THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Towards Automatic Generation of Formal Models for Highly Automated Manufacturing Systems

ASHFAQ FAROOQUI

Department of Electical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2018

Towards Automatic Generation of Formal Models for Highly Automated Manufacturing Systems

ASHFAQ FAROOQUI

Technical report number: R006/2018
ISSN 1403-266X

Department of Electical Engineering
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone + 46 (0)31 – 772 1000

Typeset by the author using LaTeX.

Chalmers Reproservice
Göteborg, Sweden 2018

to those who seek

Abstract

The manufacturing industry is undergoing a digital revolution, often referred to as Industry 4.0. The aim of this revolution is to transform the factories into, so called, *smart factories*. These smart factories will be modular, decentralized, and interconnected, to achieve higher level automation and flexibility. Additionally, a smart factory will have a *digital twin*, a virtual replica that allows testing, monitoring, and visualization of the factory behavior. As these factories are aimed to be completely automated, ensuring correctness and safety of the control logic in each sub-system of the factory is of utmost importance.

The need for having digitalized tools that support operators and engineers was identified in a survey that was conducted to understand the problems faced during maintenance of manufacturing systems. To this end, this thesis provides an architecture that can be applied on old legacy systems as well as new state-of-the-art systems to collect data from the factory floor. The data obtained can be visualized in the form of Gantt charts to help operators keep track of the execution of the station. Furthermore, a model that captures the behavior of the system can be created by applying Process Mining algorithms to the collected data.

Model-based techniques have shown to be beneficial in developing control logic for highly automated and flexible manufacturing systems, as these techniques offer tools to test and formally verify the control logic to guarantee its correctness. These formal tools operate on such a model of the behavior of the system. However, manually constructing a model on which these tools can be applied is a tedious and error prone task, seldom deemed to be worth the effort. Thus, supporting engineers to build models will improve the adoption of formal tools within the manufacturing industry.

In order to obtain a formal model during the early development phase of the manufacturing system, this thesis studies the possibility to automatically infer a model of a system by interacting with its digital twin. The suggested L^+ algorithm, an extension of the well-known L^* algorithm, shows that it is possible to automatically build formal models in this way. Additionally, certain shortcomings are identified and need to be addressed before being able to these methods in a practical setting.

Keywords: Formal Methods, Industrial Automation, Automata Learning, Visualization, Operations

Acknowledgments

When I embarked on this long and lonely journey exploring the realms of academia as a PhD student, I had not anticipated the way it would transform me. Now, here I am at the half-way mark eager to move ahead; but stopping to reflect upon my metamorphosis so far, and those that have contributed to this metamorphosis. Like in all journeys, the people encountered have left an impact; some more than others, but each profound and beautiful in their own ways. And to each, I owe a debt of gratitude. This journey would certainly not have been possible without the guidance, support, and patience of many important people.

First of all, I would like to thank my supervisor Martin Fabian. Your ability to ask the most pressing questions and provide blunt and honest feedback has helped me mature as a researcher. The detailed comments on the nitty-gritties of writing have helped me develop my skills as a writer, and for that, I am ever grateful. You have been an impeccable example of what a supervisor must be, not just for the guidance and support you provide, but for teaching me the what, why, and how, of supervision.

I would also like to thank my co-supervisor Petter Falkman for, firstly, for offering me this opportunity to enter academia. Secondly, and more importantly, for helping me keep my work anchored by helping me find use cases for all my ideas.

Håkan Pettersson from Volvo Cars Corporation was a great host during my short stint at Volvo Torslanda. Thank you!

My gratitude to Fredrik and Kristoffer for the insightful discussions that have helped change my perspectives at times I needed to most. To all the present and former members of the “discrete klubb” a big thank you for all the interesting discussions and support in different forms.

Stephen King says “The scariest moment is always just before you start,” I am grateful to Raghav, Charul, Martin (Viktorsson), Per-Lage, and Patrik who not just inspired me to embark on this journey but whose support helped me get over the scariest times.

When one has not written a thesis, the customary acknowledgments to the author’s family seems meaningless, but when one has spent long weekends and evenings absent from family activities, it reaches its full meaning. This journey

would have been impossible without the constant support of loved ones. My family, though far away, are a source of strength and support. I am forever grateful to my parents for being ever so supportive in everything I have done. Tania, my wife, has been by my side through the ups and downs of this journey. More importantly, she has helped me learn the beauty of balance; the balance that has helped sustain this journey; the balance that has made this journey even more beautiful. Thank you!

Lastly, a big cheers to the Free Software communities, chiefly, Manjaro – my platform for a few years now; Emacs – for being everything but a good text editor; And, LaTeX– for making writing tedious but beautiful!

Ashfaq Farooqui
Göteborg, August 2018

This work has been supported by Vinnova FFI VIRTCOM (2014-01408), ITEA3 VINNOVA ENTOC (2016-02716), VINNOVA LISA 2 (2014-06258), and VR SyTeC (2016-06204).

List of Publications

This thesis is based on the following appended papers:

Paper 1 **Ashfaq Farooqui**, Patrik Bergagård, Petter Falkman, and Martin Fabian. Error Handling Within Highly Automated Automotive Industry: Current Practice and Research Needs. *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA),* 2016, Berlin, Germany.

Paper 2 **Ashfaq Farooqui**, Kristofer Bengtsson, Petter Falkman, and Martin Fabian. From Factory Floor to Process Models: A Data Gathering Approach to Generate, Transform, and Visualize Manufacturing Processes. *Submitted for possible journal publication.* 2018

Paper 3 **Ashfaq Farooqui**, Petter Falkman, and Martin Fabian. Towards Automatic Learning of Discrete-Event Models using Queries and Observations. *Submitted for possible journal publication,* 2018

In what follows, these papers will be referred to as Paper 1, Paper 2, and Paper 3, respectively. The individual contributions of each paper are outlined in Chapter 4.

Other publications

The following publications, authored by the author of this thesis, are relevant but not included in the thesis:

Ashfaq Farooqui, Kristofer Bengtsson, Petter Falkman, and Martin Fabian. Real-time Visualization of Robot Operation Sequences. *2018 IFAC Symposium on Information Control Problems in Manufacturing (INCOM),* 2018, Bergamo, Italy.

Ashfaq Farooqui, Petter Falkman, and Martin Fabian. Towards Automatic Learning of Discrete-Event Models from Simulations. *14th IEEE Conference on Automation Science and Engineering (CASE),* 2018, Munich, Germany.

Contents

Part I

Introductory Chapters

Chapter 1

Introduction

The complexity of the manufacturing industry constantly increases to keep up with advancements in technology, market trends, legislative requirements, and most of all high quality products. Industry 4.0 [1], also called the *fourth industrial revolution*, can be seen as a collection of various technologies – Internet of Everything (IoE), Cyber-physical Systems (CPS), and smart factories – to create the next generation of industrial systems [2]. From the design principles of Industry 4.0 provided by Hermann et al. [2], distributed modular systems are key to build these next generation factories. The different distributed modules in an Industry 4.0 setting are normally provided by different manufacturers and have different properties, but when put together need to work seamlessly. In general, the systems developed consist of several industrial robots supported by conveyors and fixtures, and are designed to be completely automated with minimal manual handling. The system design must not only take into account flexibility, efficiency, and development time, but also account for fault tolerant behavior.

Development of these complex automated manufacturing systems is a demanding task. In the automotive industry, the development process typically begins with the company describing the product and design requirements. Based on these requirements, suitable components are chosen and an overall layout is decided on, followed by physically building the system. In parallel, the *control system* is developed that is to control the physical system so as to manufacture the desired product. Physically building the manufacturing station is commonly known as *physical commissioning*. This includes installation of the physical system consisting of robots, conveyors, fixtures, sensors etc., and the control system that is responsible for control, supervision and coordination throughout the production. The requirements specified are not always accurate and often ambiguous. Hence, the resulting station undergoes several iterations before it can be used in production. These iterations can significantly delay the time to market and increase costs, thus they are undesirable.

In order to reduce time to market and save on costs in the long run, the use

of *Virtual Commissioning* (VC) [3] is rapidly increasing today. With VC, a simulation model of the station is first created using some simulation software, such as Process Simulate from Siemens [4] or Experior from Xcelgo [5]. Then, the different manufacturing scenarios are simulated to check if all requirements are fulfilled. Furthermore, the control system can also be connected to the simulation model and its logic can be tested. This is done to find and fix faults and bugs in the control logic. Additionally, by visually simulating the manufacturing system, errors due to collisions can be detected early on. Physical commissioning is then done only after the VC model with the control function is acceptable. By correcting most of the faults during the VC phase, unnecessary time need not be spent testing the physical system. Thereby, production can start earlier.

However, both physical and virtual commissioning require the control logic, which is there to ensure that the station works as intended and is safe to operate. Thus, its correctness is of utmost importance. Since the requirements typically keep changing during the development process, the control logic needs to be continuously updated and debugged, which makes its manual development a tedious and error prone task.

One approach to manage this type of development process, with ever changing requirements, is to rely on mathematically well-defined *formal methods* [6]. By applying formal methods, it is possible to analyze and understand the system in part and as a whole. Formal methods make use of computerized calculations to analyze the system using a formal model, usually in the form of a *discrete event system* (DES) [6], that models the system to be analyzed. Hence, engineers can focus on defining their system and then use computerized algorithms to analyze, verify, and validate the models.

While the use of formal methods eases the engineering task of building the control logic, the burden instead gets placed on building the models. As a model grows in size with incorporating more resources and their operation, the modeling burden grows exponentially. To alleviate this, the models can be built *modularly*, where the engineers build models of the respective resources and their operations, and compose them in a mathematically well defined way into a model of the overall system. In addition, modeling the requirements as DES, the control logic can even be automatically generated by *synthesis* [6], so as to be correct-by-construction, which further alleviates the risk of introducing errors.

The discussion provided above focuses on building models for the control logic of new manufacturing stations. Building new systems are not done very often, though, while changes to existing legacy systems are done daily in order to improve their behavior, such as cycle time adjustment, bug fixing, etc., or on a bigger scale to introduce new products or machinery. To be able to do this in a secure way avoiding disruptions of existing behavior and the production on the whole, manufacturers are interested in applying formal analysis and to test

changes before commissioning them to the physical system. However, a problem here is that, more often than not, virtually commissioned or formal models do not exist for legacy systems.

Manually creating the required models is hard and time consuming, requiring a high degree of knowledge in formal methods and automation system design. Incorrect or incomplete models are misleading and the use of formal methods may serve no purpose. To benefit from the use of formal methods, in terms of verification and synthesis of control logic, the task of building models needs to be done correctly. In order to do so, this thesis proposes a way to automatically create a model of the system using computerized tools.

1.1 Research Questions

This thesis aims to explore the following questions.

RQ1 *How does the manufacturing industry generally handle errors and perform maintenance? And what are the challenges faced?*

Error recovery techniques have been studied in academia [7, 8, 9, 10, 11], but these techniques have not really found their way into industrial practices. Identification of reasons for this disconnect might help bridging the gap between academia and industry.

RQ2 *How can operators be supported with tools and processes that will make it possible to make more data driven decisions?*

Maintenance of manufacturing systems is not an easy task. Decisions need to be made, tracked, and evaluated. Manually doing so is burdensome and often ineffective. Having digital tools to support operators can improve the quality of maintenance [12].

RQ3 *Is it feasible to automatically learn formal models of manufacturing systems? If so, what would be required to make it a reality?*

Model-based techniques have shown to be useful for a variety of reasons. The lack of usable models and the difficulty to create them manually is a deterrent to using model-based techniques to build and maintain manufacturing systems. Automatic creation of formal models has been studied [13, 14], but these algorithms are computationally heavy and have typically been designed for smaller applications. Recent advances in computer technology have led to computationally powerful computers, and availability of advanced simulation tools, which might hold the key to making it possible to learn a formal model of large manufacturing systems.

1.2 Objective and Contribution

The objective of this thesis is twofold. First, to highlight the problems faced by the automotive industry related to maintenance and error handling. Then, to propose methods, techniques, and tools to help during the commissioning and production phase so as to increase quality and efficiency. This is achieved by the following contributions:

- A survey to identify the problems faced by the industry and how these problems are currently solved.
- A survey of ongoing state-of-the-art projects that provide a glimpse into future manufacturing technologies.
- An approach to collect and transform data from the factory floor.
- Analysis and visualization of ongoing manufacturing processes in real-time using Gantt charts, and creation of a behavioral model using Process Mining.
- A study into methods that enable automatic generation of formal models from a simulation model of the system using the L^+ algorithm. The insights gained from this study will help identify avenues that will allow automatic creation of models for practical manufacturing systems.

The papers presented in this thesis build upon the ideas presented in this introduction. Paper 1 highlights the common problems faced by the automotive manufacturing industry with regard to error handling and presents work needed to address these problems. An outcome of this study was the need to have access to data from the factory floor to get a better understanding of the systems. Paper 2 presents a flexible and scalable architecture to collect and transform data from the factory floor. This architecture can be applied to existing and new manufacturing stations. The resulting data is then transformed into a more understandable abstraction in the form of operation descriptions. This data is then used to build models of existing manufacturing stations using process mining techniques to help improve and maintain the stations.

Paper 3 presents the possibility to build a formal model of a system from its simulation model. The functions of the simulation model are modeled as independent operations which can be executed from an external interface. The L^+ algorithm, an extension of the L^* algorithm [13], is interfaced to the simulation software, where it can observe the state of the simulation. Then, by posing queries the algorithm constructs a formal model capturing the behavior of the system.

1.3 Method

The outcome of this thesis is a set of activities and tools that can help build more reliable and efficient manufacturing systems. Most of the work involved implementing and testing the algorithms to demonstrate the applicability of the proposed methods. From a more theoretical perspective, the main research activity was to identify gaps in the field of automated modeling that will improve the practical applicability of these methods.

In general, the method followed was to first, by interacting with industry representatives find pressing problems that are faced by the industries; problems related to error handling and, more specifically, modeling of automated manufacturing systems, were considered. Then, solutions to these problems were proposed by suggesting a method – as a set of activities – to follow, along with accompanying algorithms. These algorithms were then implemented to demonstrate and evaluate the possibility of applying them to real-world scenarios. The evaluations showed that it is indeed possible to apply in practice the presented approach. However, there are also some parts missing and parts that need to be improved. These are summarized in Chapter 5.

1.4 Outline

This thesis is divided into two parts. This first part contains introductory chapters that aim to help the reader to better understand the ideas and concepts discussed in the included papers, and also to provide a direction of the research work. The second part contains the included papers.

This introduction presents a high-level process of building a manufacturing station and highlights the problems normally faced by engineers involved with building and maintaining these stations. Chapter 2 puts the thesis into perspective by positioning this work in the broader picture. To this end, existing industry practices and challenges are presented, followed by a new updated work-flow that aims to support engineers and operators during the development of a manufacturing system. Chapter 3 introduces the reader to the field of automatically inferring formal models. Chapter 4 provides a summary of the included papers. Finally, some concluding remarks in Chapter 5 sum up the work and provide a glimpse of future work.

Chapter 2

The Broader Picture

The aim of this chapter is to provide a context and help position the contribution of this thesis within the broader picture. In doing so, this chapter will give a general overview of existing methodologies followed during the development of a manufacturing system, and some of their challenges. Additionally, this chapter, will outline the new methodology that is proposed to help mitigate the challenges. The tools required to achieve the suggested methodologies in reality are also briefly introduced.

The traditional procedure followed during the development of a manufacturing system starts with a pre-study on the product that will be manufactured. The pre-study results in a set of requirements that need to be fulfilled by the manufacturing system; a bill of materials that contains the components, such as, robots, conveyors, fixtures, etc, that need to be procured; and the system layout representing the physical placements of those components. The bill of materials is procured and the physical building of the system is started, in smaller functional parts or as a whole. Based on the requirements, different tasks and actions needed to manufacture the product are planned. This is known as *process planning* [15, 16]. The end result of the process planning stage is the generation of the control logic. This control logic is responsible for controlling the resources in the manufacturing system so as to fulfill the objective of the manufacturing system in an efficient and safe manner. The control logic then needs to be tested on the physical system, which is usually done in two phases. In the first stage, components are tested individually or in groups. The second stage is performed after the system is physically commissioned. Here, testing is done until all requirements are satisfied, and the system behaves satisfactorily.

The consideration of control logic later in the development phase has several negative consequences resulting in high financial costs and loss of time. As seen in Paper 1, fixing software bugs might, as a side effect, introduce more bugs. Additionally, performing the tests on the physical system increases the risk of collisions and component failures. During the survey conducted in Pa-

per 1 it was also found that the initial testing done at the line manufacturers site was functional testing. Later, only when the complete system is physically commissioned, complete end-to-end testing is performed. Directly testing on the physical system might result in breakage and may require a fresh order of components. All these factors add to the unnecessary effort, cost, and time expended during the physical commissioning phase.

Development of manufacturing systems is just the first phase of its life-cycle. Building these stations is expensive, and the manufacturing company expects them to last for several years to even out the cost. Hence, these systems need to be well maintained to ensure good productivity during their life-cycle. Paper 1 presents the current industrial state of maintenance procedures. The concluding points in Paper 1 are to have a work-flow that incorporates ways to fix bugs in the software safely, and a possibility to digitalize the system to be able to use digital tools to monitor, visualize, and reason about it. Hence, the tools and processes created in the new work-flow need to not only account for a better manufacturing process but also need to cater to maintenance requirements.

2.1 The New Workflow

The Division of Systems and Control, at the Department of Electrical Engineering at Chalmers University of Technology, has been active in developing a work-flow and tools to support the development of manufacturing systems. A high level work-flow that is pursued at the department is presented in Figure 2.1. This new work-flow starts similar to the old work-flow by identifying the bill of materials. But the process planning starts already at the initial phase along with the bill of materials to identify the process, tasks and actions needed, resulting in the *bill of processes*. Based on the bill of processes, the components identified are digitally created in a 3D simulation environment. These digital replicas are placed according to the required system layout. At this point, potential collisions can be identified, and the layout can be updated to ensure safe operation. The engineers then take the bill of processes and convert this to actions to be performed in the simulation software. These simulated actions need to be transformed into control logic that can be run on a Programmable Logic Controller (PLC). The system is *virtually commissioned* by connecting the PLC to the simulation tool to verify functionality and test for errors. At this point, the bill of materials is finalized, and the components are procured. Following which the system is physically commissioned.

During the development phase, requirements are constantly changing due to new insights. Hence, the control logic needs to be constantly updated and tested to ensure that the requirements are met. One approach is to use mathematically well-defined formal methods to verify the behavior of the system and to generate

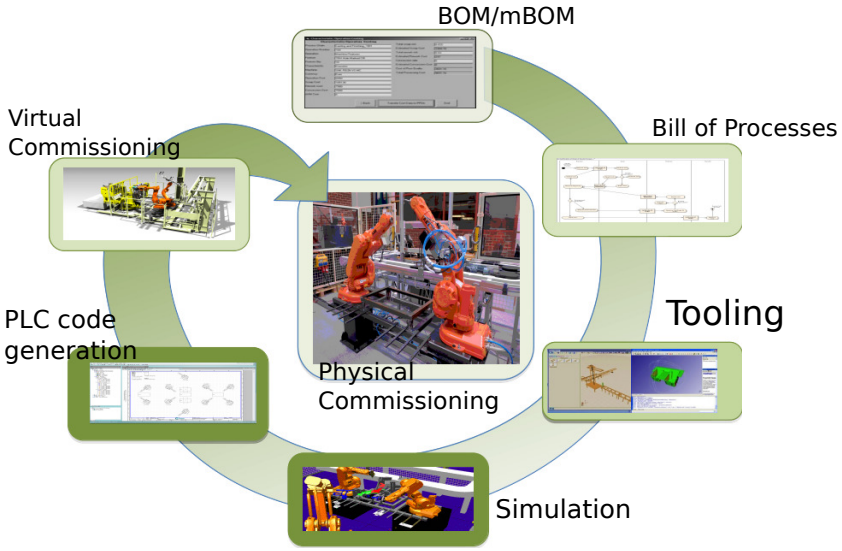


Figure 2.1: The new workflow

the control logic. With the use of formal methods, the focus is shifted from manually developing and testing the control logic, to developing a model that describes the behavior of the system.

To build a formal model the engineer needs to consider the level of abstraction needed to be described by the model. By modeling at a very low level the engineer risks making the model unmanageably large. On the other hand, by modeling at a high level, a number of details could be missed, and the model might not be usable. Hence, finding a level of abstraction that suits the system is key to developing a model. Once a suitable level of abstraction is decided, the engineer has to ensure that the model correctly captures the behavior. This is where this thesis contributes towards the development of manufacturing systems. It looks at possibilities to automatically infer a behavioral model of the system.

A by-product of the defined work-flow is the availability of a *digital twin* – a digital replica of the physical system. This digital twin behaves in the same way as its physical counterpart and can be used for monitoring, testing, and modifications. To enable the effective use of a digital twin, additional tools need to be created that can synchronize between the physical system and its digital twin. These tools need to capture and store operating data from the physical system for further processing.

The remainder of this chapter will introduce the different components needed to follow this thesis, specifically, virtual commissioning and formalisms for formal methods.

2.2 Virtual Commissioning

Simulation technology has come to a point where it is now possible to simulate systems at the sensor level. Software programs such as Process Simulate [4], Xcelgo Experior [5], and ABB Robot studio [17] are capable of simulating conveyors, fixtures, robots, and even humans, to a relatively high level of accuracy.

These simulation tools can also be controlled externally, for example, by using a PLC. Such a setup would constitute *virtual commissioning*, where a digital replica of the system is controlled by the PLC code that will eventually run on the shop floor controlling the physical system.

In order to be able to infer a model of the system, which is the aim here, the learning algorithm – the learner, introduced later in this thesis – requires an interface to the virtually commissioned system. This interface must allow the learning algorithm to execute actions which are at a pre-defined abstraction level, and allow the learning algorithm to observe the output of the virtual model. An example of such an interface is the OPC-UA [18] standard for communication. This standard is supported by many component vendors and can be used with the virtual and physical system. The OPC-UA, allows control and observation at the input/output level, and also facilitates observation of internal variables. Hence, it is an ideal candidate to use as an interface for the learning algorithm.

2.3 Modeling Operations

As discussed in Section 2.1, choosing a sufficiently moderate level of abstraction becomes important to create a model. The work in this thesis uses the abstraction of *operations* [19]. An operation is a task performed in a manufacturing system by one or more resources. Broadly, an operation can be defined as a set of actions that change the state of the system to accomplish an objective. The level of detail of an operation is not fixed, it could be used to define an update in one resource or can affect several resources. *Guards* are used to determine when a operation is allowed (or not allowed) to execute. These operations, along with their guards, are programmed such that they can be executed by the control system. A more formal definition is provided in Paper 3, Section 2.3.

2.4 Formal Methods

Formal methods are design techniques that use mathematical models to build software and hardware systems. These methods make use of mathematical proofs to ensure correctness. As systems become more complicated and safety becomes an important concern, a formal approach to the system design offers a certain

level of insurance.

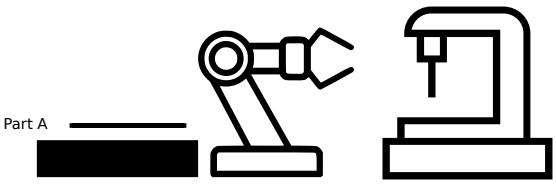
To apply formal methods, designers need to have access to a formal model that describes the behavior of the system. One of the many ways to create such a model is to make use of *finite-state machines* (a.k.a *finite-state automata* or simply *automata*) [20], which are commonly used to model discrete-event systems. The system is then abstracted into *events* and *states*, where the occurrence of an event moves the system from one state to another. The original state of the system, before the occurrence of any event is the *initial state*. A particular sequence of events can lead the system to some desired state, such a desired state is called an *accepted state* and the sequence of events an accepted sequence.

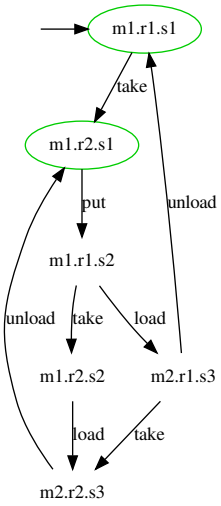
When talking about discrete-event systems as finite-state machines a sequence of events is referred to as a *word*. The set of words that are accepted by the finite-state machine is the accepted *language*. The automaton represents the *grammar* of the system; that is, when presented with a word, the grammar decides the validity of this word. Formal definitions of words and languages are provided in Paper 3, Section 2.

Chapter 3

Inference of Formal Models

Consider a small but realistic example consisting of a robot and a machine, as seen in Figure 3.1a. In this system, the robot takes the part (part A) from the pallet and puts it on the machine. The machine then loads the part, processes it, and unloads it. In the meantime, the robot can take the next part or do nothing. But the robot has to wait until the machine has unloaded its part before putting the next part on the machine. An automaton representing this setup is shown in Figure 3.1b, from which it is seen that the system is accepting as long as there is no part loaded in the machine.





(a) Representation of the physical system. (b) Representation of the system behavior.

Figure 3.1: A robot and a machine. The robot takes a part and puts it on the machine. The machine loads the part, and after processing unloads it.

Assume now that we do not have the model of this system, and want to obtain the model without manually building it. That is, we are provided access to the

actual physical system as in Figure 3.1a and would like to infer a model of it as seen in Figure 3.1b. There exist tools and techniques that would help to infer such models. Broadly, models can be inferred using *Grammar Inference* [21] or *Process Mining* [14] techniques.

The aim of this chapter is to provide a glimpse into existing ideas that deal with automatically inferring models. Both the fields of study, Grammatical Inference and Process Mining, are well established and quite vast. It is beyond the scope of this thesis to cover the details of the different algorithms. However, the general ideas employed in each of the fields will be discussed to equip the reader with the basics concepts. This will help interested readers to further delve into the details.

3.1 Grammar Inference

Grammar Inference (GI) has been studied both as a theoretical problem, where its goal is to uncover some hidden function, or as a practical problem of attempting to represent some knowledge as an automaton. GI finds its origin in various fields of study: computational linguistics, machine learning, formal learning theory, pattern recognition, and computational biology. Hence, it is also known by different names depending on the field: Automata Learning, Grammar Induction, Grammar Learning, etc. Though the different names can have different connotations, they all refer to similar ideas and processes.

The majority of GI algorithms work on generalizing some form of knowledge about the system to be learnt. The learning algorithm referred to as the *learner*, internally creates a representation of this knowledge. This representation is continuously refined and generalized to satisfy certain properties specific to that learner. When the learner is satisfied with the generalization reached, its internal representation can be converted into some meaningful representation. Of the many methods studied and developed under GI, several of them deal with learning finite-state machines. These methods can be classified as *Passive learning* or *Active learning*.

In-depth surveys of GI techniques are provided by [22, 23, 24].

3.1.1 Passive Learning

Passive Learning, sometimes referred to as Informed Learning, is a setting in which labeled data is provided to the learner, and the learner is tasked to find an automaton that generalizes this data. This data is usually an observation log from the system and consists of words. These words are labelled as accepted (or rejected) if they belong (or do not belong) to the accepted language.

Passive learning algorithms start by first generating a hypothesis from the available data. This is done by constructing a *prefix tree acceptor* (PTA), which is a tree-like automaton constructed by looking at the accepted words in the data. This PTA can be constructed in linear time and contains no loops or converging paths. In the example above, accepted words could be the set $\{\langle$take$\rangle$, $\langle$take, put, load, unload$\rangle$, $\langle$take, put, take, load, unload$\rangle\}$, and non-accepting words could be $\{\langle$take, put$\rangle$, $\langle$take, put, load$\rangle$, $\langle$take, put, take$\rangle\}$. The first step is to construct the PTA using the accepted data. The PTA obtained is a crude representation of the available data as seen in Figure 3.2.

The next step is to refine the PTA. The method of building the PTA and then refining it using non-accepting data samples is called *regular positive and negative inference* [25]. This method provides a basis for more advanced algorithms that focus on the refinement phase. A number of different strategies for PTA refinement have been presented in the literature [21, 26, 27, 28]; a detailed survey of those is beyond the scope of this thesis.

The most common refinement technique, though, is *state merging* [29]. The merging of states usually consists of three stages, the first is the search to identify the states to be merged; then the actual merging, where the merged states are represented by a single state yet retain the incoming and outgoing transitions; the final stage is called *promotion*, which keeps track of the states already tested and the next states to be evaluated. Several algorithms have been suggested to efficiently perform the search, merging and promotion. In [27] a general survey of state merging algorithms is presented where they are classified as *Exact Algorithms* or *Approximate Algorithms*.

Exact algorithms aim to create a model that exactly represents the input data. Some such algorithms are MMM [30], BICA [31], and EXBAR [32], all of which start with a basic trivial model as initial PTA. The algorithms then start looking for states to merge. On every merge done the algorithms search the input data to check if the samples are consistent with the automaton. Searching through the input data for inconsistencies is expensive. The algorithms mentioned above use different techniques and heuristics to perform this search efficiently.

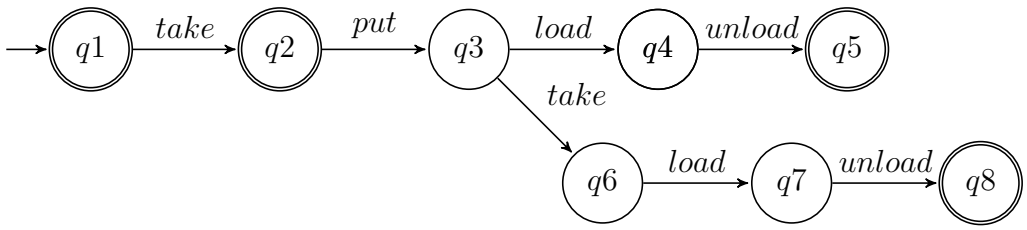


Figure 3.2: The PTA generated from the set of accepted words $\{\langle$take$\rangle$, $\langle$take, put, load, unload$\rangle$, $\langle$take, put, take, load, unload$\rangle\}$

Approximate algorithms, on the other hand, rely on heuristics to provide an automaton that approximately describes the input data. Examples of these algorithms are EDSM [33], SAGE [34], and ED-BEAM [32]. These algorithms work in similar ways to their exact counterparts, the difference lies in the way they perform the state merging. Additionally, these algorithms keep updating the PTA by backtracking and improving previous merges.

In order to obtain an accurate model of the system, the learner needs to have access to accepting and non-accepting data [35]. In a manufacturing context access to accepting data is usually never a problem. However, there does not seem to exist ways to obtain non-accepting data. Hence, the models identified using passive learning methods are limited to the fact that they represent only the observed behavior. One major problem when applying passive algorithms is the state-space explosion. It has been proved that learning from sampled data is NP-complete [35]. Thus, most of the algorithms are created and tested on problems with a significantly smaller state-space [27] compared to what is needed in a manufacturing setting.

Paper 3 uses ideas from passive learning in conjunction with active learning methods to obtain a more accurate model of the system.

3.1.2 Active Learning

Active Learning, or learning with queries, is a setting where it is possible to interact with an oracle. The algorithms available under active learning make the assumption that there is a *minimal adequate teacher*, which is an oracle that can answer queries. An active learner poses queries to the oracle and based on the responses constructs a model.

The analogy of a teacher and student fits well to explain the working of general active learning techniques. The student is the learner, and the teacher is the oracle. The student is given prior knowledge about the different events possible, in the example above it would be the set {take, put, load, unload}. And, the student is allowed to pose two types of queries to the teacher. The first type is about the membership of a given sequence of events; to this the teacher can respond positively if the sequence results in an accepted state, else the response is negative. The second type of query, called equivalence query, occurs when the student presents a hypothesis model to the teacher. If the teacher responds positively, that is, acknowledges that the model learnt by the student represents the actual system fairly accurately, then a model is found and the learning terminates. Else, in case the teacher finds the model incorrect, the student is presented with a counterexample – a sequence of events that is allowed in the hypothesis but not in reality, or vice versa. The student then updates its model to exclude or include the behavior of the counterexample, and continues asking queries. This

process iterates until a fairly accurate model is found.

For the sake of an example, lets say the student starts by asking if ⟨take⟩ is a member; the teacher responds positively. If the student then proposes a model with a single event, then since the model is invalid the teacher provides a counterexample by giving, say, the sequence ⟨take, put, load, unload⟩ The student will then take the presented counterexample into consideration, make new queries, and eventually present a new hypothesis. This process continues until the teacher decides that an acceptable model is found. The procedure highlighted above is intended only to provide an analogy into the actual learning process and to show how this type of learning is closer to how humans learn. The actual algorithms are far more complex. One of the most fundamental active learning algorithms is the L^* algorithm introduced by [13], a detailed explanation for which is provided in Paper 3, Section 4. This algorithm is known to run in polynomial time and guarantees termination [13].

The L^* algorithm has been a starting point for most of the research in the field of active learning [13]. From an algorithmic perspective, there have been only a handful of improvements and new approaches suggested. Schapire et. al. [36] improve the L^* algorithm by handling counterexamples that include a homing sequence when it is not possible to reset the target system. In the case when a robot is allowed to explore its surroundings to infer a map, it is tedious to always reset the robot to its initial position. Hence, Schapire et. al [36] present an algorithm that creates a homing sequence. Using this homing sequence allows the robot to infer its current state. Kearns and Vazirani [26] introduce the idea of discrimination trees to internally represent the knowledge of the system. The idea of discrimination trees is further explored by Malte et al. [37] who suggest the TTT [1] algorithm.

The L^* algorithm has been used successfully in practical settings to learn automata. Active automata learning has been applied to verify communication protocols using Mealy machines [38, 39]. By using a suitable abstraction interface, Arts [40] learn IO automata. Other techniques are directed towards learning models of software systems; Malte et al. [41] apply active automata learning towards learning models of software programs modeled as register automata, while Smeenk et al. [42] focus on learning embedded software programs.

The active learning algorithms assume the existence of an oracle. If a computerized oracle would exist, then there would be no problem to solve, since the model would be available in the oracle, and the task would then be to extract the model from the oracle. However, with the advancement of technology and easy availability of simulation software, it is possible to create a digital twin of the actual system to play the part of an oracle. These simulations are built up of sev-

[1] The name is derived from *Spanning Tree*, *Discrimination Tree*, and *Discriminator Trie*; the three concepts fundamental to the algorithm.

eral functions that can be executed using an external interface to the simulation software. For example, the event "take" would have a function that makes the robot pick the part from the pallet, or "put" would place the part on the machine, and so forth. The teacher is then no longer an oracle, but an interface to this simulation. It can then reply to queries by simulating the given sequence. The accepted states are then identified by observing the simulation, which could be achieved by reading the internal variables of the simulation and defining a predicate expression over these variables. This setup is further elaborated in Paper 3, Section 6.

However, in order to be able to apply these techniques to learn models there needs to be a way to find counterexamples. Queries for membership are fairly easy to handle, equivalence queries on the other hand are proved to be an NP-complete problem [43]. Hence, finding counterexamples is a bottleneck for active learning methods. In the case when the model of an existing system is to be created, which also has a virtually commissioned counterpart, Paper 3 integrates active and passive learning methods and presents the L^+ algorithm that uses observation data from the actual system to find counterexamples.

3.2 Process Mining

Process mining [14] is a field of study comprised of process discovery, conformance checking, and model enhancement. Process discovery, similar to passive learning, aims at creating a process model that imitates the observed process. The difference between process discovery and passive learning lies in their objectives. While process discovery takes a more pragmatic approach towards building and analyzing the underlying process in large organization, passive learning is a more theoretical method focusing on finding grammars that represent the data. What sets process discovery apart from passive learning is its integration with the other components of Process Mining, conformance checking and model enhancement, for investigating and maintaining the process models.

Process discovery techniques are used to understand task flows in large organizations where tasks can be performed for various resources or individuals. To do this, these techniques rely on process logs as input. The minimal requirement for the logs is a tuple containing `<caseId, event name, attributes>`. Here the `caseId` is a unique identifier that identifies the case being handled; in a manufacturing context it could be viewed as the product on which operations are run. The `event name` identifies the task executed. A number of other `attributes`, such as time stamps, the name of the resource that performs the operation, product variant, can be appended to improve the analysis. The resulting output from a discovery algorithm is a work-flow model that can be represented by formalisms such as Petri nets [44], Transition

Graphs [45], or Business Process Model and Notation diagrams [46].

The basic idea, in almost all Process Mining algorithms, is to construct a relationship table from the input logs. The relationship table contains information about the position of each event relative to the other events. In the example above, irrespective of the actual sequence of events, every product will log the sequence ⟨take, put, load, unload⟩. The relationship table will thus reflect a "directly follows" relation between the pairs ⟨take, put⟩, ⟨put, load⟩, and ⟨load, unload⟩. Thus, the resulting work-flow model is a straight sequence of these events. In a more complex setting, where there are different paths observed, the relationship table can additionally describe relations such as parallel or arbitrary.

The process outlined above closely resembles the alpha algorithm [47], the most basic algorithm in Process Mining. However, the alpha algorithm is not immune to noise in the logs, and also cannot be applied to systems that contain loops in their execution. Several advanced Process Mining algorithms have been presented that tackle these problems. The Heuristic Miner [48], for example, builds a relationship table that contains the number of times an event occurs before or after other events. Then, using heuristics to normalize this table, the algorithm constructs a model representing the different relationships. Another such algorithm, the Fuzzy Miner [49], aims to find closely related patterns and group them into clusters. In doing so, it becomes possible to not focus on the details, thus an abstract process is presented for possible further analysis.

A model defining relations between events is not the only form of output achievable from Process Mining. It is also possible to get a model from various perspectives and analyze different properties. For instance, by logging the execution time and executing resource for each event as additional attributes it is possible to analyze the system for bottlenecks and resource utilization. To this end, a model representing the relationship between the resources is first created. Then by aggregating the time spent at each resource, this model can include the time each resource is occupied.

Process mining has shown significant benefits in understanding underlying task flows, bottlenecks, resource utilization and many other factors within large corporations [50, 51], and has also proved beneficial in health-care [52, 53, 54] to learn and improve the underlying processes. Within the manufacturing domain, though, there has been only a handful of studies of applying Process Mining. Yang et. al. [55] and Viale et. al. [56] present a method to apply Process Mining on manufacturing data. While the former uses structured and unstructured data generated from a manufacturing system along with operators or workers to provide domain level knowledge, the latter works with definitions of the system provided by domain experts to find inconsistencies between the model and the actual process. Yahya [57] shares interesting insights into using process mining to understand manufacturing systems using artificially created logs. However,

there seem to be two main factors missing when aiming to apply Process Mining to manufacturing systems:

1. There is a lack of methods to capture and collect usable data from the factory floor.

2. There is a lack of a defined data structure/abstraction useful for generating models from factory data.

Paper 2 uses Process Mining to analyze manufacturing systems by collecting data from the factory floor. Section 2 of Paper 2 provides a general architecture that generates an event stream of program pointer data from the robots in a manufacturing system. Furthermore, the paper elaborates on transforming the generated event streams into the usable abstraction of *operations*. This abstracted data is used as input to Process Mining algorithms to obtain models and analyze the system. Section 4.3 of Paper 2 discusses visualization from an operation perspective, where the relations between different operations are studied to identify unique patterns that correspond to different products. Section 4.4 then shows the use of Process Mining to visualize how the product flows between the resources.

Chapter 4

Summary of Contributions

This chapter provides a brief summary of the papers that are included as part of this thesis.

Paper 1

> **Ashfaq Farooqui**, Patrik Bergagård, Petter Falkman, and Martin Fabian.
> Error Handling Within Highly Automated Automotive Industry: Current Practice and Research Needs. *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA),* 2016, Berlin, Germany.

This paper presents a study on error handling in the Swedish automotive manufacturing industry, specifically the body-in-white segment. To this end, a survey was conducted with several industry partners to get a glimpse into what were the most common errors encountered and the measures taken to avoid them. Additionally, the paper looks at ongoing research that is aimed towards making manufacturing highly automated. Based on the survey with industrial partners and ongoing research, the paper identifies future directions of work that will help industries handle error scenarios. Tool breakage, resource malfunctions, software bugs, and emergency stops were a few of the most common issues faced in the industry. To be equipped to handle these problems operators need to be well trained. Therefore additional effort is required to ensure training handles the standard scenarios. A common need towards alleviating the problems identified is the digitalization of the system and its processes.

Paper 2

Ashfaq Farooqui, Kristofer Bengtsson, Petter Falkman, and Martin Fabian.
From Factory Floor to Process Models: A Data Gathering Approach to Generate, Transform, and Visualize Manufacturing Processes. *Submitted for possible journal publication.* 2018

This paper provides a software architecture to collect data from robot manufacturing stations. The suggested architecture was designed to be applicable to existing and new manufacturing stations. The collected data is abstracted into the form of operations and can be visualized in real-time aiding the operators. To show the applicability of the data, the software architecture was applied and tested on manufacturing stations consisting of several robots. Then, process mining methods are used to process the obtained data to construct a general model that represents the activities of the station. Process mining algorithms are also used to identify the relationship between the resources in the station.

Paper 3

Ashfaq Farooqui, Petter Falkman, and Martin Fabian.
Towards Automatic Learning of Discrete-Event Models using Queries and Observations. *Submitted for possible journal publication,* 2018

This paper presents an approach to integrate active and passive learning by introducing the L^+ algorithm, an extension to the L^* algorithm. The L^* algorithm, presented in Paper 5, shows the possibility to learn formal models from a simulated environment. A major bottleneck to practically use the L^* on manufacturing stations relates to finding counterexamples. The approach presented in this paper, specifically the L^+ algorithm, uses previously collected sequential operation data to find counterexamples to a model created by the L^* algorithm.

Chapter 5

Concluding Remarks and Future Work

Building correct and error free control logic for manufacturing systems is a challenge. Mathematically well-defined formal methods provide the possibility to analyze and understand the system. These formal methods can ease the task of building control logic by focusing on building models that define the behavior of the system, and analysis can then be performed on these models to verify that given requirements are guaranteed to be fulfilled. Doing so allows the engineer to focus on the behavior and not the underlying details of the system. However, the models grow exponentially as the number of resources and operations performed by them increases, and this makes it hard to manually create models. Hence, it would be beneficial to be able to automatically build models that can then be used by engineers to perform analysis by applying formal methods.

This thesis focuses on automatically building models of manufacturing systems to help during the maintenance stages, as well as during the early virtual commissioning phase. To help operators doing maintenance, the current state of the manufacturing system needs to be tracked and visualized in a way understandable by the operators. To this end, an architecture to collect and transform execution data from robots in manufacturing stations is presented in Paper 2. This data is used to aid operators, by visualizing the execution of the system, to better understand and service the station in a timely manner. Furthermore, the logged data is also used to infer a model that describes the behavior of the system using Process Mining approaches. This model describes the relationship between the different operations in the system, and the general product flow between the resources.

The possibility to collect data and build a model in real-time creates many more avenues to support operators. Live data from the manufacturing system can be replayed and simulated in the model, in doing so any behavior not already present in the model can be captured and presented to the operator for inspection.

Similarly, the generated data can be used to continuously update the model, that is, build the model in real-time. This live model can be verified against the requirements of the manufacturing system to find deviations. Furthermore, these models can be used to perform predictive maintenance on the resources, predict delays, predict the throughput, thereby supporting the engineers and operators.

Another area that this thesis focuses on is learning of formal models from a digital twin. Engineers can couple the digital twin with active learning during the development process to analyze the effects of changing requirements. By defining executable operations – sets of actions – that perform a specific task, the learner is tasked with inferring a complete model that describes the behavior of the simulation model. This helps in building flexible manufacturing systems in which the engineers need not spend additional effort to manually build models for every product variant. Thus, the learner can infer models for the different products, using the virtual model, based on their requirements.

In conclusion, this thesis studied problems faced during error handling in the manufacturing industry. Certain key problems were identified, and the remainder of the thesis presented tools and techniques to tackle them. The presented approaches were tested and validated on small systems and toy examples to show a proof of concept. However, to be able to fully use these benefits in an industrial setting, the algorithms need to be improved and implemented.

Future work

To be able to fully realize the work presented in this thesis and apply it in real demonstrations, there are several challenges that need to be handled. The state-space explosion problem is one of the major challenges faced while trying to learn models automatically. An approach to alleviate this challenge is to use richer formalism to describe models; one such formalism is Extended Finite Automata [58], an extension of finite-state machines that is built up of bounded discrete variables and uses guards and actions to read and update variables while executing transitions. In order to automatically generate guards for an Extended Finite Automaton, it might be worth to investigate if Binary Decision Diagrams [59] can be used to encode state information in the observation tables. Then, by consolidating the state information in a smart way, to generate guards for each transition.

Concerning generation of data from the factory floor as presented in Paper 2, the current work is limited to gathering data from industrial robots. However, most of the logic in complex systems is contained within the PLC. Therefore, generation of execution data from PLCs into a reasonable abstraction is the next natural step.

In a more broader context, a major challenge is to reason about the quality

of the obtained models. Models are not perfect; they represent only a part of the reality. But to be able to reason and make decisions by using such models, there needs to be some metrics to help classify them. These metrics can provide the engineer with a degree of confidence when performing analysis on the model, and also help reason about the meaning of these results in relation to the physical system.

Bibliography

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014. [Online]. Available: http://dx.doi.org/10.1007/s12599-014-0334-4

[2] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, Jan 2016, pp. 3928–3937.

[3] C. G. Lee and S. C. Park, "Survey on the virtual commissioning of manufacturing systems," *Journal of Computational Design and Engineering*, vol. 1, 2014.

[4] "Process Simulate." [Online]. Available: https://www.plm.automation.siemens.com/global/en/products/tecnomatix/assembly-simulation.html

[5] "Xcelgo Experior." [Online]. Available: https://xcelgo.com/experior/

[6] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[7] P. Loborg, "Error recovery in automation an overview," in *AAAI-94 Spring Symposium on Detecting and Resolving Errors in Manufacturing Systems, Stanford, Ca, USA*, 1994.

[8] P. Loborg and A. Törne, "Manufacturing control system principles supporting error recovery," in *Proceedings of the AAAI Spring Symposium on Detecting and Resolving Errors in Manufacturing Systems, Palo Alto, CA, USA*, vol. 2123, 1994.

[9] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques ;part i: Fault diagnosis with model-based and signal-based approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, June 2015.

[10] B. Vogel-Heuser, S. Rösch, J. Fischer, T. Simon, S. Ulewicz, and J. Folmer, "Fault handling in PLC-based industry 4.0 automated production systems

as a basis for restart and self-configuration and its evaluation," *Journal of Software Engineering and Applications*, vol. 9, no. 1, p. 1, 2016.

[11] P. Bergagård, M. Fabian, P. S, and K. Bengtsson, "Implementing restart in a manufacturing system using restart states."

[12] A. Farooqui, P. Bergagard, P. Falkman, and M. Fabian, "Error handling within highly automated automotive industry: Current practice and research needs," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 9 2016.

[13] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, no. 2, pp. 87 – 106, 1987.

[14] W. van der Aalst, *Process Mining*. Springer Nature, 2016.

[15] T.-C. Chang, *Expert process planning for manufacturing*. Addison-Wesley Longman, 1990.

[16] H. Marri, A. Gunasekaran, and R. Grieve, "Computer-aided process planning: a state of art," *The International Journal of Advanced Manufacturing Technology*, vol. 14, no. 4, pp. 261–268, 1998.

[17] "ABB Robot Studio." [Online]. Available: https://new.abb.com/products/robotics/sv/robotstudio

[18] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[19] K. Bengtsson, B. Lennartson, and C. Yuan, "The origin of operations: Interactions between the product and the manufacturing automation control system," *IFAC Proceedings Volumes*, vol. 42, 2009.

[20] J. E. Hopcroft, R. Motwani, Rotwani, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computability*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[21] C. de la Higuera, *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[22] ——, "A bibliographical study of grammatical inference," *Pattern Recognition*, vol. 38, no. 9, 2005.

[23] M. Bugalho and A. L. Oliveira, "Inference of regular languages using state merging algorithms with search," *Pattern Recogn.*, vol. 38, no. 9, 2005.

[24] R. Parekh and V. Honavar, "Grammar inference, automata induction, and language acquisition," *Handbook of natural language processing*, pp. 727–764, 2000.

[25] J. Oncina and P. Garcia, "Inferring regular languages in polynomial update time," in *Pattern Recognition and Image Analysis*, ser. Series in Machine Perception and Artificial Intelligence, N. P. de la Blanca, A. Sanfeliu, and E. Vidal, Eds., vol. 1. World Scientific, Singapore, 1992, pp. 49–61.

[26] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. Cambridge, MA, USA: MIT Press, 1994.

[27] M. Bugalho and A. L. Oliveira, "Inference of regular languages using state merging algorithms with search," *Pattern Recogn.*, vol. 38, no. 9, Sep. 2005.

[28] E. Gold, "System identification via state characterization," *Automatica*, vol. 8, no. 5, pp. 621 – 636, 1972. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109872900337

[29] W. Wieczorek, *Grammatical Inference: Algorithms, Routines and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[30] A. L. Oliveira and S. Edwards, "Limits of exact algorithms for inference of minimum size finite state machines," in *Proceedings of the 7th International Workshop on Algorithmic Learning Theory*, ser. ALT '96, 1996.

[31] A. L. Oliveira and J. a. P. M. Silva, "Efficient algorithms for the inference of minimum size DFAs," *Mach. Learn.*, vol. 44, no. 1-2, pp. 93–119, Jul. 2001.

[32] K. J. Lang, "Faster algorithms for finding minimal consistent DFAs," Tech. Rep., 1999.

[33] S. M. Lucas and T. J. Reynolds, "Learning DFA: evolution versus evidence driven state merging," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 1, Dec 2003, pp. 351–358.

[34] H. Juillé and J. B. Pollack, "A stochastic search approach to grammar induction," in *Grammatical Inference*, 1998.

[35] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, no. 5, pp. 447 – 474, 1967. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019995867911655

[36] R. E. Schapire, *The Design and Analysis of Efficient Learning Algorithms*. Cambridge, MA, USA: MIT Press, 1992.

[37] M. Isberner, F. Howar, and B. Steffen, "The TTT algorithm: A redundancy-free approach to active automata learning," in *Runtime Verification*, B. Bonakdarpour and S. A. Smolka, Eds. Springer International Publishing, 2014, pp. 307–322.

[38] B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer, 2011, pp. 256–296.

[39] B. Jonsson, *Learning of Automata Models Extended with Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 327–349.

[40] F. Aarts and F. Vaandrager, "Learning I/O automata," in *CONCUR 2010 - Concurrency Theory*, P. Gastin and F. Laroussinie, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 71–85.

[41] M. Isberner, F. Howar, and B. Steffen, "Learning register automata: from languages to program structures," *Machine Learning*, vol. 96, no. 1, pp. 65–98, Jul 2014.

[42] W. Smeenk, J. Moerman, F. Vaandrager, and D. N. Jansen, "Applying automata learning to embedded control software," in *Formal Methods and Software Engineering*. Springer International Publishing, 2015.

[43] S. Goldman and M. Kearns, "On the complexity of teaching," *J. Comput. Syst. Sci.*, vol. 50, 1995.

[44] J. L. Peterson, "Petri nets," *ACM Comput. Surv.*, vol. 9, no. 3, pp. 223–252, Sep. 1977. [Online]. Available: http://doi.acm.org/10.1145/356698.356702

[45] M. Yoeli, "The cascade decomposition of sequential machines," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 4, pp. 587–592, Dec 1961.

[46] R. Dijkman, J. Hofstetter, and J. Koehler, *Business Process Model and Notation*. Springer, 2011.

[47] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, Sept 2004.

[48] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, pp. 1–34, 2006.

[49] C. W. Günther and W. M. Van Der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics," in *International Conference on Business Process Management*. Springer, 2007, pp. 328–343.

[50] W. van der Aalst *et al.*, "Business process mining: An industrial application," *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

[51] W. M. P. van der Aalst, "Business process management: A comprehensive survey," *ISRN Software Engineering*, vol. 2013, pp. 1–37, 2013.

[52] R. S. Mans, M. H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker, *Application of Process Mining in Healthcare - A Case Study in a Dutch Hospital*, ser. Biomedical Engineering Systems and Technologies. Springer Nature, 2008, pp. 425–438.

[53] A. Partington, M. Wynn, S. Suriadi, C. Ouyang, and J. Karnon, "Process mining for clinical processes," *ACM Transactions on Management Information Systems*, vol. 5, no. 4, pp. 1–18, 2015.

[54] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro, "Process mining in healthcare: A literature review," *Journal of Biomedical Informatics*, vol. 61, pp. 224–236, 2016.

[55] H. Yang, M. Park, M. Cho, M. Song, and S. Kim, "A system architecture for manufacturing process analysis based on big data and process mining techniques," in *2014 IEEE International Conference on Big Data (Big Data)*, 10 2014.

[56] P. Viale, C. Frydman, and J. Pinaton, "New methodology for modeling large scale manufacturing process: Using process mining methods and experts' knowledge," in *2011 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, 12 2011.

[57] B. N. Yahya, "The development of manufacturing process analysis: Lesson learned from process mining," *Jurnal Teknik Industri*, vol. 16, no. 2, 2014.

[58] R. Malik, M. Fabian, and K. Åkesson, "Modelling large-scale discrete-event systems using modules, aliases, and extended finite-state automata," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 7000 – 7005, 2011, 18th IFAC World Congress.

[59] R. E. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams," *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 293–318, 1992.