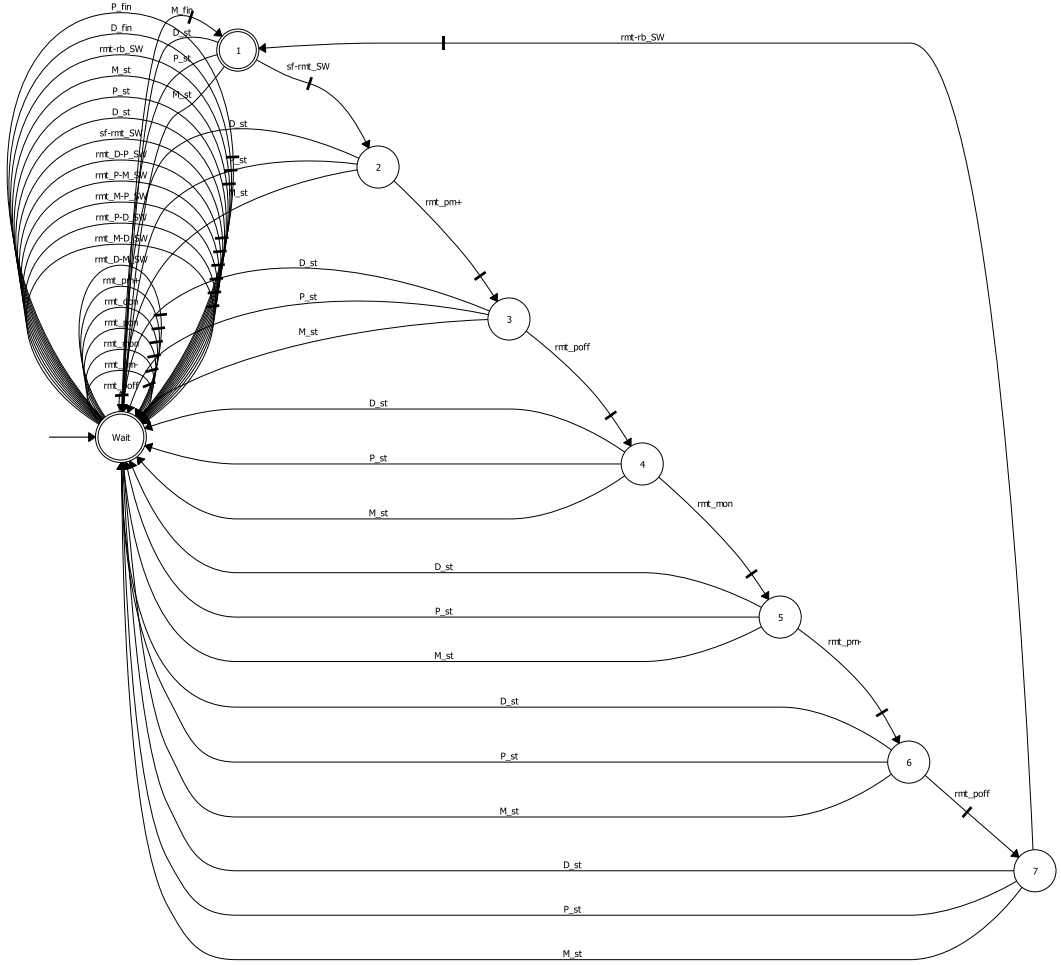


Figure 3.16: Reconfiguration supervisor \overline{S}^M .

of reconfiguration to $\rho \in \mathcal{C}$, \overline{S} moves the plant state to the initial state of the new configuration as fast as possible. Hence, \overline{S} allows to switch between different configuration. Since \overline{S} has 60 states, it is too large to be shown in this thesis. As is mentioned in Section 3.1, it is possible to compute a hierarchical abstraction by applying the natural projection to \overline{S} . The abstraction of \overline{S} is shown in Figure 3.17.

As we said, the overall reconfiguration supervisor is given by $S^{\text{rec}} = \overline{S} \parallel \overline{S}^D \parallel \overline{S}^P \parallel \overline{S}^M$. Again S^{rec} is too large to be shown because it has 75 states. We compute the abstraction of S^{rec} on the event set $\{\text{M_st}, \text{D_st}, \text{P_st}, \text{sf-rmt_SW}, \text{rmt-rb_SW}\}$ to be used as new plant model S^{hi} of the RMT. It is shown in Figure 3.18.

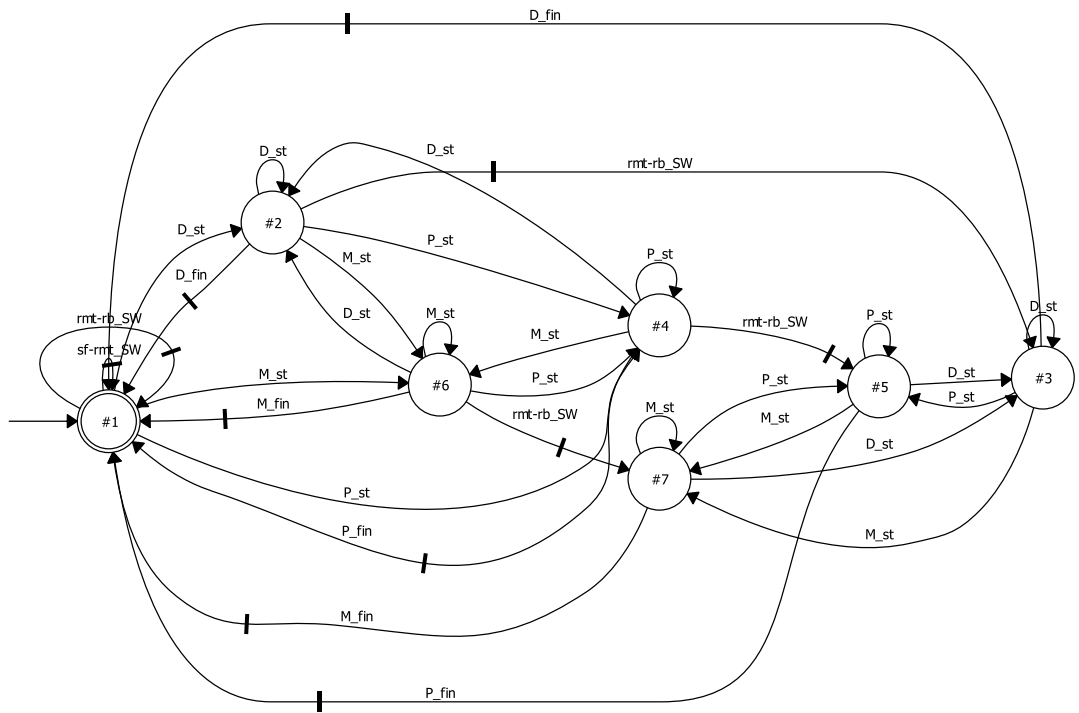


Figure 3.17: Abstraction of \bar{S} of RMT.

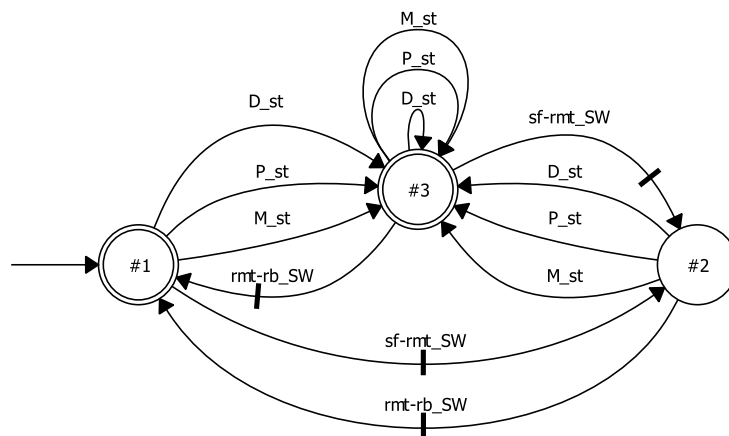


Figure 3.18: Reconfiguration supervisor S^{rec} , high plant of RMT $G_{RMT\text{plant}}$.

3.3 Modeling of Reconfigurable Manufacturing Systems

Our example RMS consists of different manufacturing components:

- one stack feeder (SF)
- one reconfigurable machine tool (RMT)
- one rotary table (RT)
- three exit slides (XS).

A schematic of the RMS is shown in the following figure.

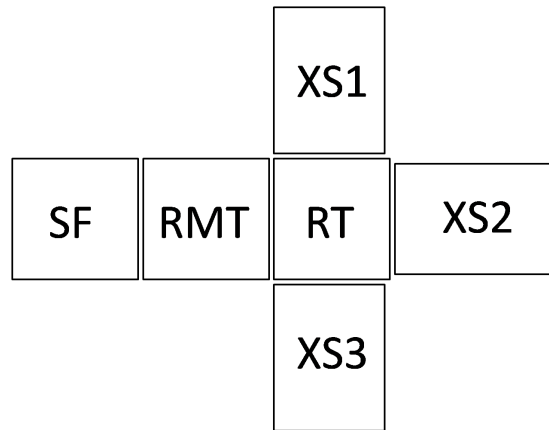


Figure 3.19: Diagram of the example RMS.

The description of the desired operation of the RMS example is as follows. The process begins from the stack feeder, where products enter the RMS. From there, products are moved to the RMT and are processed according to the current RMT configuration. After leaving the RMT, products are transported to different exit slides (XS) by the rotary table (RT). The exit slides represent storage units, from where the ready product can be taken.

It is our goal to apply the controller design method, that is developed in Section 2.4, to this RMS example. The first task is hence, to obtain an automata model of the whole RMS. Considering that a model of the RMT was already found

in the previous section, it remains to model the remaining components: SF, RT and XS.

3.3.1 Stack Feeder

The stack feeder consists of a tower that can hold unprocessed products, and a belt with a vertical bar for pushing the workpieces to the neighboring component (in our case, the neighbor is the RMT). The position of the vertical bar is detected by a magnetic sensor. If the bar is next to the sensor, we say it is in its *rest position*. In addition, there is a photoelectric barrier, that sees if a product is currently present in the SF. The motion of the SF's belt is controlled from a motor that can be switched on and off. Fig. 3.20 shows a picture of the stack feeder.

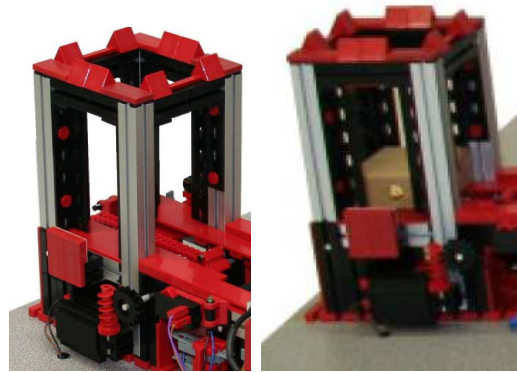


Figure 3.20: Picture of the stack feeder.

The operation of the SF is as follows. The sensor in the middle of the belt detects when a workpiece (product) arrives on the SF, which is shown by the event `sf_wpar`. The operation of the SF can then be started (`sf-rmt_SW`) and the belt is switched on with the event `sf_fdon`. The bar on the belt leaves the rest position (`sf_nr`) and returns to the rest position after one round (`sf_r`). While moving, the bar pushes the workpiece away (`sf_wplv`). Then, the motor will stop again (`sf_fdoff`) when the bar is at the rest position.

The automaton model of the stack feeder is shown in Figure 3.21. This model exactly represents the SF operation as described before. In the overall RMS, we want to use small models of the different components. Because of this reason,

we again use the idea of abstraction as developed in [16] and obtain a high-level automaton model of the stack feeder as shown in Fig. 3.22. As can be seen in the figure, it is only required to keep the event `sf-rmt_SW` in the model.

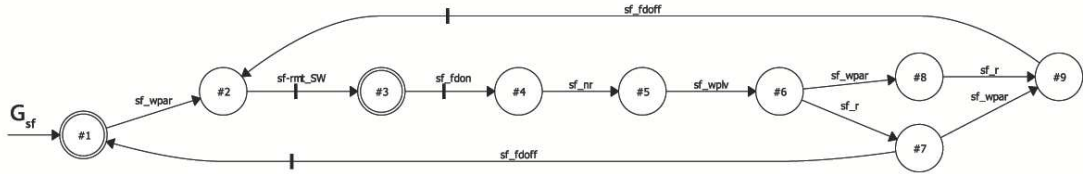


Figure 3.21: An automaton model G_{sf} for the stack feeder.

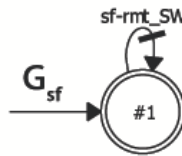


Figure 3.22: High level stack feeder model G_{sf} .

3.3.2 Rotary Table

The rotary table consists of a conveyor belt and a table, that can rotate in clockwise and counter-clockwise direction. As a result, the rotary table can send and receive workpieces to and from four direction – north, south, east and west. Fig. 3.23 shows the rotary table.

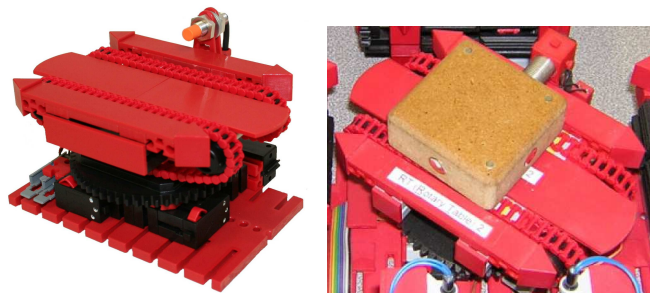


Figure 3.23: Picture of the rotary table.

In order to keep the discussion simple, we only present an already abstracted model of the rotary table. It describes moving workpieces to the rotary table,

removing workpieces from the rotary table as well as turning the rotary table. Considering the usage of the rotary table in our RMS in Figure 3.19, we know that the rotary table is located among the RMT and three exit slides. Workpieces can enter from the RMT (rmt-rb_SW) but not from the exit slides. In addition, workpieces can leave the RB toward the RMT (rb-rmt_SW), and each of the exit slides (rb-xs1_SW , rb-xs2_SW , rb-xs3_SW). The orientation of the RB is changed by turning the table. The related events are rb_tcw (turn clockwise) and rb_tccw (turn counter-clockwise).

The automaton model G_{rb} of the rotary table is shown in Fig. 3.24. It can be seen that workpieces, which arrive at the RB can be placed in one of the three storage areas (events rb-xs1_SW , rb-xs2_SW and rb-xs3_SW) by changing the direction of rotary table with the events rb_rcw and rb_rccw .

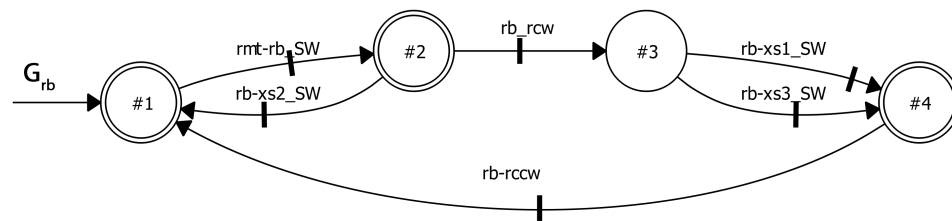


Figure 3.24: Rotary table model G_{rb} .

3.3.3 Exit Slides

Through the exit slides the workpieces can leave the manufacturing system, there is also a sensor to indicate the present of workpieces. The exit slides represent as deposit area for processed workpieces (can host up until four workpieces). Fig. 3.25 represents the exit slide.

The automaton models of the three exit slides are shown in Fig. 3.26.

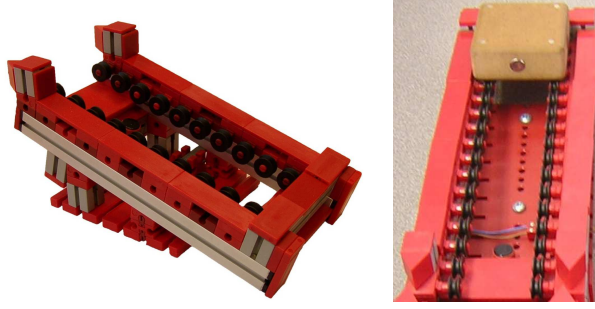


Figure 3.25: Picture of the exit slide.

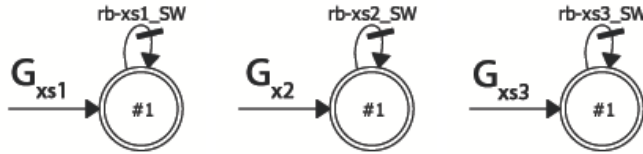


Figure 3.26: Exit slide models G_{xs1} , G_{xs2} and G_{xs3} .

3.4 Supervisory Control of RMS

We now want to apply the controller design method developed in Section 2.4 to the RMS example. The plant DES is represented by the parallel composition of the component automata. We compute the overall RMS model G_{RMS} as:

$$G_{RMS} = G_{RMTplant} \parallel G_{sf} \parallel G_{rb} \parallel G_{xs1} \parallel G_{xs2} \parallel G_{xs3}$$

The automaton G_{RMS} is shown in Fig. 3.27.

In our example, we first want to realize two configurations of the RMS. In the first configuration, workpieces enter the RMS from the SF, are drilled by the RMT and then move to exit slide XS1. In the second configuration, workpieces should be polished and leave the system on exit slide XS2. The specifications C_{conf1} and C_{conf2} in Figure 3.28 describe the desired operation.

Using these specifications, it is now possible to compute the supervisors S_{conf1} and S_{conf2} for the different configurations. They are computed such that $L_m(S_{conf1}) = SupC(L_m(C_{conf1}), G_{RMS}, \Sigma_u)$ and $L_m(S_{conf2}) = SupC(L_m(C_{conf2}), G_{RMS}, \Sigma_u)$. The operation of the different configurations is depicted in Figure 3.29.

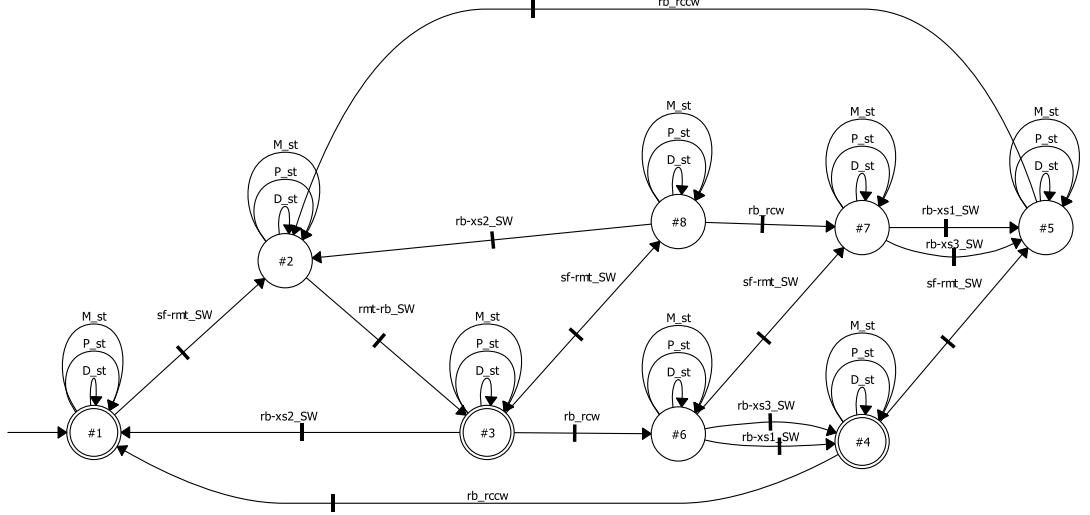


Figure 3.27: RMS system plant G_{RMS} .

It has to be noted that the supervisors in Figure 3.29 assume that the RMT is already in the correct configuration. Moreover, each supervisor is designed for separate configurations without taking care about reconfigurations. In the next step, we compute a reconfiguration supervisor for the RMS. In analogy to the method for RMTs in Section 2.3, we use two supervisors \bar{S}_{conf1} and \bar{S}_{conf2} for the different configurations. They either follow the respective configuration or remain in an inactive state "Wait" as shown in Figure 3.30 and Figure 3.31. Considering the notation, we use the new events for starting and finishing the reconfiguration. The *reconfiguration start events* are $\Sigma_{st} = \{\text{conf1_st}, \text{conf2_st}\}$ and the *reconfiguration finish events* are $\Sigma_{fin} = \{\text{conf1_fin}, \text{conf2_fin}\}$.

It remains to find the supervisor \bar{S} that changes between configurations. Here, the new technique as developed in Section 2.4 is used. In principle, the task of \bar{S} is to move the state of G_{RMS} to the initial state 1 such that a new configuration can start its operation. However, considering that a new configuration also needs a reconfiguration of the RMT, it must be guaranteed that the RMT is in the correct configuration when the new configuration of the RMS starts. This means for example if a reconfiguration to conf2 is required, \bar{S} has to move G_{RMS} to state 1 and at the same make sure that the event P_st (reconfiguration of the RMT to polishing operation) must happen. That is, instead of computing optimal

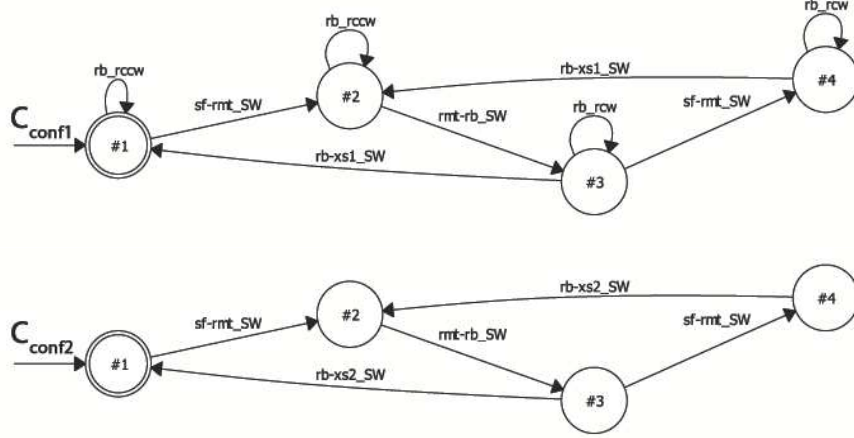


Figure 3.28: Configurations specification of RMS.

supervisors for state attraction as in Section 2.3, we now use supervisors for state attraction under language specification for \bar{S} . The language specifications for conf1 and conf2 are shown in Figure 3.32.

The coordinating part \bar{S} of the reconfiguration supervisor is constructed analogous to Section 2.3. Hence, the automaton of the supervisor \bar{S} is large to be shown (\bar{S} has 104 states), So, a part of \bar{S} is shown in Figure 3.33, we choose some states from the RMS plant in Fig. 3.27 we relate this, to supervisors for state attraction under language specification. In case of reconfiguration, if the plant is in state 2 and we reconfigure to configuration 1, \bar{S} moves the plant state to the initial state as fast as possible and should fulfill the *reconfiguration specification* language represented by reconfiguration start event D_st while moving to state 1. The reconfiguration to configuration 2 is analogous to previous reconfiguration, whereby the event P_st must happen on the path to state 1.

Together, the reconfiguration supervisor of the RMS is the parallel composition of the supervisors for each configuration and the coordinating supervisor. We obtain

$$S^{\text{rec}} = \bar{S}_{\text{conf1}} || \bar{S}_{\text{conf2}} || \bar{S}.$$

S^{rec} with two configurations has 110 states.

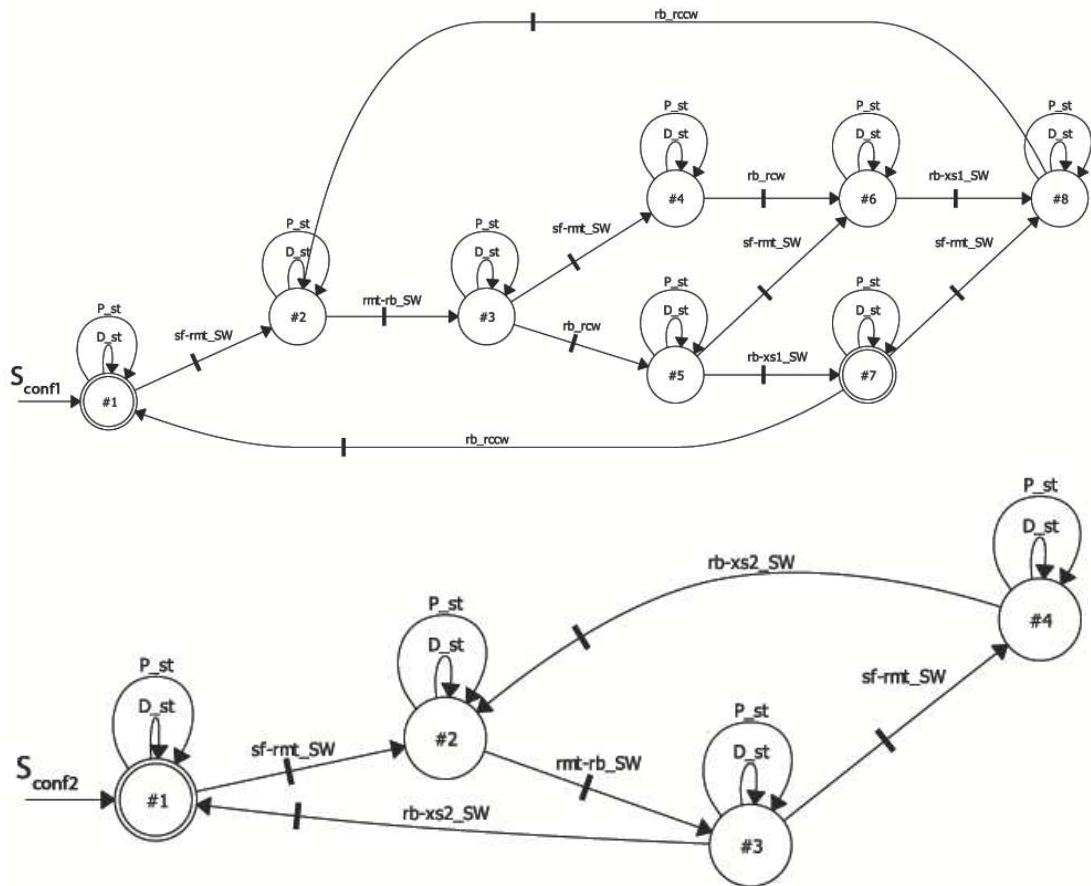


Figure 3.29: Supervisor per configuration specification of RMS.

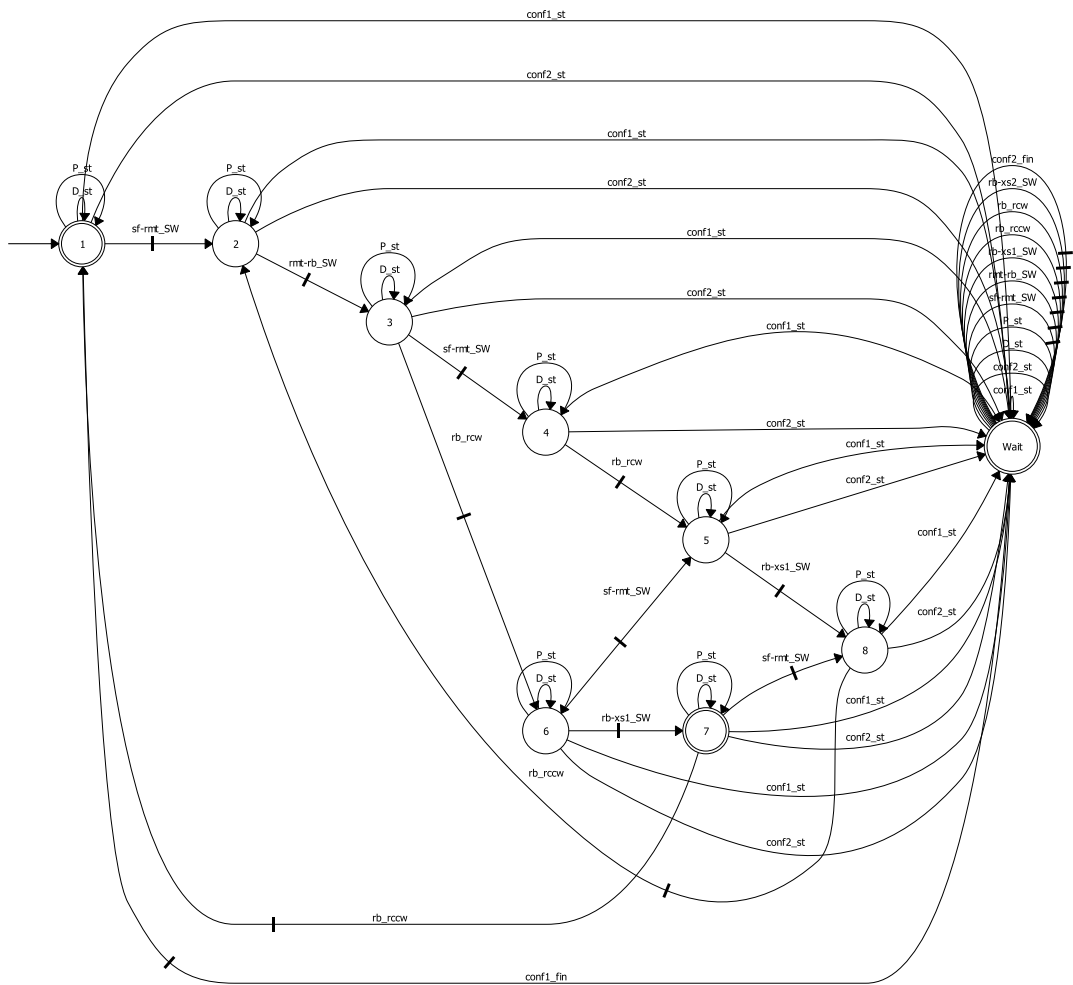


Figure 3.30: Components \overline{S}_{conf1} of the reconfiguration supervisor.

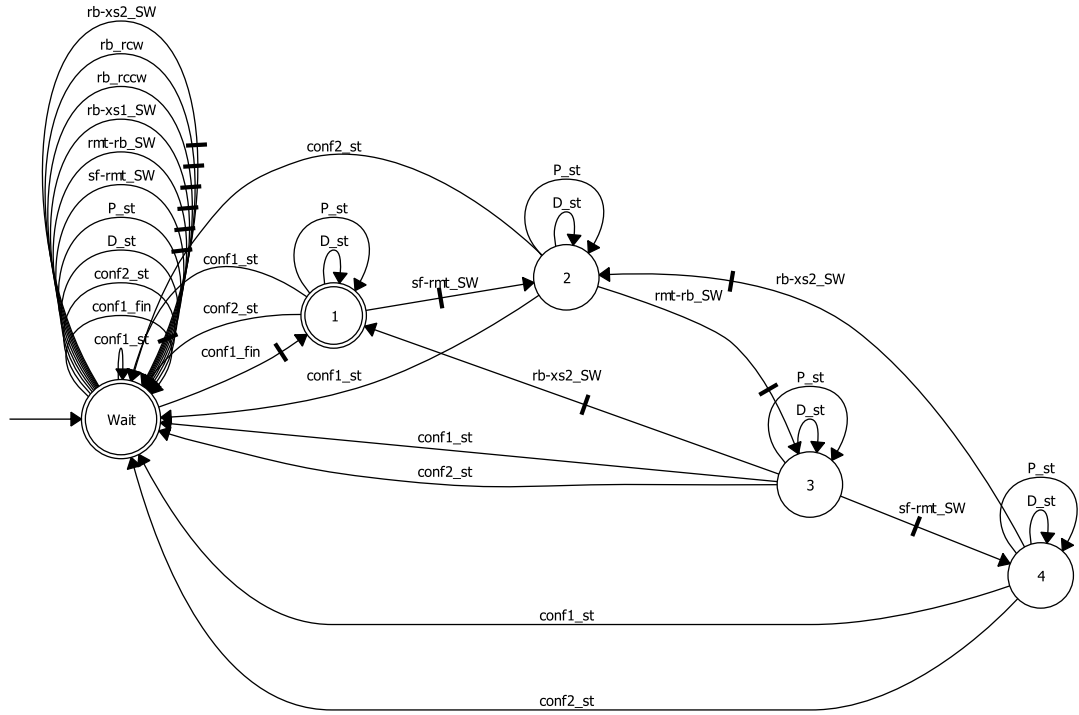


Figure 3.31: Components \bar{S}_{conf2} of the reconfiguration supervisor.

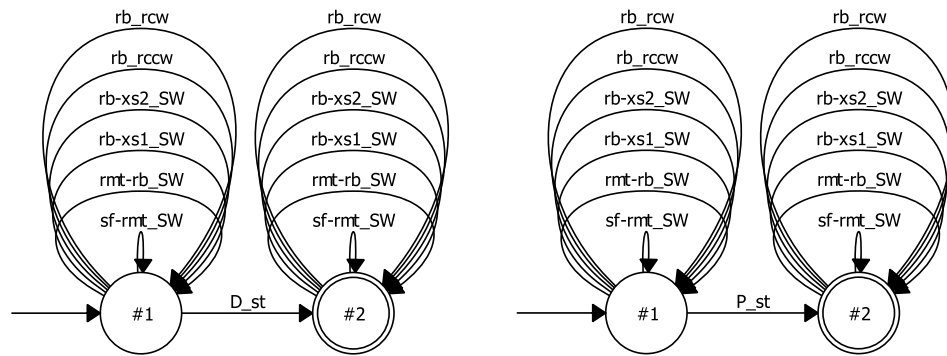


Figure 3.32: Language specifications for reconfiguration.

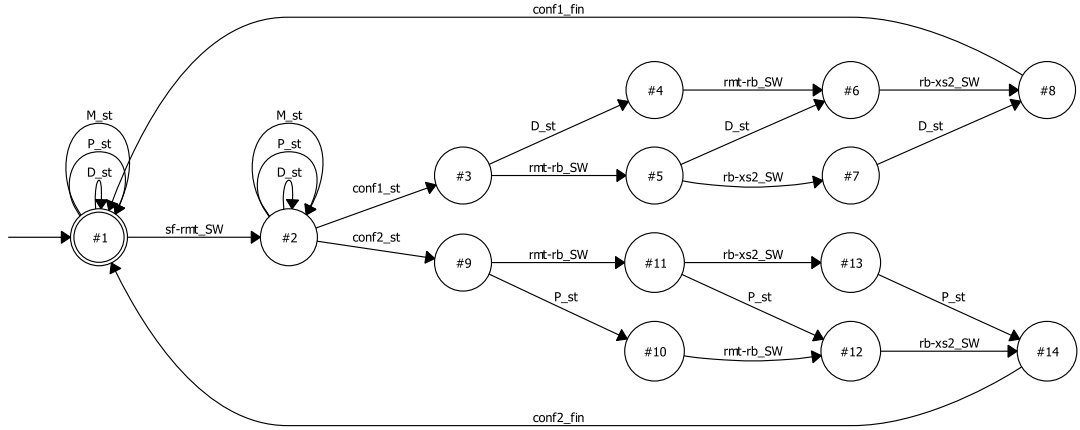


Figure 3.33: Supervisors for state attraction under language specification.

3.5 Re-usability of Controllers

We now discuss how the previously designed controllers can be re-used if a new configuration of the RMS is added. We consider adding a new configuration ρ_{new} to the RMT during operation. In that case, it is only required to compute a supervisor $\overline{S}_{\rho_{\text{new}}}$ and to re-compute the supervisor \overline{S} including the new configuration. The existing supervisors \overline{S}_{ρ} for $\rho \in \mathcal{C}$ do not have to be changed, since their transitions do not depend on the individual events of the new configuration. They only depend on their own reconfiguration events and on the overall set of reconfiguration events. For example, considering conf1 in the example in Section 3.4, the supervisor S_{conf1} in Figure 3.30 only needs to know that a configuration start event different from `conf1_st` happened in order to transition to the state "Wait". Likewise, it only needs to see its own reconfiguration finish event `conf1_fin` in restart operation of conf1. Hence, it is possible to add a new configuration during operation. It is only required to load the new (inactive) supervisor $\overline{S}_{\rho_{\text{new}}}$ and the supervisor \overline{S} (that is in a well-defined plant state) on the controller device.

We next relate the previous discussion to our example in Section 3.4. We want to add the new configuration conf3. In this configuration, workpieces should be milled and leave the system on exit slide XS3. The configuration specification C_{conf3} and the corresponding configuration supervisors S_{conf3} in Figure 3.34 describe the desired operation for this configuration.

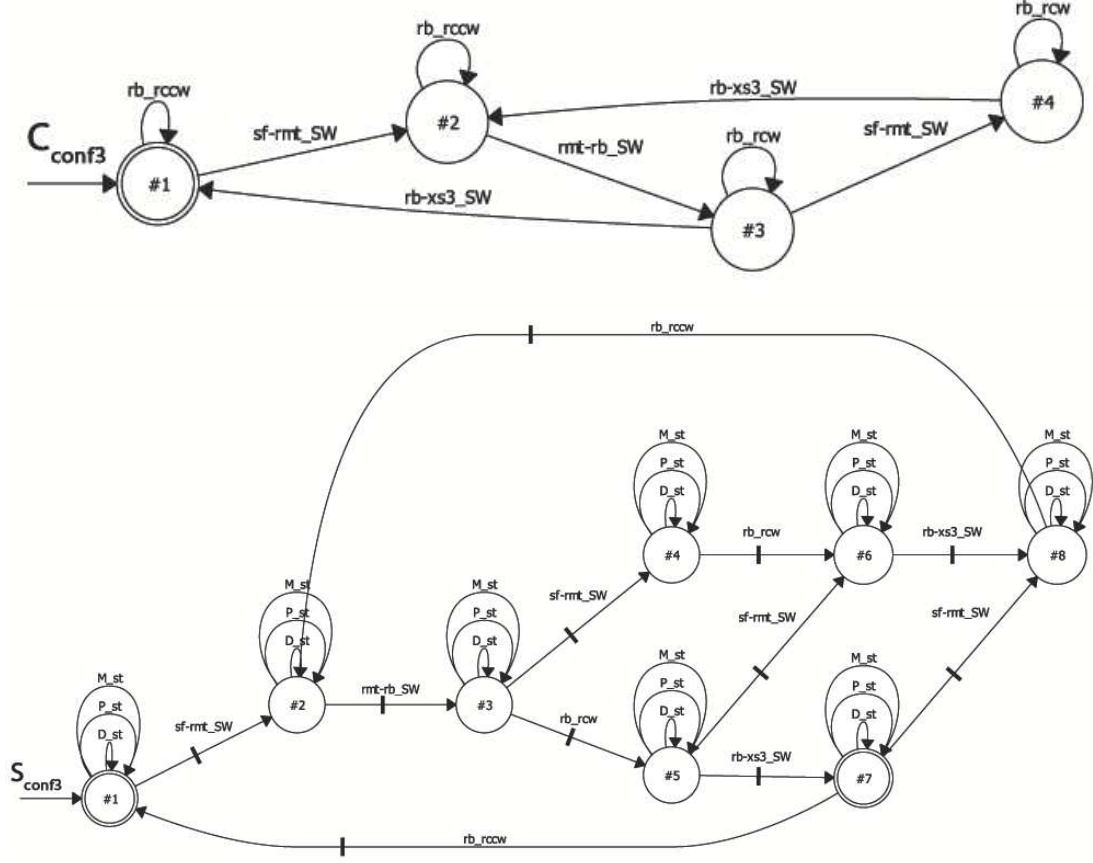


Figure 3.34: Specification and supervisor for milling configuration of RMS.

As mentioned above, it is now only required to compute a reconfiguration supervisor \bar{S}_{conf3} for conf3. Again, it either follows the corresponding configuration or remains in an inactive state "Wait" as shown in Figure 3.35.

The automaton of the supervisor \bar{S} is too large to be shown; already it has 152 states. We take part of the automaton \bar{S} as shown in Figure 3.36. Hence, the basic operation of \bar{S} is analogous to \bar{S} with two configurations as illustrated in Figure 3.33. Figure 3.36 shows the subautomaton of \bar{S} with three configurations.

Finally, the overall reconfiguration supervisor of the RMS with three configurations is the parallel composition of the supervisors for the separate three configurations and the coordinating supervisor. We obtain

$$S^{\text{rec}} = \bar{S}_{conf1} || \bar{S}_{conf2} || \bar{S}_{conf3} || \bar{S}.$$

Hence, S^{rec} with three configurations has 166 states.

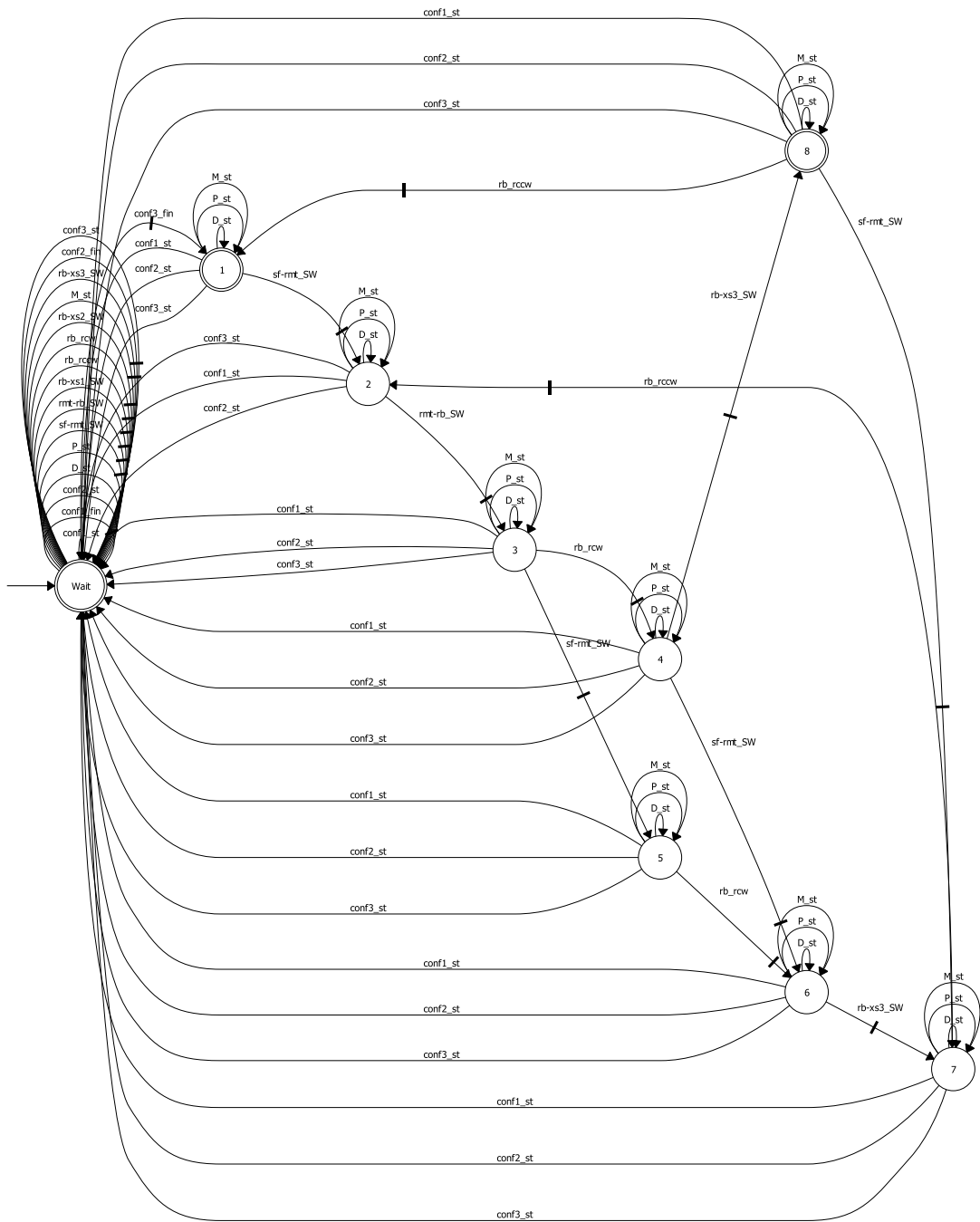


Figure 3.35: Components \overline{S}_{conf3} of the reconfiguration supervisor.

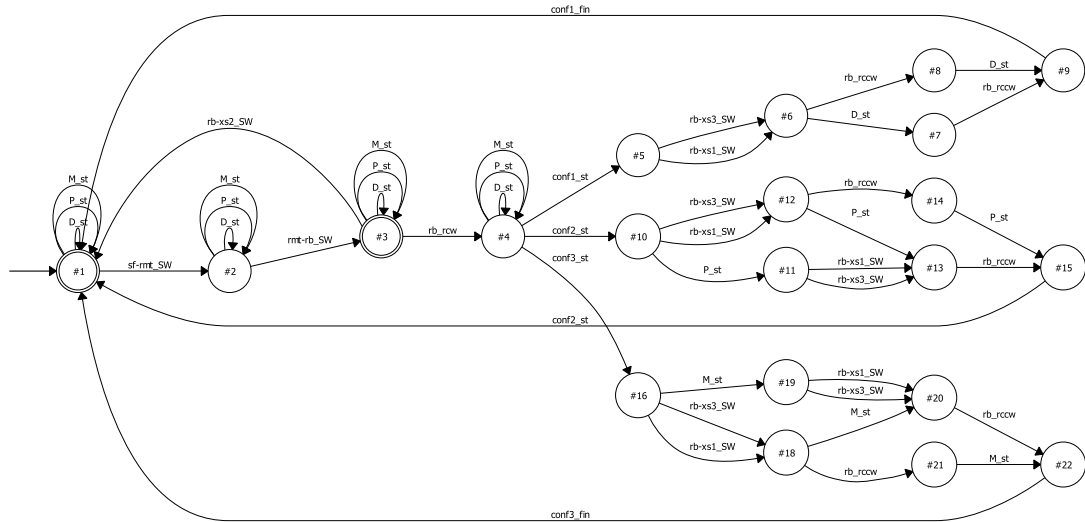


Figure 3.36: Coordinator of RMS.

3.6 RMS Simulation

Our experimental example setup consists of a total number of 6 electro-mechanical components including 1 Stack Feeder (SF), 1 processing machine represented by a reconfigurable machine tool (RMT), 1 Rotary Table (RB), and 3 Exit Slides (XS). In a practical experiment, the operation of such components is indicated by switch-keys & inductive sensors (uncontrollable events) and Dc motors (controllable events). Since there was no practical setup available for this thesis, we now describe how to perform simulation experiments of our example system.

A schematic of such simulation setup is shown in Figure 3.37. It shows the required infrastructure, which consists of a plant simulation that is connected to a controller simulation. Together, this establishes a closed-loop system in which plant dynamics interact with controller dynamics to satisfy a closed-loop specification. The plant and the controller interact by virtual external actuator and sensor signals, The actuator signals are generated by the controller simulation and the sensor signals are generated by the plant simulation.

In our simulation we use two different simulation tools for the plant and the controller. The controller is realized by a software called DESTOOL. It is a graphical software for controller synthesis and analysis methods for discrete event systems

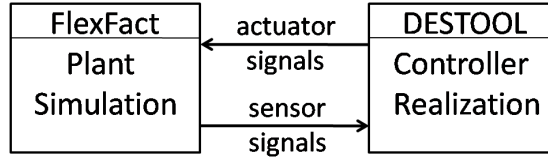


Figure 3.37: Schematic of the RMS setup.

(DES) based on the open source C++ software library `libfaudes`. We use the DES simulation feature of DESTOOL. A more detailed about DESTOOL is found in [9]. The plant simulation is realized by a software FlexFact. It is characterized as graphical environment for visualizing and simulating manufacturing components without having to construct a real system. A more detailed about software FlexFact in [9].

We verified our RMS design using the combination of DESTOOL and FlexFact. A screenshot of the setup is shown in Figure 3.38. The upper left-hand part shows the arrangement of our RMS in FlexFact, whereas the upper right-hand part shows the controller simulation in DESTOOL. The automata graph in Figure 3.38 represents the component \overline{S}_{conf_2} of our reconfiguration supervisor.

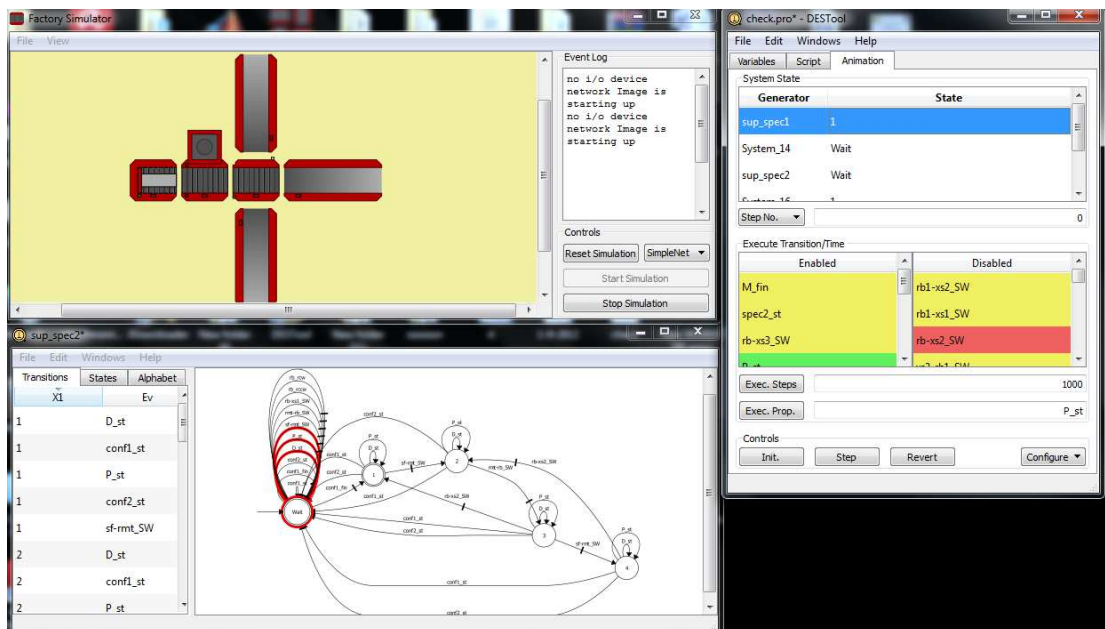


Figure 3.38: Simulation by controller synthesis.

CONCLUSIONS

This thesis investigates the supervisory control of reconfigurable manufacturing systems (RMS). Such RMS are a new manufacturing paradigm for the product types and quantities. The main contribution of the thesis is the formulation of the supervisory control problem using discrete event system (DES) models and a new controller synthesis algorithm. This new algorithm differs from existing work on this topic: we address both switching between different configurations and impose additional behavioral specifications during reconfiguration. In this thesis, this allows to automatically reconfigure an RMS and at the same time change the operation of reconfigurable machine tools (RMT).

In the formulation of this thesis, the new algorithm solves a problem defined as state attraction under language specification. This definition achieves the requirements for two different problems at the same time. The first problem corresponds to a state attraction problem, which requires that, after a bounded number of transitions, we reach a desired state of the plant state space as fast as possible. The second problem corresponds to a classical supervisory control problem in the form of a language specification. It requires to find a controller such that a given language specification must be fulfilled. Together, the solution of both problems allows to move the plant to a desired state and, on the way to the desired state, we fulfill a language specification. The thesis also provides a computational procedure to solve the stated problem.

In addition to the theoretical formulation, the reconfiguration supervisor design is applied to a laboratory example of an RMS that incorporates an RMT. First, a detailed model of RMS and RMT are obtained. Then, the developed algorithm is applied to find a reconfiguration supervisor. Finally, it is shown that the designed supervisors can be re-used in case a new configuration is added to the RMS.

REFERENCES

- [1] Brave, Y., Heymann, M. (1990), Stabilization of Discrete-Event Processes, *Int. J. Control*, 1101–1117, Vol. 51.
- [2] Brave, Y., Heymann, M. (1993), On optimal attraction of discrete-event processes, *Information Sciences*, 245–276, Vol. 67.
- [3] Cassandras, C.G., Lafortune, S. (2008), *Introduction to discrete event systems*, Second edition, Springer.
- [4] Hoda, A., ElMaraghy, M.G. (2009), *Changeable and Reconfigurable Manufacturing Systems*, Springer Series in Advanced Manufacturing.
- [5] Koren, Y. (2010), *The global manufacturing revolution*, Wiley.
- [6] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Brussel H.V. (1999), Reconfigurable Manufacturing Systems, *CIRP Annals – Manufacturing Technology*, 527–540, Vol. 48.
- [7] Kumar, R., Garg, V., Marcus. S.I. (1993), Language Stability and Stabilizability of Discrete Event Dynamical Systems, *SIAM J. Control Optim*, 132–154, Vol. 31.
- [8] Lafortune, S., Chen, E. (1990), The infimal closed controllable superlanguage and its application in supervisory control, *Automatic Control, IEEE Transactions on*, 398–405. Vol.35., no. 4.
- [9] libFAUDES. (2006-2012) libFAUDES software library for discrete event systems. [Online]. Available: www.rt.eei.uni-erlangen.de/FGdes/faudes.
- [10] Mehrabi, M.G., Ulsoy, A.G., Koren, Y. (2000), Reconfigurable manufacturing systems: Key to future manufacturing, *Journal of Intelligent Manufacturing*, 403-419. Vol. 11.
- [11] Nooruldeen, A., Schmidt, K.W. (2012), State Attraction under Language Specification for Discrete Event Systems, *Automatic Control, IEEE Transactions on* (under review).
- [12] Ramadge, P.J., Wonham, W.M. (1987), Supervisory control of a class of discrete event processes, *SIAM J. Control Optim*. 206-230. Vol. 25., no. 1.
- [13] Schmidt, K.W. (2012), Computation of Supervisors for Reconfigurable Machine Tools, in *Discrete Event Systems, Workshop*.

- [14] Schmidt, K.W. (2011-2014), Yeniden yapılandırılabilir üretim sistemlerinin kontrolcu tasarımı, bozukluk tanılaması ve bozukluktan toparlanması için formal bir işçerçevesi ve sürekli işakışı, TÜBİTAK Carrier Award 110E185.
- [15] Schmidt, K., Breindl, C. (2011), Maximally Permissive Hierarchical Control of Decentralized Discrete Event Systems, Automatic Control, IEEE Transactions on, 723-737, Vol. 56, no. 4.
- [16] Schmidt, K.W., Breindl, C. (2012), A Framework for the Stabilization of Discrete Event Systems under Partial Observation, Information Sciences (submitted).
- [17] Schmidt, K., Moor, T., Perk, S. (2008), Nonblocking Hierarchical Control of Decentralized Discrete Event Systems, Automatic Control, IEEE Transactions on, 2252-2265, Vol. 53, no. 10.
- [18] Wonham, W.M. (2010), Supervisory control of discrete-event systems, Lecture Notes, Department of Electrical and Computer Engineering, University of Toronto.

APPENDIX

CV

PERSONAL INFORMATION

Surname, Name: Nooruldeen, Anas

Nationality: Iraqi (IRQ)

Date and Place of Birth: 20 June 1988 , Kirkuk

Marital Status: Single

Phone: +90 531 498 34 37/ +964 770 219 23 77

email: anas_ali605@yahoo.com

EDUCATION

Degree	Institution	Year of Graduation
MS	Çankaya Univ. Electronic and Communication Engineering	2012
BS	College of Technology/Kirkuk Electronic and Control Engineering	2010
High School	Kirkuk Centralized	2006

FOREIGN LANGUAGES

Arabic, English, Turkish

PUBLICATIONS

A. Nooruldeen and K. W. Schmidt, "State attraction under language specification for discrete event systems," Automatic Control, IEEE Transactions on (under review), 2012.

HOBBIES

Read newspapers and journals, Soccer