Trace induction for complete manufacturing process model discovery

Koehler, W. & Jing, Y.

Original citation & hyperlink:
Koehler, W & Jing, Y 2020, 'Trace induction for complete manufacturing process model discovery', *International Journal of Advanced Manufacturing Technology*, vol. 110, pp. 29–43.
DOI: 10.1007/s00170-020-05747-3

DOI 10.1007/s00170-020-05747-3
ISSN 0268-3768
ESSN 1433-3015

Publisher: Springer

The final publication is available at Springer via http://dx.doi.org/10.1007/s00170-020-05747-3

Noname manuscript No.
(will be inserted by the editor)

Trace Induction for Complete Manufacturing Process Model Discovery

Wolfgang Koehler · Yanguo Jing

Received: date / Accepted: date

Abstract In industrial automation, there are numerous use cases for a model representing the equipment's sequence of operations. Not only can such a model be used as a debugging aid, allowing users to observe the current state of the machinery at a glance, but it also can serve as a foundation for process improvements. Annotated process models can help pinpoint bottlenecks within the system, and a combination of models obtained from identical machines can be used to create a best-case baseline to which the devices can be optimised. Typically, the models used for such activities have been created during the equipment's design phase, which means that many of the changes made during commissioning, start-up, and production are not reflected. The research domain of Process Mining suggests that an accurate model can be obtained from an activity log. Since most industrial processes are controlled by Programmable Logic Controllers (PLCs), which offer the capability of external access through Open Platform Communications (OPC), such a log can be created automatically. The theory is that Process Mining algorithms can then be used to discover the desired model from the data recorded. Unfortunately, it turns out that this whole procedure is a game of chance. The quality of the discovered models strongly depends on the logged data. Not only can the records be flawed, but also the information needed to discover a complete model might not have been observed at all. To better judge the quality of the discovered models, numerous quality metrics have been proposed, which in the authors' opinion are a good guess at best. The industrial automation domain demands complete models, which cannot be guaranteed by any Process Mining algorithm unless it is known that the log contains all the data needed. This paper shows that for industrial equipment, it is possible to reason through the evaluation of a single recorded case, which case traces are required to guarantee that the discovered model will be complete. In the next step, the authors then introduced the concept of 'trace induction', which takes advantage of the fact that a PLC controls the observed processes. A minor change within the PLC's logic is used to force the process to execute the desired traces on demand. The resulting, minimalistic log can then be analysed by the α_{LC}-algorithm to create a guaranteed complete, model.

Keywords industrial automation · process discovery · flowchart · process mining · process model · trace induction

Wolfgang Koehler
Coventry University, Priory St, Coventry CV1 5FB, UK
Tel.: +44-746-778-5275
E-mail: koehlerw@uni.coventry.ac.uk

Yanguo Jing
Coventry University, Priory St, Coventry CV1 5FB, UK

1 Motivation

Processes, within industrial manufacturing equipment, follow a predetermined sequence which is executed by Programmable Logic Controllers. The sequence of operations is ever repeating for identical parts being manufactured. Due to equipment improvements, often, these processes no longer reflect the design intent, thus rendering any process models, developed during the design phase, useless. There are multiple use cases for an up to date, accurate process model. The most obvious application is the comparison of the 'as is' state with the design intent, often referred to as conformance checking [1]. Beyond that, process models can be used to identify bottlenecks within systems. If the process is

well understood, it is also possible to predict how much time certain activities will take. Based on this information, automated maintenance tasks like dressing the weld tips of robots, can be scheduled for the most convenient time to keep them from impacting production. Finally, complete models can also become a debugging aid by graphically displaying the current state within the process. All of these use cases have in common that they require a complete model to be meaningful. Approximations and metrics required to express confidence in these approximations are not satisfactory. Above perception is reinforced by Browning [2] stating: 'Some people confuse the real process (how work is really done) with the process model or description, which is only an abstract representation of the real process. ... Thus, a prerequisite to process improvement (changing the real process) is increasing the adequacy and accuracy of the process model'.

The remainder of this paper is structured as follows: The related works in section 2 are followed by section 3 in which the hypothesis and methodologies are presented. The experimental evaluation of said methodologies is described in chapter 4 before discussing potential issues in section 5. Finally, the paper is completed with the conclusion in chapter 6.

2 Related Works

Previously there have been different proposal for the use of data obtained from manufacturing equipment. Plant floor systems, as described by Lee [3], can be seen as an initial attempt to monitor manufacturing processes. Predefined parameters are continuously logged for later analysis. Eventually, this concept evolved into cyber-physical systems, as described by Lee et al. [4]. Their goal was to create a digital twin of the physical system, which can then be used for simulations and predictions.

Other researchers proposed the use of machine learning techniques to evaluate the data obtained. Jaber et al. [5], suggest that machine learning can predict the time of equipment failure based on data collected from vibration sensors. Banerjee et al. [6] instead used a similar approach for fault detection.

Another possible approach to gain knowledge from manufacturing equipment data is Business Model Discovery. Van der Aalst et al.[7], who is the main driving force behind this research domain, stated that event logs are a sequential record of events. Typically, there are two types of events that are associated with an activity. They are referred to as the 'start' and 'complete' events. All events belong to a case, which in industrial automation equals one full machine cycle. The sequence of events within a case is termed trace. For some process mining applications, it is possible to reduce the log to representative cases of each trace. In his 2018 paper, Van der Aalst [8] points out that the quality of the discovered Process Model depends on the quality of the log from which it was derived. This argument leads to the conclusion that the log quality should be of utmost importance.

There have been many attempts by Process Mining Practitioners to quantify the quality of the discovered models. Van der Aalst [8] sums these efforts up by explaining that event logs can only capture example cases which makes it difficult to determine the quality of the model relative to the underlying record. He proposed a set of four quality measures. Recall expresses how well the model reflects the behaviour found in the log, while precision gauges how good the model discourages unrelated actions. The third metric is a generalization, which is supposed to make the discovered model also suitable for yet unseen cases, thus trying to prevent overfitting. Because in industrial automation, only one sequence of operations is expected for a single part style, the authors of this paper argue that overfitting is desired as long as all of the meaningful case traces have been observed. The final quality indicator is the simplicity of the model. Van der Aalst, however, warns that the some of the existing approaches to gauge the model quality might be flawed.

Leemans et al. [9] also expresses the need for quality guarantees while acknowledging the need for a nearly complete log in which most of the behaviours possible are present. Besides the model quality, Leemans et al. also focus on the scalability of Process Mining algorithms. Their proposal enables the discovery of models from logs with millions of events from thousands of activities. DeWeerdt et al. [10] take a slightly different approach by evaluating the model quality with the help of artificial generated negative events. This aids in establishing the precision measure previously mentioned because event logs usually do not contain cases that are not allowed. The logic behind creating artificial events is that events that have not been seen at a specific position within a case before must be unrelated and thus, should not be allowed by the process model. De Weerdt et al. point out that this approach allows for a good comparison of process models that have been obtained, based on the same log, by different algorithms. At the same time, they also caution that possible overfitting is not detected. Finally, just like Leemans et al., they want the reader to understand that many of the process discovery algorithms presume the presents of a nearly complete log in order to yield the desired results.

In 2009, Wen et al. [13] proposed the β-algorithm (to be found in ProM as Tsinghua-α-algorithm). It is an extension of the original α-algorithm. Instead of considering only one timestamp per activity, it considers the 'start' and the 'complete' event. The time between the 'start' and the 'complete' event has been termed the 'activity life cycle'. Wen et al. state that the inclusion of the activity's life cycle enables the detection of parallelism. They further point out that most of the established algorithms do not take the life cycle into account, which prevents them from discovering parallel processes. The β-algorithm, just like the α_{LC}-algorithm, requires a complete log for proper discovery.

Burattin [14] acknowledged in 2015, above described benefits of including the activity life cycle. He proposed an extension to the Heuristic Miner, which he named Heuristic++. It is worth noting that this extension is backward compatible, which allows it to handle logs with one or two events per activity. The Heuristic++ Miner, just like the Heuristic algorithm, is based on statistical principals.

Before continuing, the use of the terms 'variant' and 'trace' needs to be clarified. Lee [11] refers in his work to process variants, which he describes as the result of sequence changes applied to an original process model. His work goes into detail about the difficulties in determining the difference between such process variants and case variants within an event log. It is these case variants that Guenther [12] refers to when using the term variant. Van der Aalst [23], on the other hand, calls the sequence, found within a case, 'trace'. In the course of this paper, the authors use the term trace when referring to case variants or traces.

In 2006, Hu et al. [15] attempted to apply Process Mining techniques to manufacturing lines to validate and improve processes. Their solution was also based on Van der Aalst's α-algorithm to which they introduced resources with the help of a second matrix. It was not until 2014 when Son et al. [16] published their paper 'Process Mining For Manufacturing Processes', that Process Mining algorithms once again were applied to manufacturing processes. They aimed to create a model representing all process steps from manufacturing to shipping. Yahya [17], who had a similar research goal, concluded that it was necessary to customise the process model to be discovered based on the analysis' purpose, which means that granularity needs to be chosen accordingly.

The application of Process Model Discovery algorithms to manufacturing equipment data was continued by Farooqui et al. [18] who demonstrated a more detail-oriented approach by implementing additional code into industrial robots. This code allowed them to record program pointer positions, as well as new signal state events which then are used for model discovery. The purpose of the models was to aid decision making and maintenance efforts. Around the same time Nowaczyk et al. [19] were taking a different approach by proposing an unsupervised learning-based framework that evaluates similar systems. A flag is raised if a deviation of one of the systems from the remaining ones is detected.

Evaluating the Process Model Discovery algorithms mentioned above it can be found that some of them are based on matrices into which the relations, found within the log, are parsed. The α-algorithm, for example, uses a footprint matrix to mark direct, parallel or non-existing dependencies between activities. Within the industrial automation domain, the Design Structure Matrix (DSM) introduced by Steward [20] is commonly used for process descriptions. Eppinger et al. [21] acknowledge in their article that DSM is most often used in engineering management because of its compact and scalable system representation. Researchers differentiate between a numerical DSM, where the number of dependencies is counted, or additional parameters are recorded and a binary DSM in which a mark indicates merely the presence of dependencies. A DSM is based on an N-square matrix with identical row and column labels. Such a matrix is for example used by Browning [22] to suggest problem solving approaches which could potentially also be transferred to the industrial automation domain.

Above literature review shows that gauging the quality of the discovered process model is a big concern within the Process Discovery domain. Process Discovery postulates that an increased number of recorded cases, and thus hopefully an increased number of traces, holds the potential to improve the quality of the Process Discovery model. None of the papers reported any attempts to define which traces are needed to discover a highly accuracy model. This research closes said gap by attempting to define rules to pinpoint the needed traces. In addition it offers a methodology that allows for the recording of these traces within a few machine cycles.

3 Hypotheses And Methodologies

3.1 Brief Introduction To The α_{LC}-Algorithm

The α_{LC}-algorithm has been introduced in an yet to be published paper. A summary is provided to obtain a better understanding of its inner workings, and how it relates to this paper. For industrial automation process

model discovery, the authors propose an extension to the rules established for the α-algorithm by Van der Aalst [23]. In the below definitions, '$a >_L b$' should be read as 'a is directly followed by b', while '$b \not>_L a$' means 'b is not directly followed by a'. '$a \rightarrow_L b$' expresses the causal relationship between a and b. Contrary, '$a \#_L b$' shows that there is no relation between a and b.

α_{LC}-algorithm definitions: Let L be an event log over $\mathscr{A}$; i.e. $L \in \mathbb{B}(\mathscr{A}^*)$ Let sub-log $l \in L$ be limited to one-part style and one case per trace. Let a, b, c $\in \mathscr{A}$, t_s a start event and t_c a complete event.

1. $a >_l b$ only if there is a trace $\sigma = \langle t_1, t_2, t_3, ..., t_n \rangle$ and $i \in \{1, ..., n-1\}$ where the timestamp $(\tau(t_i) < \tau(t_{i+1})$ or $\tau(t_i) = \tau(t_{i+1})$ as long as not $(t_i = t_c$ and $t_{i+1} = t_s)$ and $k \in \{i+1, ..., n\}$ such that $\sigma \in l$ and $t_i = a_c$ and $t_k = b_s$ and there is no j such that $i < j < k$ and $t_j = c_c$;
2. $a \rightarrow_l b$ if and only if $a_c >_l b_s$ and $b_s \not>_l a_c$;
3. $a \#_l b$ if and only if $a_c \not>_l b_s$ and $b_c \not>_l a_s$;
4. $a \|_l b$ if and only if $a_c >_l b_s$ and $b_s >_l a_c$;

Although α_{LC}-definition #3 & #4 holds, they are of no importance for the proposed algorithm. Gantt charts are best suited to explain the two other definitions more in detail. According to Sommer [24] they are typically used in the automation domain to visualise the sequence of operation for a machine. The columns represent time increments while bars, displayed in the rows, stand for the duration of an activity where the beginning of the bar marks its 'start' time and the end the 'complete' time. For completeness, it needs to be mentioned that Gantt charts often also include links to express the dependencies between the different activities. It is precisely those dependencies that Process Discovery aims to discover. Therefore, the Gantt charts shown within this paper are drawn without the links: α_{LC}-definition # 1 stipulates that $a >_l b$ if the timestamp $\tau_s(b) \geq \tau_c(a)$. Also, there cannot be a complete-timestamp of another activity in between unless $\tau_s(b)$ is the first start-timestamp following $\tau_c(a)$. Figure 1 shows that the activities b and c, contrary to activity d, fulfill these requirements concerning activity a. If activity c, is recorded a little earlier, as shown in figure 2, then the α_{LC}-definition #1 leads to the conclusion that $a, c >_l b, d$. α_{LC}-definition #2 excludes a dependency found according to the α_{LC}-definition #1 if any case is observed where the timestamp $\tau_c(a) > \tau_s(b)$. Possible such scenarios are shown in figure 3 in red for activity a in relation to activity b.

In the α_{LC}-algorithm, the two definitions are implemented with the help of two design structure matrices, which capture the relations, as well as the opposing relations of all activities. Matrix operations, namely inversion, transposition and logical operations are then used to overlay the two matrices to reveal a resulting DSM with all dependencies found. This final matrix can then be converted into a flowchart for easier viewing. The advantage of the α_{LC}-algorithm over the other Process Mining algorithms is that it is not only less complicated but it also can derive a complete Process Model based on the minimum number of case traces available.

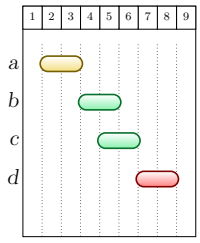


Fig. 1: $a \rightarrow_l b, c$; $c \rightarrow_l d$

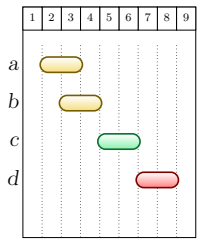


Fig. 2: $a, c \rightarrow_l b, d$

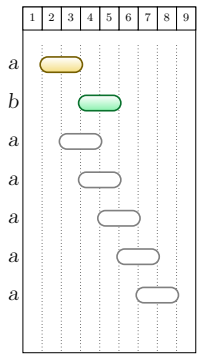


Fig. 3: a / b dependencies

3.2 dependencies matrices

As described in the related works, dependencies matrices are a tool often used in the Process Mining domain. They consist of a square matrix with equal column and row names. For this research, the column/row names are the names of the activities within a case sorted by their start timestamps. An example can be seen in figure 4. Here the rows are related to the start timestamp while the columns represent the complete timestamp. The diagonal of this matrix represent instances where an activity depends on itself. These cases are referred to as short loops in the Process Mining domain and do not exist for industrial assembly processes. A '1' marks a discovered dependency while a '0' marks a not existing dependency. The lower triangle of the matrix is used to mark situations where a start timestamp, according to the definitions of section 3.1 directly follows the complete timestamp of its previous activity ($a_c >_L b_s$). The upper triangle is not used because the proposed algorithm considers life cycle information which cannot be represented in a single dependencies matrix. The opposing dependencies matrix, representing complete timestamps (rows) following start timestamps (columns) ($a_s >_L b_c$), uses the upper triangle only. Inverting and transposing the opposing matrix onto the initial matrix using a logical 'AND' function creates the final matrix from which a flowchart can be constructed.

Fig. 4: Sample Dependencies Matrix

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	1	1	0	0	1	0	1	0	0	0
	initiate R02 (6)	1	1	1	1	0	0	1	0	0	0	0
	R01 weld / clear (7)	0	1	1	1	1	0	0	1	1	1	0
	R02 weld / reposition (8)	0	1	1	1	1	1	0	0	0	0	0
	open clamps 3 (9)	0	0	0	1	0	1	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	1	0	1	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

3.3 Traces And Their Usage

Traces found in business processes are primarily caused by either faulty/incomplete logs or human intervention. Traces in industrial automation processes, on the other hand, are also triggered by different part styles and parallel processes. Since these concurrent processes are often independent of each other, it has to be expected that the position of their activities within a log will vary. Also, the duration of the activities can vary, which mostly is caused by fault conditions. For example, a clamp might close slower, because of weld slack collecting on it, which in turn will lead to the depending activities to show up later within the log.

A casual observer typically concludes that the sequence of activities within a case clearly describes the process. This observation, however, is a misconception. Table 1 shows an example log representing the sequence of activities for a single part style. Applying the α_{LC}-algorithm to one case only will yield the flowchart shown in figure 5, which differs from the complete model shown in figure 6. The reason for these discrepancies is the lack of dependencies which lets the onlooker infer that the start of a new activity depends on all of the recently completed activities. This confusion is best seen in the associated Gantt chart figure 7.

Table 1: Log Example Style 1

caseID	style	motionID	startTime	completeTime
1	1	1	00:00:00	00:00:04
1	1	2	00:00:04	00:00:05
1	1	4	00:00:04	00:00:05
1	1	5	00:00:04	00:00:06
1	1	7	00:00:04	00:00:05
1	1	8	00:00:05	00:00:12
1	1	6	00:00:06	00:00:10
1	1	9	00:00:13	00:00:14
1	1	10	00:00:14	00:00:17
1	1	11	00:00:18	00:00:19
1	1	13	00:00:19	00:00:23

The discovery of dependencies is only possible with the help of additional traces being recorded. Figure 8, for example, is a case of an equipment where activity $a, b \rightarrow_l c$ and activity $d \rightarrow_l e$. Looking at this one case only will lead to the conclusion that activity $a, d >_l e$ and $b, e >_l c$. Only when the second case (shown in figure 9) is considered it will become apparent that $a \rightarrow_l c$ and that $a \#_l e$. Therefore traces aid the model discovery by breaking up falsely perceived links and revealing previously unseen dependencies.

3.4 Hypothesis

During previous experiments, it was found that the α_{LC}-algorithm is capable of discovering a highly accurate process model from a log comprised of just three,

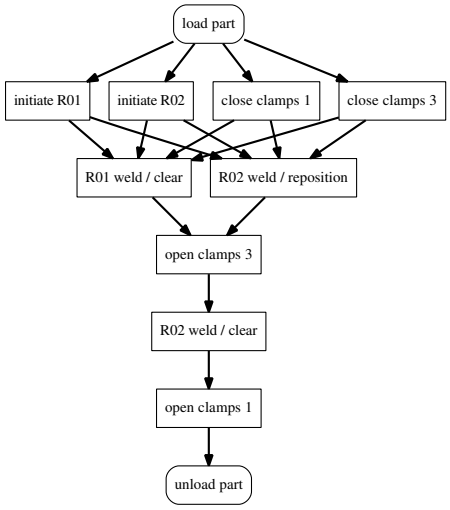


Fig. 5: The One Trace Model

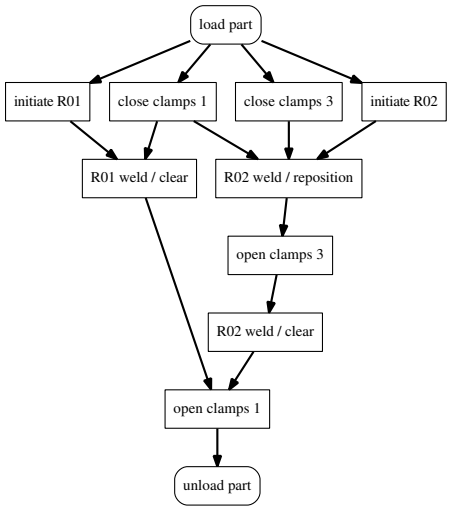


Fig. 6: The Complete One Style Model

purposely selected cases. This led to the following hypothesis: **An algorithm should be able to determine which minimum traces need to be present to allow for a highly accurate model discovery.** If this hypothesis holds then it would justify a second hypothesis that states that **it must be possible to force a process, controlled by a PLC, to execute the traces deemed necessary in the previous step.**

3.5 Required Traces

Taking the Gantt chart figure 7 as an example, the potential causes of confusion, due to the lack of links, can be pinpointed. The 'start' and 'complete' events of activities 2, 4, 5 and 6 are identical. Also, the 'start' events of activities 7 and 8 follow directly after that. Therefore,

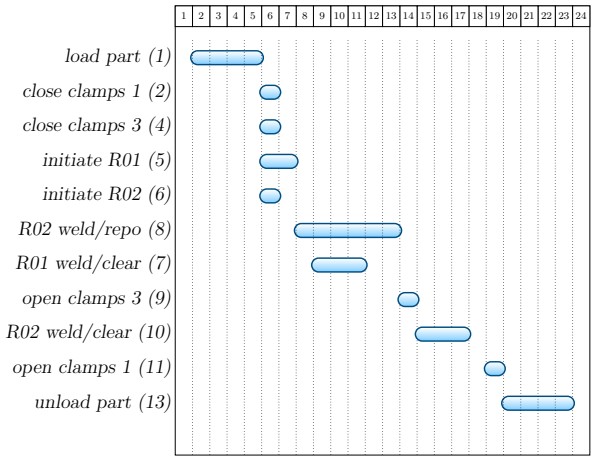


Fig. 7: Random Case

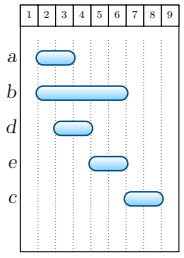


Fig. 8: Example Case 1

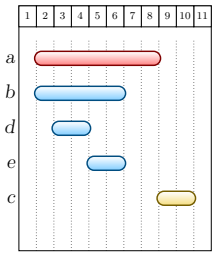


Fig. 9: Example Case 2

it is not possible to determine which activities depend on each other. This confusion can only be solved if one of the four previously mentioned activities completes later than the other three. Only then it will become clear which other activities depend on it. All dependencies, therefore, will be known if such a scenario has been observed for all four activities. The same reasoning can be applied to activities 7 and 8 since they are executed in parallel. The following theorems have been developed to limit the traces to be recorded to a minimum:

Theorem 1 *Contrary to business processes the edge in a flowchart representing an industrial automation process can be considered instantaneous, which means that an activity starts immediately after the last preceding activity has been completed. If there is only a single preceding activity the dependency can be considered as correct and does not need further checking.*

Theorem 2 *If activity 'b' directly follows activity 'a'* ($a >_l b$) *and activity 'c' directly follows activity 'b'* ($b >_l c$) *then there is no explicit dependency between activity 'a' and 'c'* ($a \#_l c$). *This also holds for longer dependency chains.*

Theorem 3 *Definition two of the α_{LC}-algorithm (see section 3.1) holds also which states that b_s cannot depend on a_c if a_c has been observed later than b_s as well.*

The dependencies matrices introduced in chapter 3.2 also can be helpful to determine which minimum traces are required to discover a highly accurate process model. Parsing the random case, depicted in figure 7 will yield the dependencies matrix shown in figure 10. In this initial matrix all direct following dependencies are marked by an 'X' because none of the above definitions have been applied and therefore the dependencies have not been confirmed.

Applying theorem 1 allows the values of the cells highlighted in green in figure 11 to be changed to '1' because there is only one preceding activity. According to theorem 2 all the values of the red shaded cells can be set to '0' since they would be part of chained dependencies. Finally, based on theorem 3 the cells marked in blue can be set to '0' as well because parallel activities cannot depend on each other.

Above example shows that the evaluation of a single trace already lead to the discovery of 50% of the existing or non-existing dependencies. The cells still marked with an 'X' and the blank cells are yet to be evaluated. Now it is possible to repeat this process by interactively adding, purposely chosen traces that can explain the remaining dependencies. The presence or absence of these traces within the log allows for a better judgement regarding the quality of the resulting process model.

3.6 Trace Induction

In section 3.5, the criteria for a minimum set of traces needed for a complete model discovery, have been laid out. Fortunately, the automation domain offers opportunity to influence the process. This paper, therefore, proposes the concept of 'trace induction'. In many of today's PLC programs, physical sensor inputs are mapped to meaningful tags. Instead of hoping for weld slack, that slows down the clamp, to build up, an additional condition can be added to that mapping, which allows the sensor tag to be delayed (see figure 12). Initiating that delay, manually or through OPC, will result in a new trace being recorded.

The rules as stated in section 3.5, first can be used to determine which of the activities are suspect. Once identified, the corresponding sensor tags can be delayed one by one, until all unrelated activities are completed. This process needs to be repeated for every style. The outcome is a record of all traces, which are required to describe the process.

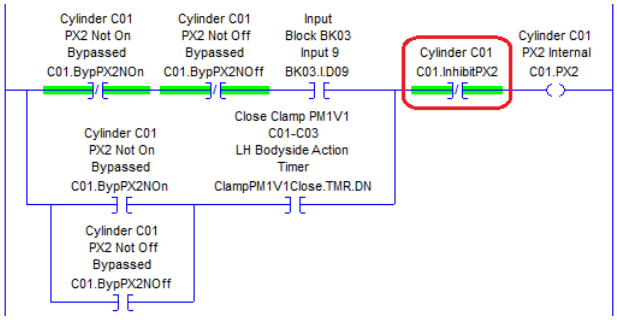


Fig. 12: Required Logic Modification

If the inclusion of this additional tag becomes standard, there will be no extra time required for its implementation into new production equipment. Modifying legacy equipment will require only little effort, as long as its control logic adheres to some tag standardisation.

The main advantage of 'trace induction', is that the process of model discovery no longer depends on chance. Instead, the required traces are induced on demand. Since this can be done at a convenient time, it has minimum impact on production. At the same time, external processes impacting the cases recorded, can be kept to a minimum, thus increasing the quality of the log. Because the α_{LC}-algorithm can discover the desired model with a minimum of traces available, the time for data logging can be reduced from weeks, or even months to just a few hours. Besides, the processing time for such a minimalistic log is decreased dramatically as well.

4 Experiments And Results

4.1 Viability Study

As a first step, it had to be proven that it is possible to delay the sensor inputs of the different motions, as suggested in section 3.6. This delay was achieved by implementing the change shown in figure 12 within a select number of devices in a small, standalone robotic cell. At first, the newly created tags were toggled one by one manually, and the resulting equipment behaviour was recorded. Reviewing the log proved that the concept worked as expected. Afterward, the same test was repeated, controlling the tags through an OPC connection. Using OPC required that the tags are enabled for remote writing, which is the standard setting for the RSLogix controller available for the test. The resulting log again, matched expectations.

Fig. 10: Initial Depend. Matrix

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	X										
	close clamps 3 (4)	X										
	initiate R01 (5)	X										
	initiate R02 (6)	X										
	R01 weld / clear (7)		X	X	X	X						
	R02 weld / reposition (8)		X	X	X	X						
	open clamps 3 (9)							X				
	R02 weld / clear (10)								X			
	open clamps 1 (11)									X		
	unload part (13)										X	

Fig. 11: Depend. Matrix 1st Iteration

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	X	X	X	X						
	R02 weld / reposition (8)	0	X	X	X	X	0					
	open clamps 3 (9)	0						1				
	R02 weld / clear (10)	0						0	1			
	open clamps 1 (11)	0						0	0	1		
	unload part (13)	0						0	0	0	1	

4.2 Artificial Log

For the remaining experiments, it was decided to use an artificial log instead. Such a record allowed for better control over the log, and also enabled the introduction of extreme imbalances, which might cause the process to fail. A random case for that artificial log is shown in table 1. This log has been derived from the process shown in the flowchart 6. For better understanding, the Gantt chart of that case is shown in figure 7.

4.3 Trace Induction

Interactive trace induction is proposed to solve the problem of unknown dependencies. As previously described, this is achieved by artificially delaying the 'complete' events, of questionable activities, one at a time. This forces all of the dependent, downstream activities to be isolated, and thus, reveal their actual dependencies. Note: The below example depicts, for better understanding, only one dependencies matrix for each of the traces to be evaluated. To discover the parallel activities according to theorem 3 (marked in blue) with a software algorithm it would be necessary to work with a second dependencies matrix in which the opposing dependencies are recorded.

Gantt chart figure 7 represents a random case example. It is provided without any links because the dependencies between the activities are not yet known. The data is parsed into a decision matrix, marking potential dependencies with 'x', as shown in figure 13. Applying theorem 1 allows the values of the cells highlighted in green in figure 14 to be changed to '1' because there is only one preceding activity. According to theorem 2 all the values of the red shaded cells can be set to '0' since they would be part of chained dependencies. Finally, based on theorem 3 the cells marked in blue can be set to '0' as well because parallel activities cannot depend on each other.

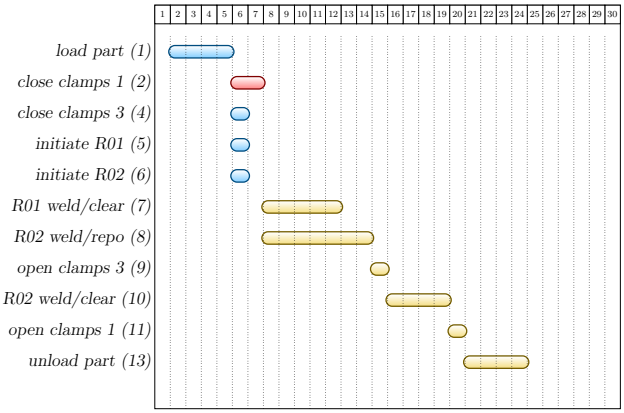


Fig. 15: Activity 2 Delayed

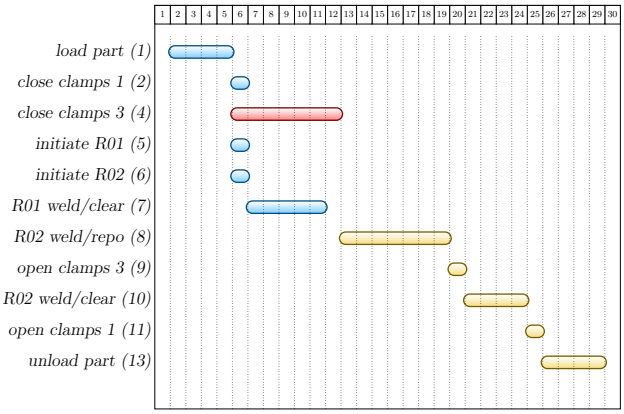


Fig. 16: Activity 4 Delayed

Fig. 13: Initial Depend. Matrix

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	X										
	close clamps 3 (4)	X										
	initiate R01 (5)	X										
	initiate R02 (6)	X										
	R01 weld / clear (7)		X	X	X	X						
	R02 weld / reposition (8)		X	X	X	X						
	open clamps 3 (9)							X				
	R02 weld / clear (10)								X			
	open clamps 1 (11)									X		
	unload part (13)										X	

Fig. 14: Depend. Matrix 1st Iteration

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	X	X	X	X						
	R02 weld / reposition (8)	0	X	X	X	X						
	open clamps 3 (9)	0						1				
	R02 weld / clear (10)	0						0	1			
	open clamps 1 (11)	0						0	0	1		
	unload part (13)	0						0	0	0	1	

The unknown dependencies are addressed one by one. The first question is if 'R01 weld/clear (7)' depends on 'close clamps 1 (2)'. To determine that activity 2 needs to be artificially delayed and the corresponding machine cycle recorded. For the example this would result in the trace shown in figure 15. This Gantt chart clearly shows that not only activity 7 but also activity 8 directly follow activity 2 ($2 > 7, 8$) as postulated in theorem 1. These two dependencies are therefore marked with '1' (highlighted in green) within the decision matrix figure 17. Now since a parallelism between the events 7 and 8 has been established theorem 3 allows for the blue marked cell to be set to 0. Based on theorem 2 it can also be concluded that the activities $9, 10, 11, 13\#2$ (do not depend on 2). Therefore the value '0' can be assigned (marked in red).

Looking at the decision matrix 17 the next question is if activity 7 depends on activity 3. Delaying activity 3 will produce the Gantt chart plotted in figure 16. It shows activity 4 and 7 in parallel which, according to theorem 3 means that there is no dependency. Consequently the value of the corresponding cell can be set to '0' in figure 18 (shown in blue). Activity 8 however directly follows activity 4 and therefore that value can be set to '1'. Based on theorem 2 the activities $9, 10, 11, 13\#3$ and their cells are set to '0' as highlighted in red.

Continuing on, the relation between activity 7 and activity 5 is questioned. Delaying activity 5 will result in the Gantt chart 19. Based on theorem 1 it can be concluded that a dependency exists between activity 7 and activity 5 as well as between activity 11 and activity 7 (both marked in green in figure 21) while the theorems 2 and 3 exclude the dependencies shown in red and blue.

Now the dependency between activity 7 and activity 6 needs to be evaluated by delaying activity 6 as shown in Gantt chart 20. As marked in the decision matrix 22, theorem 1 determines that there is a relation between

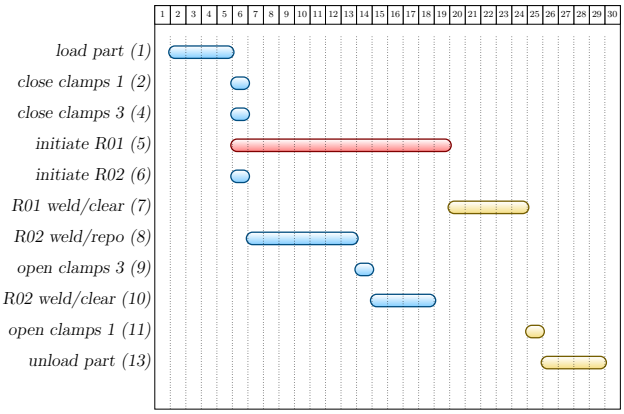


Fig. 19: Activity 5 Delayed

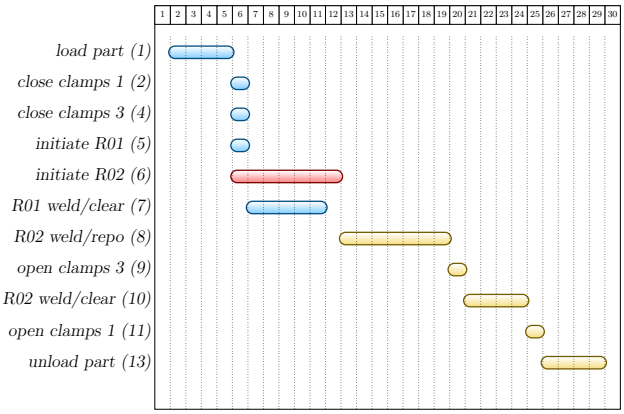
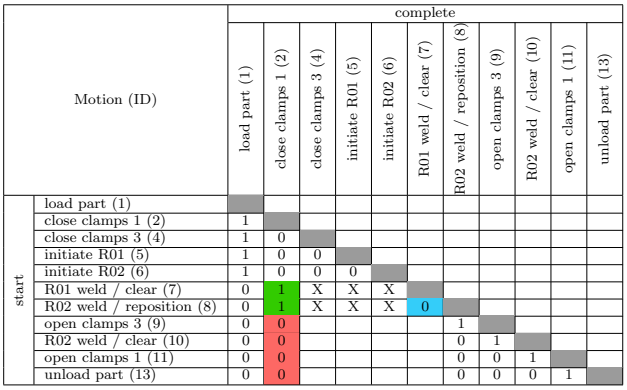


Fig. 20: Activity 6 Delayed

the activities 8 and 6 (marked in green) while the remaining theorems allow for the conclusion that there aren't any dependencies for the cells highlighted in red and blue.

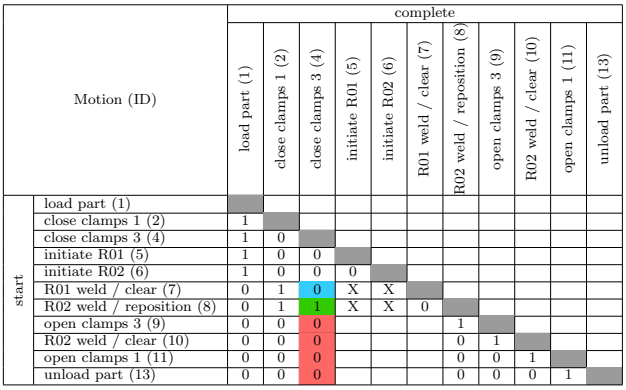
At this point all dependencies and non-dependencies are clear and no further traces are required. As a nice

Fig. 17: Depend. Matrix 2nd Iteration



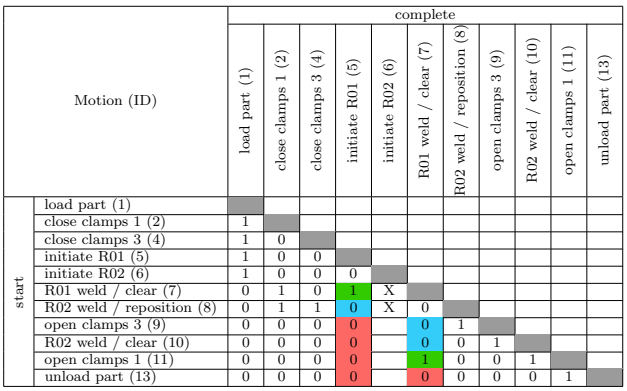
Motion (ID)	load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
load part (1)											
close clamps 1 (2)	1										
close clamps 3 (4)	1	0									
initiate R01 (5)	1	0	0								
initiate R02 (6)	1	0	0	0							
R01 weld / clear (7)	0	1	X	X	X						
R02 weld / reposition (8)	0	1	X	X	X	0					
open clamps 3 (9)	0	0					1				
R02 weld / clear (10)	0	0					0	1			
open clamps 1 (11)	0	0					0	0	1		
unload part (13)	0	0					0	0	0	1	

Fig. 18: Depend. Matrix 3rd Iteration



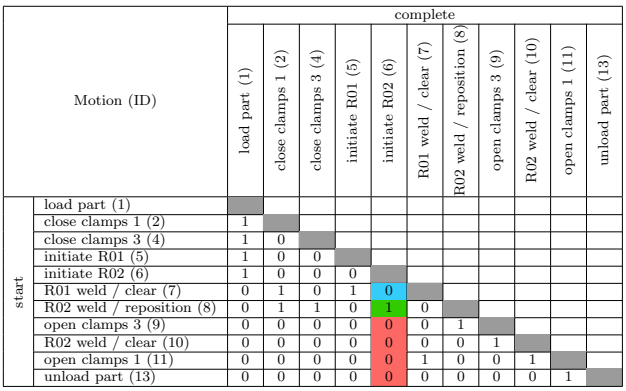
Motion (ID)	load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
load part (1)											
close clamps 1 (2)	1										
close clamps 3 (4)	1	0									
initiate R01 (5)	1	0	0								
initiate R02 (6)	1	0	0	0							
R01 weld / clear (7)	0	1	0	X	X						
R02 weld / reposition (8)	0	1	1	X	X	0					
open clamps 3 (9)	0	0	0				1				
R02 weld / clear (10)	0	0	0				0	1			
open clamps 1 (11)	0	0	0				0	0	1		
unload part (13)	0	0	0				0	0	0	1	

Fig. 21: Depend. Matrix 4th Iteration



Motion (ID)	load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
load part (1)											
close clamps 1 (2)	1										
close clamps 3 (4)	1	0									
initiate R01 (5)	1	0	0								
initiate R02 (6)	1	0	0	0							
R01 weld / clear (7)	0	1	0	1	X						
R02 weld / reposition (8)	0	1	1	0	X	0					
open clamps 3 (9)	0	0	0	0		0	1				
R02 weld / clear (10)	0	0	0	0		0	0	1			
open clamps 1 (11)	0	0	0	0		1	0	0	1		
unload part (13)	0	0	0	0		0	0	0	0	1	

Fig. 22: Depend. Matrix 5th Iteration



Motion (ID)	load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
load part (1)											
close clamps 1 (2)	1										
close clamps 3 (4)	1	0									
initiate R01 (5)	1	0	0								
initiate R02 (6)	1	0	0	0							
R01 weld / clear (7)	0	1	0	1	0						
R02 weld / reposition (8)	0	1	1	0	1	0					
open clamps 3 (9)	0	0	0	0	0	0	1				
R02 weld / clear (10)	0	0	0	0	0	0	0	1			
open clamps 1 (11)	0	0	0	0	0	1	0	0	1		
unload part (13)	0	0	0	0	0	0	0	0	0	1	

side effect it needs to be mentioned that the final dependencies matrix 22 also equals the result matrix of the α_{LC} algorithm applied to the above recorded traces. This in turn means that the desired process model, in form of a flow chart, can be generated based on matrix 22 without any further mining effort.

To prove this the α_{LC}-algorithm is applied to the four induced cases, generating the dependencies matrix shown in figure 23.

4.4 Alternative Mining Approaches

To prevent being partial to the α_{LC}-algorithm, the above-described process 4.3 for 'trace induction' and model discovery was repeated using the β-algorithm, as well as the Heuristic++ Miner. These two algorithms were chosen because during previous research, they proved to be the most competitive when compared to the α_{LC}-algorithm. Unfortunately, both algorithm implementations do not allow for the extraction of underlying matrices. Instead, the DSM, for the randomly recorded case, had to be inferred from the graphical model they produced which did not match the DSM provided by the α_{LC}-algorithm, and therefore, did not allow for the natural selection of the activities that need further interrogation.

The final step of creating a process model from the four induced process traces, was only completed satisfactorily by the β-algorithm. The model created by the Heuristic++, on the other hand, did not match the expected, complete model. These tests allowed for the conclusion that the α_{LC}-algorithm is the best choice for the proposed 'trace induction' concept.

5 Discussion

It is often presumed that the quality of the discovered model is likely to increase with the number of cases recorded for that process. Based on the findings of this paper, it can be concluded that the quality of the discovered model depends on the necessary process traces being recorded, and a log spanning a more extended period increases the chance of doing so. Unfortunately, the reasons for process traces within business processes seems to differ from industrial automation processes. Therefore, the concept of 'trace induction' might only

Fig. 23: α_{LC} Result: Interactive Trace Induction Validation

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	0	0	0	0	0	0	0	0	0	0
	initiate R02 (6)	1	0	0	0	0	0	0	0	0	0	0
	R01 weld / clear (7)	0	1	0	1	0	0	0	0	0	0	0
	R02 weld / reposition (8)	0	1	1	0	1	0	0	0	0	0	0
	open clamps 3 (9)	0	0	0	0	0	0	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	0	0	0	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

be partially applicable for the discovery of concurrent processes within the business domain.

When applying 'trace induction' to real-life processes, one should be aware that parallel processes are not the only reason leading to the manifestation of multiple process traces. Another, often more dominant cause, is that the logs are flawed due to wrongly recorded or missing events. Therefore, while executing 'trace induction', it is still necessary to validate the completeness of the log obtained. Such verification could be as simple as comparing the number of events recorded for each of the cases. Previous research, with real-life logs [26], showed that the majority of cases are indeed flawless. Therefore, it can be concluded that the number of events found in the majority of cases is correct. Alternatively, the 'trace induction' could be executed multiple times to ensure that the log is accurate. Deviating cases could be discarded.

This paper purposely does not gauge the discovery results with the help of any of the established quality metrics. The reason is that the α_{LC}-algorithm is not based on any statistical methods and also does not strive for generalisation. Therefore, the discovered model fits the log data available. Some practitioners might term that as over-fitting, but the authors argue that within the industrial automation realm, this is preferred over generalisation. This view is supported by Schimm [27] who states: 'In many cases, the benefit of workflow mining depends on the exactness of the mined models'. Also, confidence metrics can be misleading. Looking at example 15,16,19,20, it can be seen that a complete model also can be discovered with just three of the four traces, because the first trace (figure 15) does not lead to any information gain. The hypotheses, however, state for this process that four traces need to be discovered to obtain a complete log. If the record does not contain the non value-added trace described above, confidence would be lowered by up to 75%, although the discovered model will still be complete.

'Trace induction' has at least two easily overlooked side effects. When delaying the completion of an activity, one inadvertently triggers the watchdog mechanism that should be present in any PLC logic. This behaviour can be seen as a positive benefit because it enables a simple, automated procedure for testing the PLC's fault logic. At the same time, 'trace induction' might force the equipment into a status that will never occur during regular operation. Therefore, its implications might not have been considered by the programmer. In a worst-case scenario, this could lead to a collision within the machine. It is recommended that the responsible controls personnel is engaged before executing 'trace induction' to minimise risk.

6 Conclusion

Over the course of this paper, it has been shown that there is a need for a complete process model within the industrial automation domain. A Process Mining based model discovery, from an activity log, however, is strongly dependent upon the observed process traces. This paper introduces criteria that allows for gauging whether the data contained within an activity log includes all of the traces necessary to discover a guaranteed complete model. This thought then is taken to the next step by introducing the concept of 'trace induction'. A slight modification is made to the PLC code, controlling the process, that allows delaying the 'complete' event of a chosen activity. This modification forces all of the depending activities to be delayed as well, while the independent activities are executed as usual. The resulting log then allows for a clear distinction of the dependencies. Experiments with an artificial record proved the viability of the concept through the

discovery of complete process models with the help of the α_{LC}-algorithm.

Future research should focus on determining if it is possible to evaluate business process logs similarly. Such an approach potentially could lead to new, possibly more meaningful metrics, although it is still questionable if a confidence rating of less than 100% is helpful.

Acknowledgements The authors would like to thank Opel Automobile GmbH and Vauxhall for sponsoring this research, and providing the equipment and workforce necessary for the evaluation trials.

References

1. A.A. Kalenkova, W.M. P. Van Der Aalst,I.A. Lomazova,V.A. Rubin (2017), "Process Mining Using Bpmn: Relating Event Logs And Process Models.", Software & Systems Modeling, Vol. 16, pp. 1019–1048, , DOI. 10.1007/s10270-015-0502-0
2. T.R. Browning (2002), "Process Integration Using The Design Structure Matrix", System Engineering, Vol. 5, pp. 180-193, DOI. 10.1002/sys.10023
3. J.Lee (2003), "E-Manufacturing—Fundamental, Tools, And Transformation. Robotics And Computer-Integrated Manufacturing", Robotics And Computer-Integrated Manufacturing , Vol. 19, pp. 501-507, DOI. 10.1016/S0736-5845(03)00060-7
4. Q. Qi, F. Tao, "Digital Twin And Big Data Towards Smart Manufacturing And Industry 4.0: 360 Degree Comparison", IEEE Access, Vol.6, pp.3585 - 3593, 2018, DOI.10.1109/ACCESS.2018.2793265
5. A.A. Jaber, R, Bicker (2014), "The State Of The Art In Research Into The Condition Monitoring Of Industrial Machinery", International Journal Of Current Engineering And Technology, Vol. 4, pp. 1986-2001,
6. T.P. Banerjee, S.Das (2012), "Multi-Sensor Data Fusion Using Support Vector Machine For Motor Fault Detection", Information Sciences, Vol. 217, pp. 96-107, DOI. 10.1016/j.ins.2012.06.016
7. W. Van Der Aalst, A. Adriansyah, A.K.A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, A. Burattin (2011), "Process Mining Manifesto", Business Process Management Workshops. Bpm 2011. Lecture Notes In Business Information Processing, Vol. 99, pp. 169-194, DOI. 10.1007/978-3-642-28108-2_19
8. W.M.P Van Der Aalst (2018), "Relating Process Models And Event Logs-21 Conformance Propositions", in Ceur Workshop Proceedings, Aachen, Germany, pp. 56-74
9. S.J. J. Leemans, D. Fahland, W.M.P. Van Der Aalst (2018), "Scalable Process Discovery And Conformance Checking", Software & Systems Modeling, Vol. 17, pp. 599–631, DOI. 10.1007/s10270-016-0545-x
10. J.De Weerdt, M. De Backer, J.Vanthienen,B.Baesens (2011), "A Robust F-Measure For Evaluating Discovered Process Models", in 2011 Ieee Symposium On Computational Intelligence And Data Mining (Cidm), Paris, France, pp. 148-155
11. C.Li, M. Reichert, A. Wombacher (2008), "Mining Process Variants: Goals And Issues", in 2008 Ieee International Conference On Services Computing, Honolulu, USA, pp. 573-576
12. C.W. Günther, A. Rozinat (2012), "Disco: Discover Your Processes", in Demonstration Track Of The 10Th International Conference On Business Process Management (Bpm 2012), Tallin, Estonia, pp. 40-44
13. L. Wen, J.Wang, W.M.P. Van Der Aalst, B. Huang (2009), "A Novel Approach For Process Mining Based On Event Types", Journal Of Intelligent Information Systems, Vol. 32, pp. 163-190, DOI. 10.1007/s10844-007-0052-1
14. A. Burattin (2015), "Heuristics Miner For Time Interval. In Process Mining Techniques In Business Environments ", Process Mining Techniques In Business Environments , pp. 85-95, DOI. 10.1007/978-3-319-17482-2_11
15. H.Hu, Z. Li, A.Wang (2006), "Mining Of Flexible Manufacturing System Using Work Event Logs And Petri Nets", in International Conference On Advanced Data Mining And Applications Adma 2006, Xian, China, pp. 360-387
16. S. Son, B.N. Yahya, M. Song, S.Choi, J. Hyeon, B. Lee, Y.Jang, N.Sung (2014), "Process Mining For Manufacturing Process Analysis: A Case Study", in Proceeding Of 2Nd Asia Pacific Conference On Business Process Management, Brisbane, Australia,
17. B.N. Yahya (2014), "The Development Of Manufacturing Process Analysis: Lesson Learned From Process Mining", Jurnal Teknik Industri, Vol. 16, pp. 97-107, DOI. 10.9744/jti.16.2.95-106
18. A.Farooqui, K.Bengtsson, P.Falkman,M.Fabian (2018), "From Factory Floor To Process Models: A Data Gathering Approach To Generate, Transform, And Visualize Manufacturing Processes", Cirp Journal Of Manufacturing Science And Technology, Vol. 24, pp. 6-16, DOI. 10.1016/j.cirpj.2018.12.002
19. S.Nowaczyk, A. Sant'Anna, E.Calikus, Y. Fan (2018), "Monitoring Equipment Operation Through Model And Event Discovery", in Intelligent Data Engineering And Automated Learning – Ideal 2019, Madrid, Spain, pp. 41-53
20. D.S. Steward (1981), "The Design Structure System: A Method For Managing The Design Of Complex Systems", Ieee Transactions On Engineering Management , Vol. EM-28, pp. 71-74, DOI. 10.1109/TEM.1981.6448589
21. S.D.Eppinger, T.R. Browning , "Design Structure Matrix Methods And Applications (2012)", in Design Structure Matrix Methods And Applications, Cambridge, UK, Mit Press,
22. T.R. Browning (2001), "Applying The Design Structure Matrix To System Decomposition And Integration Problems: A Review And New Directions", Ieee Transactions On Engineering Management, Vol. 48, pp. 292-306, DOI. 10.1109/17.946528
23. W. Van Der Aalst, "Process Mining: Data Science In Action (2016)", in Process Mining: Data Science In Action, Edition 1, Berlin, Heidelberg, Springer, pp. 3-23
24. M. Sommer, "Zeitliche Darstellung Und Modellierung Von Prozessen Mit Hilfe Von Gantt-Diagrammen ", M.S. thesis, University Of Ulm, Ulm, Germany, 2012
25. A.Bolt, W. M. P. Van Der Aalst, M. De Leoni (2017), "Finding Process Variants In Event Logs", in Otm 2017: On The Move To Meaningful Internet Systems. Otm 2017 Conferences, Rhodes, Greece, pp. 45-52
26. W. Koehler, Y. Jing (2018), "Automated, Nomenclature Based Data Point Selection For Industrial Event Log Generation", in Intelligent Data Engineering And Automated Learning – Ideal 2018, Cham, Switzerland, pp. 31-40
27. G. Shimm (2004), "Mining Exact Models Of Concurrent Workflows", Computers In Industry, Vol. 53, pp. 265-281, DOI. 10.1016/j.compind.2003.10.003