On Global Completeness of Event Logs

Hedong Yang[1], Arthur HM ter Hofstede[2,3]**, B.F. van Dongen[3], Moe T. Wynn[2], and Jianmin Wang[1]

[1] Tsinghua University, Beijing, China, 10084
yanghd06@mails.tsinghua.edu.cn,jimwang@tsinghua.edu.cn
[2] Queensland University of Technology, Brisbane, Australia
{a.terhofstede,m.wynn}@qut.edu.au
[3] Eindhoven University of Technology, Eindhoven, The Netherlands
b.f.v.dongen@tue.nl

Abstract. The field of process mining provides a collection of techniques and tools that aim to support the extraction of information out of event logs. This information may provide businesses insight into actual execution and performance of their business processes and may help identify ways of improving these processes. While the quality of the results of the application of mining algorithms depends on the degree of completeness of the event log (some even assuming that the logs are complete), there is currently no approach to estimate completeness of an event log in an objective manner. In this paper such an approach is proposed in the context of mining control-flow dependencies. An experimental evaluation of the approach is also presented.

1 Introduction

Business Process Management (BPM) is concerned, among others, with providing support for (re)design, deployment, and analysis of business processes. Process mining can be seen as a subfield of BPM providing a powerful collection of techniques for deriving information from event logs. Two classical applications of process mining are *model discovery*, where the objective is to derive a process model from an event log [11], and *conformance checking*, where actual process behaviour, captured in an event log, is compared to expected process behaviour, captured in a process model. Over time the field of process mining has evolved, and broadly speaking, application of the techniques developed may provide valuable insight into operational performance of business processes and may help identify opportunities for their improvement. An overview of various mining algorithms can be found in [4,2,9]. An open source platform, ProM, exists which provides support for many of these algorithms and which has been used in industrial applications [8].

Although many problems in the field of process mining have been solved in a satisfactory manner, unsolved problems remain and these present impediments

** Senior Visiting Scholar of Tsinghua University.

to advancement of the field [4]. One of these problems, which has received very little attention in the literature, deals with completeness of the event logs.

Generally speaking, the quality of results mined from a given event log depends on the algorithm used and on the amount of information present in this log. Mining algorithms can be classified into several categories as argued in [9]. The first category consists of algorithms which assume the event log to be *globally complete*, i.e. all possible behaviour should have been recorded in the log. The second category contains those algorithms that make specific assumptions on completeness but do not require all possible behaviour to be captured, i.e. they require the log to be *locally complete*. The third category consists of algorithms where the quality of the result correlates with the degree of completeness of the event log, but no assumptions on completeness are made at all [9]. Instead, for these algorithms hold that the more complete a log is, the better the result.

For the first category of algorithms it is necessary to be able to determine whether or not an event log indeed contains all possible behaviour. For the second category, it is important to say whether or not a log is complete with respect to the specific assumptions, while for the third type of algorithm, any information on completeness is relevant for predicting the applicability of that algorithm.

This then leads to the question: how can one determine global or local completeness of an event log *in an objective manner*?

Naturally the completeness of an event log depends on the type of information one is interested in. For example, one may wish to mine organizational structures or temporal information about task or case completions. In this paper the focus is on mining of control-flow dependencies, in other words, determining which tasks exist and what their relative execution order is. The objective of mining such dependencies is to obtain a process model of which the (repeated) execution could have led to the event log. Model discovery has been widely studied and was among the first applications of process mining.

Having established a focus on control-flow dependencies, the next issue to address is how to measure or estimate the degree of completeness with respect to control-flow information in an event log. A logical choice for unit of information is the concept of a trace. While different definitions of traces are possible, e.g. the various moments of choice could be preserved as part of a trace [14], here we simply see a trace as a sequence of actions which constitute a successful completion of a process instance. Typically, completeness assumptions are formulated in terms of a process model that can generate the log. However, as we do not have access to that process model to determine whether this is the case, we estimate the degree of completeness in a probabilistic manner.

First we determine how many essentially different traces we have observed in the event log. This leads to a number of observed *trace classes*. For each trace class we then compute the occurrence error rate determined by the difference between the actual occurrence frequency of this class and its expected occurrence frequency. It is a central assumption of our work that this error rate provides an indication of the number of trace classes missing from the log. By selecting the right classification approach to obtain trace classes, several notions of global

and local completeness can be considered. However, in this paper, we focus on global completeness, i.e. the sequence of activities in a trace provides the trace class and traces are considered different if and only if their activity sequences are different.

It is important to note that our method for determining the degree of completeness of a log is not dependent on the choice of a specific mining algorithm. However, as different mining algorithms have different assumptions on completeness, we need to select a classification method that corresponds to these assumptions. Since we selected global completeness for this paper, our results can directly be used for those algorithms that assume global completeness [18,6,10]. For other algorithms, requiring local completeness, different trace classification methods have to be developed for our approach to be applicable.

The remainder of this paper is organized as follows. Section 2 describes basic concepts needed to define the problem and to describe our approach, while Section 3 contains definitions related to completeness (with respect to control-flow dependencies) of event logs. Then the proposed approach for determining the degree of completeness of an event log is outlined in a step-wise manner in Section 4. In Section 5 the results obtained are evaluated and examined in an experimental manner. Section 7 describes related work, while Section 8 concludes the paper and outlines future work.

2 Problem Definition

In this section a number of basic definitions are presented that are used to formalise our approach to determining log completeness followed by a fomulation of the two main problems that we aim to solve.

2.1 Basic Definitions

A *task* is an activity to be performed in the context of a business process. A *process model* provides an abstraction of a business process capturing its tasks and all possible execution orders of these tasks in a formal manner. A *process instance*, sometimes referred to as a *case* [3], represents the actual execution of a business process. A *trace* is the result of the successful completion of a process instance and consists of a sequence of events, where each *event* corresponds to the execution of a task [3].

The following definitions capture some of these concepts in a precise manner and they are almost the same as those presented in [4]. In this paper, our starting point is a set of tasks that are being executed in real life. We assume that we know all possible tasks involved in a process in advance and we assume the log ranges over these tasks. Furthermore, all events in the log should be ordered, typically using the time at which they were recorded.

Definition 1 (Log). *Let T be a finite set of tasks. We define $L = (E, C, \gamma, \tau, >)$ as an event log over T, where E is a set of events, C is a set of case identifiers,*

$\gamma : E \to C$ a surjective function relating events to cases, $\tau : E \to T$ a function relating events to tasks and $>\subseteq E \times E$ a total ordering on events.

Definition 2 (Trace). *Let T be a finite set of tasks and $L = (E, C, \gamma, \tau, >)$ a log over T. For all cases $c \in C$, we define the trace $\sigma_c = \{e \in E \mid \gamma(e) = c\}$ as all events relating to case c. Since the events in E are totally ordered, the events in σ_c are also totally ordered.*

The execution of instances of various process models manifests itself through the corresponding traces whose events are recorded in process logs.

Definition 3 (Event log length). *Let T be a finite set of tasks and $L = (E, C, \gamma, \tau, >)$ a log over T. We define the length of the event log L as the number of elements in C, which is denoted as $|C|$.*

Note that therefore an event log's length is identical to its number of traces.

As stated before, our approach is based on the identification of trace classes in the log and we focus on global completeness, i.e. the trace class of a trace is based on the sequence of tasks it represents.

Definition 4 (Trace equivalence). *Let T be a finite set of tasks and $L = (E, C, \gamma, \tau, >)$ a log over T. Furthermore, let $c_1, c_2 \in C$ be two case identifiers and $\sigma_{c_1}, \sigma_{c_2}$ the corresponding traces. We say that σ_{c_1} and σ_{c_2} are equivalent denoted by $\sigma_{c_1} \equiv \sigma_{c_2}$ if and only if for all $e_1 \in \sigma_{c_1}$ and $e_2 \in \sigma_{c_2}$ holds that $(\#_{e \in \sigma_{c_1}} e < e_1 = \#_{e \in \sigma_{c_2}} e < e_2) \implies \tau(e_1) = \tau(e_2)$ and $|\sigma_{c_1}| = |\sigma_{c_2}|$.*

Definition!4 states that two traces σ_{c_1} and σ_{c_2} are equivalent if and only if their lengths are equal and every event of the first trace refers to the same task as the corresponding event of the second trace, i.e. the one having the same position if put in a sequence based on the ordering relation.

Clearly, the trace equivalence relation defined above forms an *equivalence relation* and we will refer to the corresponding equivalence classes as *trace classes*. Sometimes we will refer to two equivalent traces as being the same and we will refer to a trace σ occuring n times in an event log L if and only if the corresponding trace class has n elements.

Definition 5 (Global completeness of an event log). *Let T be a finite set of tasks, P a process model over T and L a log over T. We say that L is* globally complete *if and only if the number of trace classes in the log according to the $\equiv$ relation equals the number of all possible execution orders of T in P.*

It should be pointed out that a process model's behaviour cannot be fully characterised through its trace classes. This is due to the fact that the notion of trace used is limited and does not preserve moments of choice, but also because no currently existing process mining algorithm is capable of discovering all workflow control-flow patterns [9], which constitutes an inherent limitation to process model discovery.

2.2 Problem Formulation

While different control-flow mining algorithms may require different kinds of input information, their purpose is the same, to derive a process model capturing the control-flow dependencies that are reflected in event logs. The essence of control-flow information is captured by the tasks that occur in the various events in an event log and their execution order in the various process instances. The more different traces the event log contains the more one can derive about the actual control-flow dependencies between tasks.

In [9], an overview of mining algorithms is presented. For each algorithm, insights are given into the requirement with respect to completeness of the log, i.e. if a log does not satisfy its requirements, then the result is not guaranteed to actually represent the process under consideration. In that case, more traces should be recorded.

However, once a log satisfies the completeness requirements of an algorithm, the addition of new traces does not influence the outcome anymore. In particular, if a log is global complete, then the addition of a trace will never introduce new trace classes.

Unfortunately, most completeness metrics are defined on the log and the model used to generate the log. As such a model is usually not present, it's not possible to determine the completeness of the log. (In fact, if the model was present, then process mining becomes a useless excercise.)

Allthough in [9], several completeness metrics are specified, in this paper, we are concerned with finding the answers to the following two problems, both related to estimating the extent to which a log is *globally complete*:

Problem 1 (Log length problem). Given a set of tasks T, an event log L over T for a process model P (not known), a confidence level K $(0 < K < 1)$ and a maximum error ε $(0 < \varepsilon < 1)$, what should the minimal length N of the log be in order to be able to assert with confidence level K that the maximum error between the real distribution and the empirical distribution of traces is no more than ε?

Problem 2 (Completeness problem). Given a set of tasks T, an event log L over T and a confidence level K $(0 < K < 1)$, how can we determine with confidence level K what the ratio is between the number of observed trace classes in L versus the total number of trace classes in P?

There is an interesting alternative formulation to Problem 2. The challenge is to guess for an urn with an unknown number of marbles, how many different colours these marbles may have. The only input to the problem is a number of selections of marbles where it was recorded what the colour of the marble was before it was put back in the urn [17]. This reformulation of the problem depends on a number of assumptions, which will be discussed in the next section.

3 Assumptions

In this section the assumptions that underpin our work are made explicit. Each assumption is described in detail and it is argued that the assumption is reasonable, why it is needed, and what would go wrong if the assumption was not made.

As defined before, we consider an event log to be complete if and only if it contains all possible trace classes of a certain process model. As we do not know this process model we cannot be certain whether an event log is complete or not. There are however ways of dealing with this uncertainty based on assumptions that can be reasonably made for event logs that help us estimate the probability of the appearance of a new trace for a given log. If this probability is small, then the degree of completeness of the log may be considered high. This approach provides us with an indirect way of estimating the degree of completeness of a given log.

Assumption 1 *Given a log, we know which tasks can occur in the log, i.e. we know in advance which tasks are present in the unknown process the log originated from.*

By assuming all tasks are known, we eliminate the possibility that each newly recorded case adds new tasks to the log, thus continuously increasing the number of trace classes. Since process mining algorithms generally only construct models using the tasks actually found in the log, this assumption is reasonable.

Assumption 2 *Traces of a process model appear randomly and independently.*

In a process-aware information system deployed in a large organisation, there are typically many instances of process models running concurrently. Instances of a process model may be initiated by a variety of stakeholders and may come about under a variety of circumstances. By observing the execution log, it cannot be predicted what the next trace will be, based on already observed traces. It seems reasonable to assume that traces are produced randomly and independently.

If the appearance of new traces depended on the already observed traces, we have to deal with correlated traces. Such traces should be treated as different appearances of the same trace as they can be (partially) derived from existing traces. If we cannot deduce how traces correlate with each other we may overestimate the number of existing trace classes and thus underestimate the degree of completeness of event logs.

Assumption 3 *Any trace of a process model appears with a constant but unknown probability. While constant for a particular trace class, this probability may vary across the various trace classes.*

A trace class represents a particular application scenario of a process model. When a process-aware information system has been running for years, the same scenario may appear periodically. As time goes by the *occurrence frequency* of

the various traces becomes relatively stable and in the long run a trace may be perceived to appear with a constant probability, which we will refer to as its *occurrence probability*.

Based on this assumption, we can approach the problem of determining the degree of completeness of an event log by treating it as a probability distribution estimation problem. Given an event log, we can obtain observed occurrence frequencies of trace classes. These frequencies form the *empirical distribution* of trace occurrences. The real occurrence probabilities of trace classes on the other hand, which are unknown, form the *real distribution* of trace occurrences. As mentioned before, the difference between these two provides a measure for the degree of completeness of an event log.

If this assumption is not satisfied, we cannot solve the problem of determining the degree of completeness of an event log without further information about occurrence frequencies of traces. It is worthwhile noting that this means that our approach does not work so well for event logs that result from processes that have not been running very long as trace occurrence frequencies may not have sufficiently stabilised.

Assumption 4 *The event log is noise-free.*

Noise in an event log represents incorrect information. The presence of noise in process logs still poses formidable problems for process mining. Some research in this area has been conducted and it seems that a commonly occurring approach to tackling the problem is to assume the presence of two noise patterns (see e.g. [20,26]). In one pattern the occurrence of some tasks was not recorded in the log, while in the other pattern events in a trace randomly exchanged start times. The solution to dealing with the occurrence of both patterns is to use filtering techniques. A trace or a task relationship is treated as noise if its occurrence frequency is lower than some specified threshold. Filtering can take place before application of a mining algorithm or afterwards. Noise-polluted traces or task relationships cannot be identified without additional knowledge or assumptions.

It seems reasonable to assume that event logs are not polluted. Noise is caused by hardware failures or by software bugs and the detection and remedy of these problems could lead one to also clean-up the event logs involved.

As we do not have ways of identifying noise in event logs, a noise-polluted log may lead our approach to overestimate the number of possible trace classes and thus to an underestimation of the actual degree of completeness of the log.

Assumption 5 *For all cases in the log, there is an event that identifies the first and an event that identifies the last task executed for this case.*

When recording events in a certain time interval, there will most likely be events that belong to traces of which the first event occured before the start of the logging period, as well as traces for which events will still occur after the logging period has ended. Again, filtering techniques can be used to eliminate these traces and to keep only those traces in the log to which no events will be appended in the future.

The reason for assuming only completed traces to be present in the log is that we want to ensure that the trace class of a trace does not change over time.

4 Determining the Degree of Completeness of an Event Log

In this section we discuss the proposed approach to determine event log completeness and provide the proofs for a number of fundamental results.

4.1 Approach

For an event log generated by an unknown process model we can determine the observed trace classes and on this basis, we can try to estimate the real occurrence probabilities of these trace classes. This does not provide us however with the number of trace classes missing. While it is hard to estimate the ratio of the number of observed trace classes versus the actual number of trace classes in a direct manner, as we do not know the actual process model, an indication of this ratio is provided by the differences between the observed probabilities of the appearance of traces of the various classes versus the actual probabilities of their appearance. The difference between the empirical distribution and the corresponding real distribution of traces provides a lower bound for the degree of completeness of a given event log. Note that this difference is expected to be smaller for large logs, as over time, with the size of the log increasing, the empirical distribution of occurrence probabilities of trace classes tends to converge to the real distribution of these probabilities due to the law of large numbers [21,16] (and using Assumptions 2 and 3). This corresponds to our intuition: larger logs tend to be more informative.

The error ε, reflecting the aforementioned difference between real and empirical distribution of trace class frequencies, consists of two parts. The first part is determined by the difference between the number of observed trace classes and the total number of trace classes, while the second part is determined by the difference between the appearance frequencies of observed trace classes and their real frequencies. As the second part of the error is not negative, the first part of the error does not exceed ε. Therefore $1 - \varepsilon$ provides a lower bound for the degree of completeness of an event log, since ε provides an upper bound. The key challenge now is to find the error ε where the real distribution is unknown. To solve this problem we apply Chebyshev's Inequality [22,16].

To summarise, we have converted the problem of determining the degree of completeness of event logs into the problem of estimating the difference between the empirical distribution and the real distribution of trace appearances in these logs.

First, we present the notational conventions used in this section, which are summarized in table 1.

- For a given finite set of tasks T and an unknown process model P, we assume that there are W trace classes in total and a trace class is denoted as T_j where $0 < j \leq W$.

- The occurrence probability of a trace class T_j is denoted as $P(T_j) = p_j$, where $0 < j \leq W$. If no confusion can occur we write p instead of p_j.
- An event log $L = (E, C, \gamma, \tau, >)$ over a set of tasks T contains $N = |C|$ traces of M classes, where $1 \leq M \leq |C|$.
- Given an event log L over a set of tasks T and a trace class T_j, then T_j occurs μ_j times in the log.
- The occurrence frequency of a trace class T_j is denoted as $\frac{\mu_j}{N}$.
- Trace classes from T_{M+1} to T_W are unobserved and hence, their occurrence frequencies have the value of zero.

Table 1. The occurrence probabilities and the occurrence frequencies of trace classes

Trace classes	T_1	T_2	$\cdots$	T_{M-1}	T_M	T_{M+1}	$\cdots$	T_W
Probability	p_1	p_2	$\cdots$	p_{M-1}	p_M	p_{M+1}	$\cdots$	p_W
Frequency	$\frac{\mu_1}{N}$	$\frac{\mu_2}{N}$	$\cdots$	$\frac{\mu_{M-1}}{N}$	$\frac{\mu_M}{N}$	0	$\cdots$	0

Section 4.2 describes how the bounds for an error probability for a single trace class and for all observed trace classes in a given log can be calculated. Section 4.3 demonstrates how these results are used to determine a lower bound for the length of a log. Section 4.4 demonstrates how to estimate the degree of completeness of a given log. Section 4.5 discusses our insight into the relationships between the four variables used in the calculations, namely, the error ε, the log length N, the number of observed trace classes M and the confidence level K.

4.2 Determining the Error Probability for Trace Classes

Theorem 1 shows how an upper bound on the error probability for a single trace class can be estimated for a given log.

Theorem 1. *Let ε_j be a maximum error specified for trace class T_j of a log L with length N. It holds that*

$$P\left\{\left|\frac{\mu_j}{N} - p_j\right| \geq \varepsilon_j\right\} \leq \frac{1}{4N\varepsilon_j^2},$$

where $|\frac{\mu_j}{N} - p_j|$ represents the difference between the occurrence frequency of T_j and its occurrence probability.

Proof. A random variable X_i is used to describe the appearance of the ith trace in L. Let $X_i = 1$ if the ith trace of L belongs to trace class T_j, otherwise $X_i = 0$. Thus, $P(X_i = 1) = p_j$, and $P(X_i = 0) = 1 - p_j$. Then $X_i \sim Bern(p_j)$ is a

Bernoulli distribution [21], and its mean value EX_i and variance value DX_i are calculated as follows.

$$EX_i = p_j, \tag{1}$$
$$DX_i = p_j(1-p_j) \tag{2}$$
$$= \frac{1}{4} - (p_j - \frac{1}{2})^2 \tag{3}$$
$$\leq \frac{1}{4}. \tag{4}$$

Similarly, a random variable can be defined for each of the other traces in N. Based on Assumptions 2 and 3 in Section 3, $\{X_i, 0 \leq i \leq N\}$ are independent identical distributed (i.i.d.) [21] as the various X_i do not impact each other and they all have identical distributions with values 0 and 1.

Let $\mu_j = \sum_{i=1}^{N} X_i$ be the occurrence frequency of trace class T_j, then its mean and variance can be calculated as follows.

$$E\left(\frac{\mu_j}{N}\right) \tag{5}$$
$$= \frac{1}{N} E\mu_j \tag{6}$$
$$= \frac{1}{N} E(\sum_{i=1}^{N} X_i) \qquad \text{(Definition of } \mu_j\text{)} \tag{7}$$
$$= \frac{1}{N} \sum_{i=1}^{N} EX_i \tag{8}$$
$$= \frac{1}{N} \sum_{i=1}^{N} p_j \tag{9}$$
$$= \frac{1}{N} * N * p_j \tag{10}$$
$$= p_j. \tag{11}$$

$$D\left(\frac{\mu_j}{N}\right) \tag{12}$$

$$=D(\frac{1}{N}\sum_{i=1}^{N} X_i) \qquad \text{(Definition of } \mu_j\text{)} \tag{13}$$

$$=\frac{1}{N^2}D(\sum_{i=1}^{N} X_i) \tag{14}$$

$$=\frac{1}{N^2}\sum_{i=1}^{N} DX_i \tag{15}$$

$$=\frac{1}{N^2}\sum_{i=1}^{N} p_j(1-p_j) \qquad (\{X_i\} \text{ are i.i.d., where } 1 \leq i \leq N) \tag{16}$$

$$\leq\frac{1}{N^2}\sum_{i=1}^{N} \frac{1}{4} \qquad \text{(Expression 4)} \tag{17}$$

$$=\frac{1}{N^2} * \frac{N}{4} \tag{18}$$

$$=\frac{1}{4N} \tag{19}$$

According to Chebyshev's Inequality [22,16], for every $\varepsilon_j > 0$, the following holds.

$$0 \leq P\{|\frac{\mu_j}{N} - p_j| \geq \varepsilon_j\} \tag{20}$$

$$\leq \frac{1}{\varepsilon_j^2} D(\frac{\mu_j}{N}) \qquad \text{(according to Chebyshev's Inequality)} \tag{21}$$

$$\leq \frac{1}{\varepsilon_j^2} * \frac{1}{4N} \qquad \text{(Expression 19)} \tag{22}$$

$$= \frac{1}{4N\varepsilon_j^2}. \tag{23}$$

□

The result obtained in Theorem 1 takes the form of an inequality which depends on two variables only, the log length N and an upper bound for the error ε_j. In order for the inequality to be applicable, the estimation for the probability of the error rate for a single trace class should be less than one. This means that log length N should not be less than $1/(4\varepsilon_j^2)$, which can be the case for small N in combination with a small maximum error ε_j.

From the inequality in Theorem 1, it is clear that the occurrence frequency of a trace class does not appear in the expression. This implies that such a bound can be derived without considering the possible appearances of other trace classes in an event log. Hence, we can estimate the error probability of all trace classes observed in a log in a similar manner. Corollary 1 demonstrates how to calculate a lower bound for the error probability for all trace classes.

Corollary 1. *Let M be the number of observed trace classes in a given log L with length N and ε be the maximum error specified for L. It holds that*

$$P\left\{\sum_{i=1}^{M}\left|\frac{\mu_i}{N}-p_i\right|<\varepsilon\right\}\geq 1-\frac{M^3}{4N\varepsilon^2},$$

where μ_i is the number of times a trace class T_i $(0<i\leq M)$ appears in the log and p_i $(0<i\leq M)$ is the occurrence probability of T_i.

Proof.

$$P\left\{\sum_{i=1}^{M}\left|\frac{\mu_i}{N}-p_i\right|<\varepsilon\right\} \tag{24}$$

$$=P\left\{\sum_{i=1}^{M}\left|\frac{\mu_i}{N}-p_i\right|<\sum_{i=1}^{M}\frac{\varepsilon}{M}\right\} \tag{25}$$

$$\geq P\left\{\bigcap_{i=1}^{M}\left(\left|\frac{\mu_i}{N}-p_i\right|<\frac{\varepsilon}{M}\right)\right\} \tag{26}$$

$$=1-P\left\{\bigcup_{i=1}^{M}\left(\left|\frac{\mu_i}{N}-p_i\right|\geq\frac{\varepsilon}{M}\right)\right\} \tag{27}$$

$$\geq 1-\sum_{i=1}^{M}P\left\{\left|\frac{\mu_i}{N}-p_i\right|\geq\frac{\varepsilon}{M}\right\} \tag{28}$$

$$\geq 1-\sum_{i=1}^{M}\frac{1}{4N(\frac{\varepsilon}{M})^2}\quad\text{(Theorem 1)} \tag{29}$$

$$=1-\frac{M^3}{4N\varepsilon^2} \tag{30}$$

□

From the inequality, we can see that the bound for an error probability of all trace classes depends on three variables M, N and ε. For the given values of ε and M, the log length should not be less than $1-M^3/(4N\varepsilon^2)$. This represents a lower bound for the length of an event log for all observed trace classes. As stated, the error probability is greater than $1-M^3/(4N\varepsilon^2)$ if the error for any of M trace classes is less than ε/M. To make this expression valid, M^3 should not be greater than $4N\varepsilon^2$. We say that the error probability is 0 if this condition does not hold.

4.3 Calculating a Lower Bound for the Length of an Event Log

Corollary 2. *Let M be the number of observed trace classes in a log L with length N, ε be the maximum error and K be the minimum confidence level.*

Based on Assumptions 2 and 3 in Section 3, if the error between the empirical distribution and its real distribution is less than ε, and the confidence level K, a lower bound for the length of an event log is

$$\frac{M^3}{4\varepsilon^2\,(1-K)} \leq N. \tag{31}$$

Proof. From Corollary 1, we know

$$P\left\{\sum_{i=1}^{M}\left|\frac{\mu_i}{N}-p_i\right|<\varepsilon\right\}\geq 1-\frac{M^3}{4N\varepsilon^2}. \tag{32}$$

Let

$$K \leq 1-\frac{M^3}{4N\varepsilon^2}. \tag{33}$$

Then

$$\frac{M^3}{4N\varepsilon^2} \leq 1-K. \tag{34}$$

Since $0 < K < 1$, $1 - K > 0$, and thus

$$\frac{M^3}{4(1-K)\varepsilon^2} \leq N. \tag{35}$$

□

From the inequality, we can see that the lower bound for N depends on three variables M, K and ε. For the given values of ε and K, we can adjust the value of either N or M to ensure that a certain confidence interval K holds. In reality, the number of observed trace classes M cannot be controlled. Hence, the only variable that can be adjusted is N. It means that we wait for more traces to appear and hence, we work with a log with a larger value of N. Using these results, we can answer the log length question raised in Section 2.

For an event log L for a process model P (not known), a confidence level K ($0 < K < 1$), and a maximum error ε ($0 < \varepsilon < 1$), in order to assert with confidence level K that the maximum error between the real distribution and the empirical distribution of traces is no more than ε, the minimal length N of the log should be $\frac{M^3}{4(1-K)\varepsilon^2}$.

4.4 Analysis of the Magnitude of Information in a Log

As long as the total number of trace classes is finite, we can estimate the probability of unobserved trace classes without knowing the exact number of such classes.

Corollary 3. *Let W be the total number of trace classes of process model P, M be the number of observed trace classes in the given log L with length N, and ε be the maximum error. It holds that*

$$P\left\{\sum_{i=M+1}^{W} p_i < \varepsilon\right\} \geq 1 - \frac{M^3}{4N\varepsilon^2}, \tag{36}$$

where p_i $(M < i \leq W)$ is the occurrence probability of trace class T_i $(M < i \leq W)$ in the log.

Proof.

$$P\left\{\sum_{i=M+1}^{W} p_i < \varepsilon\right\} \tag{37}$$

$$= P\left\{1 - \sum_{i=1}^{M} p_i < \varepsilon\right\} \tag{38}$$

$$= P\left\{\left|1 - \sum_{i=1}^{M} p_i\right| < \varepsilon\right\} \quad (1 - \sum_{i=1}^{M} p_i \geq 0 \text{ is always true, so we can add } |\,|.) \tag{39}$$

$$= P\left\{\left|\sum_{i=1}^{M} \frac{\mu_i}{N} - \sum_{i=1}^{M} p_i\right| < \varepsilon\right\} \quad (\sum_{i=1}^{M} \frac{\mu_i}{N} = 1 \text{ holds as } \sum_{i=1}^{M} \mu_i = N.\ ) \tag{40}$$

$$= P\left\{\left|\sum_{i=1}^{M} (\frac{\mu_i}{N} - p_i)\right| < \varepsilon\right\} \tag{41}$$

$$\geq P\left\{\sum_{i=1}^{M} \left|\frac{\mu_i}{N} - p_i\right| < \varepsilon\right\} \tag{42}$$

$$\geq 1 - \frac{M^3}{4N\varepsilon^2} \quad \text{(Corollary 1)} \tag{43}$$

□

Corollary 3 shows that the probability of the sum of the probabilities of all unobserved trace classes being less than ε is greater than or equal to $1 - M^3/(4N\varepsilon^2)$. Here ε can be seen as an upper bound for the unknown information in the distribution. In other words, we can use $1 - \varepsilon$ to denote the lower bound for the completeness of a log. Similar to the lower bound for N, we can estimate the lower bound for the completeness of a log as shown in Corollary 4.

Corollary 4. *Let M be the number of observed trace classes in the given log L with length N, ε be the maximum error and K be the confidence level. With the confidence level K, a lower bound for $1 - \varepsilon$, is*

$$0 \leq 1 - \varepsilon \leq 1 - \frac{M^{\frac{3}{2}}}{2\sqrt{N\,(1-K)}} \leq 1 \tag{44}$$

Proof. From Corollary 2, it holds that

$$\frac{M^3}{4\,(1-K)\,\varepsilon^2} \leq N. \tag{45}$$

Since $N > 0$, then

$$\frac{M^3}{4N\,(1-K)} \leq \varepsilon^2. \tag{46}$$

And since $\varepsilon > 0, 1-K > 0$ and $M > 0$ then

$$\frac{M^{\frac{3}{2}}}{2\sqrt{N\,(1-K)}} \leq \varepsilon. \tag{47}$$

And then

$$1-\varepsilon \leq 1 - \frac{M^{\frac{3}{2}}}{2\sqrt{N\,(1-K)}}. \tag{48}$$

The degree of completeness should be in the range of [0..1], therefore,

$$0 \leq 1-\varepsilon \leq 1 - \frac{M^{\frac{3}{2}}}{2\sqrt{N\,(1-K)}} \leq 1 \tag{49}$$

□

From the inequality, we can see that it is a function with four variables N, M, ε and K. For a given log, N and M are known. For a given confidence level K, we can then calculate the lower bound for ε, where ε is the sum of the probabilities of all unobserved trace classes. The integral of the real distribution is 1, which represents the upper limit of the magnitude of all possible information contained in a log. We can then assert with confidence level K that the information contained in the empirical distribution is not less than $1-\varepsilon$.

Please note that without knowing the process model, the exact ratio between the number of observed trace classes and the total number of trace classes is unknown. Clearly, the magnitude of completeness, $1-\varepsilon$, is not identical to the ratio. However, since $1-\varepsilon$ is very close to the ratio and it will converge to the ratio when the log length gets closer to infinity, we use $1-\varepsilon$ to estimate the ratio and we call $1-\varepsilon$ the magnitude of completeness. Using these results, we can answer the completeness question raised in Section 2.

Given an event log L for process model P (not known) and a confidence level K ($0 < K < 1$). With confidence level K, the ratio between the number of observed trace classes in L and the total number of trace classes in P is estimated by means of occurence probability to have an upper bound value of $1 - \frac{M^{\frac{3}{2}}}{2\sqrt{N(1-K)}}$.

4.5 Interrelationships between Variables

Inequality (31) shows how a lower bound for N can be estimated provided that we know the values of the other three variables: the error ε, the confidence level K, and the number of observed trace classes M. To understand the impact of each of these variables on N, we now look at each of them in more detail. First, we give the formula on the lefthand-side of the inequality a function name $f(\varepsilon, K, M)$ and then divide the formula into three parts and give each of them a function name (i.e., $g(\varepsilon)$, $h(K)$ and $i(M)$).

$$\begin{aligned} f(\varepsilon, K, M) &= \frac{M^3}{4\varepsilon^2(1-K)} \\ &= \frac{1}{\varepsilon^2} \times \frac{1}{1-K} \times \frac{M^3}{4} \\ &= g(\varepsilon) \times h(K) \times i(M) \\ &\leq N. \end{aligned}$$

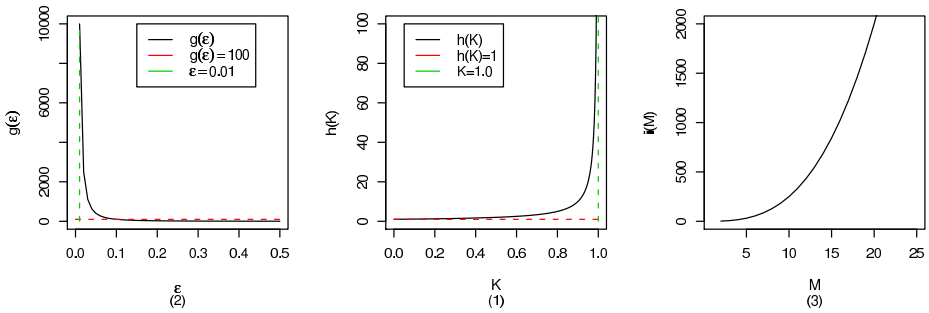

Fig. 1. The impact factors

- $g(\varepsilon)$: While estimating a lower bound for N, ε represents an accepable maximum estimation error and $1-\varepsilon$ represents the lower bound on the magnitude of information contained in a log. Fig. 1(1) shows that the value of N increases very sharply when the value of ε goes down below 0.05. This indicates that the lower the estimation error, the longer the length should be.
- $h(K)$: K is the confidence level of the estimation and takes on the value in the range of 0..1. In other words, $1-K$ represents the probability of making a mistake. Fig. 1(2) shows that the value of $h(K) = 1/(1-K)$ increases as the value of K increases. It increases at a slower rate when the value of K is nearer to 0.0 and increases at a much faster rate as the value of K gets closer to 1. This indicates that the higher the confidence level, the longer the log should be.
- $i(M)$: M is the number of observed trace classes in a given log. Fig. 1(3) shows that the value of $i(M) = M^3/4$ increases very sharply when the value of M increases. This indicates that the more trace classes we have observed, the longer the log should be.

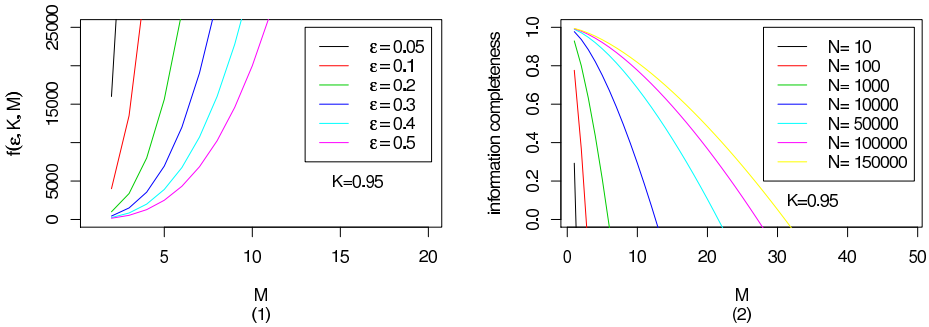

Fig. 2. The expected log length and the magnitude of completeness.

We now discuss the effects by means of the overall function $f(\varepsilon, K, M)$. Fig. 2(1) shows that the value of N increases very quickly as M increases for $K = 0.95$ and with various values of ε. It can be seen that for given values of ε and K, the speed of an increase in the lower bound for N is much faster than M^3. From this, we can conclude that the value of M determines the increasing speed and ε the magnitude of the lower bound for N.

Similarly, we can see the effects of the variables M, N and K on the required magnitude of information in a log using inequality (44). Fig. 2(2) shows that the lower bound on the completeness, $1 - M^{\frac{3}{2}} / \left(2\sqrt{N\,(1-K)}\right)$, increases with an increase in the log length N (from 10 to 150000). We can also see that this lower bound reduces quite sharply with an increase in the number of trace classes M. This is as we expected. If the traces are repeated again and again, we assume that the appearance of a new trace is less probable. And if each of the trace classes is seldom repeated or appeared only once in a given log, we may assume that there are many more new traces likely to appear.

5 Experiments

In this section, we demonstrate the applicability of the proposed approach by running experiments on a number of logs. In Corollary 2, we showed that it is possible to estimate the lower bound for N using three variables - the number of observed trace classes M, an acceptable error ε and the confidence level K as follows:

$$\frac{M^3}{4\varepsilon^2\,(1-K)} \leq N. \tag{50}$$

In Corollary 4, we showed how the completeness, $1 - \varepsilon$, of each log can be estimated using three variables - the number of observed trace classes M, the log length N and the confidence level K as follows:

$$0 \leq 1 - \varepsilon \leq 1 - \frac{M^{\frac{3}{2}}}{2\sqrt{N\,(1-K)}} \leq 1. \tag{51}$$

Based on these inequalities, we implemented a plugin for the meta-data package in the ProM framework [23]. In Section 6 we show how to access our implementation.

The input log files for our experiments were obtained from the example logs publicly available from the ProM website[4]. There were altogether 600 log files available for testing. We parsed all these logs to calculate the log lengths, all observed trace classes, and the occurrence frequencies of trace classes. Upon close inspection of these logs, we found that only five out of 600 logs were long enough for our experiments[5]. All of them contain five trace classes each ($M = 5$). The lengths of these log vary between 200 to 1800. We then calculated the lower bounds for the log length and the completeness for the example logs.

Table 2. Lower bounds for the length of example logs

	# of trace classes	K=0.95	K=0.90	K=0.85	K=0.80	K=0.75
$\varepsilon = 0.1$	5	62500	31250	20834	15625	12500
$\varepsilon = 0.2$	5	15625	7813	5209	3907	3125

Table 2 summarizes our findings on the lower bounds for the length of example logs. The results show that a lower bound for N decreases either as the confidence level K decreases or as the error value ε increases. Given the maximum error value of 0.1, the expected length for logs reduces from 62500 to 12500 when the confidence level K decreases from 0.95 to 0.75. We can also see that the expected log length increases approximately fourfold when the maximum error ε is halved from 0.2 to 0.1. This confirms that the smaller the difference between the empirical distribution and the real distribution, the more traces are needed. $1 - K$ represents the probability of making a mistake while estimating the expected length. The shorter the log is, the higher the probability is of making a mistake when asserting that the difference between the two distributions is to be no more than ε (i.e., 0.1 or 0.2).

Table 3 summarizes our findings on the lower bounds for information completeness of example logs. For example, for the log `a12f0n00_2.xml` that has 600 traces and five trace classes, we can assert with 75% confidence that the degree of completeness is 54.36%. For log `a12f0n00_3.xml` with 1000 traces, we can assert the same level of completeness of 54.36% but with a higher confidence level of 90%, since it contains an additional 400 traces over log `a12f0n00_2.xml`. As logs `a12f0n00_1.xml` and `a12f0n00_2.xml` only contain 200 and 600 traces respectively, they are not long enough to calculate completeness values for higher values of K. For the confidence level value of 80%, we can see the completeness

[4] http://prom.win.tue.nl/tools/prom/downloads/Process%20Log%20examples.zip

[5] Although the log lengths from the input files are not used in the calculations, they determine whether the calculation can be done.

Table 3. Lower bounds of completeness of example logs[a]

Log	Log Length	K=0.95	K=0.90	K=0.85	K=0.80	K=0.75
a12f0n00_1.xml	200	n/a	n/a	n/a	11.61 %	20.94 %
a12f0n00_2.xml	600	n/a	27.83 %	41.07 %	48.97 %	54.36 %
a12f0n00_3.xml	1000	20.94 %	44.10 %	54.36 %	60.47 %	64.64 %
a12f0n00_4.xml	1400	33.18 %	52.75 %	61.42 %	66.59 %	70.12 %
a12f0n00_5.xml	1800	41.07 %	58.33 %	65.98 %	70.54 %	73.65 %

[a] All the log files have 5 trace classes each.

value increases from 11.61% to 70.54% when the length of the log increases from 200 to 1800. These results show that the lower bound of the degree of completeness decreases as the confidence level increases, and increases as the log length increases.

6 Implementation

Figures 3 to 8 show how to access our implementation in ProM. Note that our implementation shows up in the nightly build version of ProM. First, as shown in Figure 3, the "Completeness" package needs to be installed.

Figure 4 shows the 5 logs from Table 3 opened in ProM. After opening a log in ProM, the "Log Meta Data Calculation" plugin needs to be started on one of the logs, as shown in Figure 5.

The "Log Meta Data Calculation" plugin presents several metrics to choose from, among which the "Completeness: Global Completeness" metric that implements the work in this paper (Figure 6). After clicking "Finish", the settings for this metric will be asked, i.e. the confidence level and the maximum error, as shown in Figure 7.

Finally, after completion of the plugin, the completeness information has been added as an attribute to the log, as can be seen in Figure 8. This log can now be exported to file again so that the completeness value is recorded for future use[6].

7 Related Work

We now present the related work to our research in the fields of process mining, statistics, and probability theory.

In the area of process mining, the notions of event, event log, and trace have been well defined [5]. A typical definition of an event log covers the name of a task, a process instance and the execution order between tasks (e.g. [25]) because these are the most important aspects for control-flow mining. This is referred to as the required *minimal information* in [2]. While the completeness problem

[6] ProM can freely be obtained from www.processmining.org

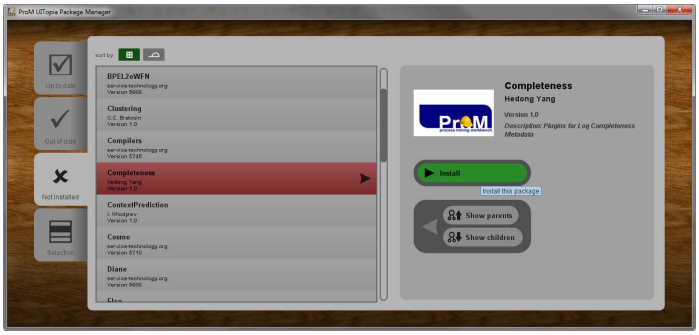

Fig. 3. Our Package in the ProM Package Manager

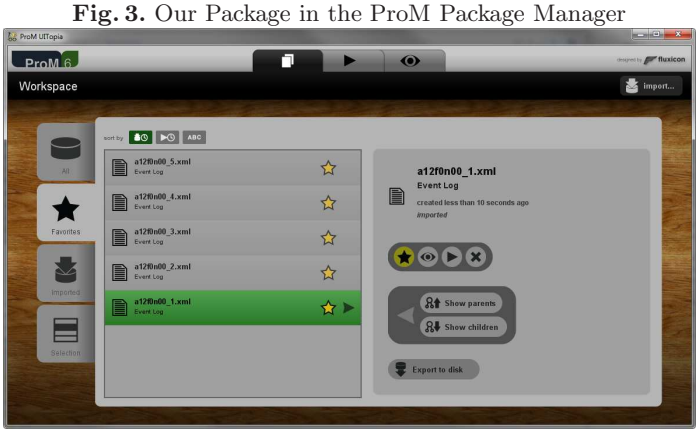

Fig. 4. The logs of Table 3 in ProM

has been mentioned in several papers (e.g. [5,19,1]), it has not been discussed in detail. The issue of log completeness was first raised by van der Aalst in [4]. The incompleteness of logs is often seen as a kind of noise [2]. Although many process mining algorithms require the log to be complete, there are other algorithms such as the genetic mining algorithm [1] that can work with incomplete logs. In [9], completeness is one of the metrics used to evaluate process mining algorithms. As the quality of mining results relies heavily on the degree of completeness of event logs, an in-depth exploration of this notion, the topic of this paper, is worthwhile.

The definition of completeness given in the literature varies depending on the objective of a process mining algorithm [9]. If an algorithm requires information about relationships between tasks, the log is considered to be incomplete if not all such relationships are present in the log. A typical definition is given in [5] where the α-algorithm uses the $>_W$ relationship defined between tasks and requires all possible pairs to appear in the log at least once. In our approach a log is considered to be complete if at least one trace of all possible trace classes is present in the log. The disadvantage of this approach is that process models

Fig. 5. The logs of Table 3 in ProM

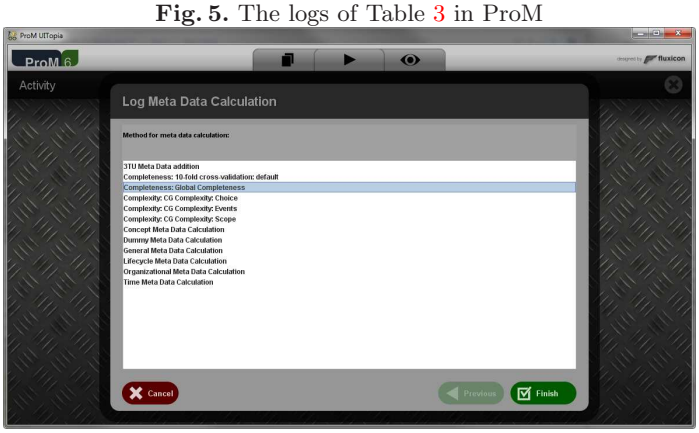

Fig. 6. The logs of Table 3 in ProM

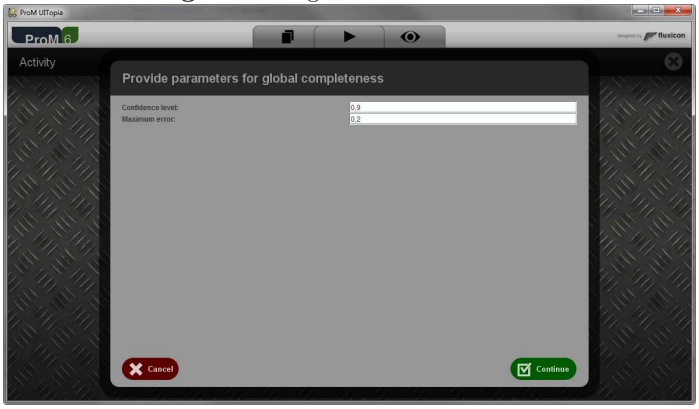

Fig. 7. The logs of Table 3 in ProM

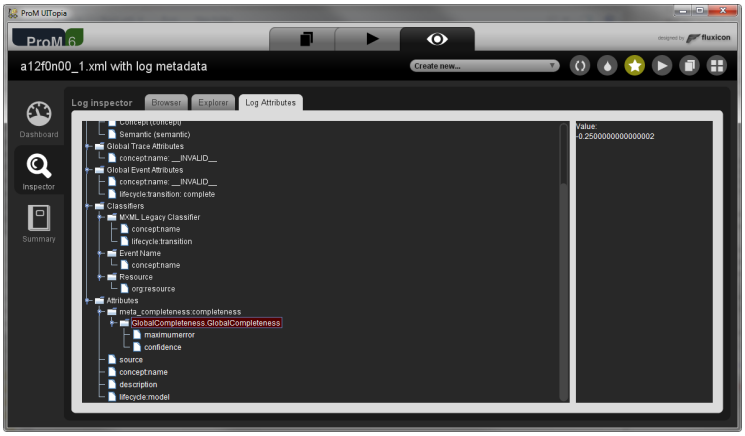

Fig. 8. The logs of Table 3 in ProM

with loops have an infinite number of traces. On the other hand, our approach aims to be general with respect to the notion of global completeness and is therefore not tailored to any specific algorithm that requires such completeness.

In this paper, we modeled the problem of completeness of event logs as the problem of estimating the difference between two probability distributions, namely the empirical distribution and the real distribution of trace appearances. Similar problems exist in the field of statistics. In our case the empirical distribution is known. If the real distribution is known, the problem is a traditional goodness-of-fit test problem, which can be solved by means of the Kolmogorov-Smirnov test (K-S test) whether the distributions are continuous or not [13]. If the distribution family of the real distribution is known, the problem of estimating the difference between a real and an empirical distribution can be solved by estimating the exact real distribution first. The maximum likelihood estimator is often a good choice for estimating the (unknown) parameters of this distribution [24]. In our case however, we do not have any a priori knowledge about the real distribution. The Dvoretzky-Kiefer-Wolfowitz Inequality can measure the difference between an empirical distribution and a real distribution with sample size n and error ratio ε as parameters [24]. This approach is not applicable for our objectives, as the number of trace classes is an important variable which is not taken into account in this inequality. The work by M.E. Lladser reported in [17] is most closely related to our approach. In [17] an iterative algorithm is proposed to estimate the unseen portions of samples. Unfortunately, this algorithm cannot directly be used to provide a lower bound for the length of a log given a desired degree of completeness and a confidence level (Problem 1). Additionally, the paper does not provide a solution to the problem of estimating degree of completeness of a log in analytical form (an equality or inequality).

The problem of completeness is similar to, but not the same as, the Coupon Collecting Problem (refer to page 322 of [22])[7] in the field of probability theory. Both problems require one to determine the number of different elements in an unknown collection through sampling with replacement. The difference between these problems is that both the total number of coupon types and the appearance probability of each coupon type are known in the Coupon Collecting Problem, while neither the number of trace classes nor the occurrence probability of each class is known in the completeness problem. Thus none of the solutions (e.g. [12]) to the Coupon Collecting Problem can be applied to directly solve the completeness problem.

8 Conclusions and Future Work

In this paper, the log length and completeness problems of event logs for control-flow mining were examined. Based on some natural and reasonable assumptions, an approach was derived to calculate a lower bound for the length of a log given an acceptable error rate ε a confidence level K, and the number of observed trace classes M. We concluded that $1 - \varepsilon$ can be used as a measure for the degree of information present in an event log and a formula was derived to provide a lower bound for this measure. Subsequently, the various factors influencing the presented lower bounds and their interrelationships were examined and experiments with some existing logs were conducted to provide some indication of their potential use in practice.

The work presented in this paper can be extended in several directions. If one considers models with loops, then our approach has a drawback. For logs that result from the execution of such models our estimations may not be very accurate as they tend to underestimate the actual degree of completeness. This is due to the fact that different iterations of the body of a loop will lead to different trace classes (while such repeated executions add less and less information). In order to extend our approach, one could consider alternative definitions of trace classes. For example, one could define trace classes based on task precedence relations as this would always yield a finite number of trace classes (given that the number of tasks is assumed to be finite). Such a definition is particularly meaningful in case of the well-known α-algorithm (as introduced in [5]). In fact, one could examine various definitions of trace classes in relation to the log length and information completeness problems, and link these definitions to process mining algorithms for which they are particularly suited. In that case the focus can be increased to include local completeness. Another extension to this work could assume the existence of noise in logs.

[7] This problem is also referred to as Coupon Collector Problem in [15,12] or as Coupon Collection Problem in [7].

Acknowledgements

The authors would like to thank Lijie Wen and Yahui Lu for their encouragement and advice on the choice of topic. The authors are also grateful to Jianxing Feng and Geng Zou for their feedback on the proofs presented in this paper.The authors would like to thank Kees van Hee for pointing them to the Coupon Collecting Problem.

The work reported in this paper is partially supported by the 973 Program of China (No. 2009CB320700), the 863 Program of China (No. 2008AA042301) and the projects of the National Natural Science Foundation of China (NSFC) (No.61073005 and No. 61003099).

References

1. van der Aalst, W.M.P., Alves de Medeiros, A.K., Weijters, A.J.M.M.: Genetic process mining. In: Ciardo, G., Darondeau, P. (eds.) Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, FL, USA, June 20-25, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3536, pp. 48–69. Springer (2005)
2. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: A survey of issues and approaches. Data and Knowledge Engineering 47(2), 237–267 (2003)
3. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, USA (2004)
4. van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining: A research agenda. Computers in Industry 53(3), 231–244 (2004)
5. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
6. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process mining based on regions of languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM'07: Proceedings of the 5th International Conference on Business Process Management. Lecture Notes in Computer Science, vol. 4714, pp. 375–383. Springer-Verlag, Berlin, Heidelberg (2007)
7. Bhattacharya, R.N., Waymire, E.C.: Stochastic Processes with Applications. John Wiley & Sons, Inc., New York, NY, USA (1990)
8. van Dongen, B.F., Alves de Medeiros, A.K., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) Application and Theory of Petri Nets 2005. vol. 3536, pp. 444–454 (2005)
9. van Dongen, B.F., Alves de Medeiros, A.K., Wen, L.: Process mining: Overview and outlook of petri net discovery algorithms. T. Petri Nets and Other Models of Concurrency 5460, 225–242 (2009)
10. van Dongen, B.F., Busi, N., Pinna, G.M., van der Aalst, W.M.P.: An iterative algorithm for applying the theory of regions in process mining. BETA Working Paper Series, WP 195, Eindhoven University of Technology, Eindhoven, the Netherlands (2007)

11. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M. (eds.): Process-Aware Information Systems : Bridging People and Software through Process Technology. Wiley-Interscience, Hoboken, NJ, USA (2005)
12. Flajolet, P., Gardy, D., Thimonier, L.: Birthday paradox, coupon collectors, caching algorithms and self-organizing search. Discrete Appl. Math. 39(3), 207–229 (1992)
13. Gleser, L.J.: Exact power of goodness-of-fit tests of kolmogorov type for discontinuous distributions. Journal of the American Statistical Association 80(392), 954–958 (1985)
14. Hidders, J., Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M., Verelst, J.: When are two workflows the same? In: Atkinson, M., Dehne, F. (eds.) Proceedings of Computing: The 11th Australasian Theory Symposium (CATS'2005). Conferences in Research and Practice in Information Technology, vol. 41, pp. 3–11. Australian Computer Society, Newcastle, Australia (February 2005)
15. Kao, E.P.C.: An Introduction to Stochastic Processes. Duxbury Press, North Scituate, MA, USA (1997)
16. Lin, Y., Liang, Z.: An Introduction to Probability Theory (in Chinese). Tsinghua University Press, Beijing, China (July 2003)
17. Lladser, M.E.: Prediction of unseen proportions in urn models with restricted sampling. In: Proceedings of the Sixth Workshop on Analytic Algorithmics and Combinatorics (ANALCO'09). pp. 85–91. SIAM, New York, NY, USA (January 2009)
18. Lorenz, R., Mauser, S., Juhás, G.: How to synthesize nets from languages : A survey. In: WSC '07: Proceedings of the 39th Conference on Winter Simulation. pp. 637–647. IEEE Press, Piscataway, NJ, USA (2007)
19. Maruster, L.: A Machine Learning Approach to Understand Business Processes. Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2003)
20. Maruster, L., Weijters, A.J.M.M., van der Aalst, W.M.P., Bosch, A.: A rule-based approach for process discovery: Dealing with noise and imbalance in process logs. Data Min. Knowl. Discov. 13(1), 67–87 (2006)
21. Papoulis, A.: Probability, Random Variables, and Stochastic Processes (2nd Edition). McGraw-Hill International Book Company, New York, NY, USA (1984)
22. Ross, S.M.: Introduction to Probability Models, Tenth Edition. Academic Press, Inc., Orlando, FL, USA (2009)
23. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Prom 6: The process mining toolkit. In: Proc. of BPM Demonstration Track 2010. vol. 615, pp. 34–39. CEUR-WS.org (2010), http://ceur-ws.org/Vol-615/paper13.pdf
24. Wasserman, L.: All of Statistics : A Concise Course in Statistical Inference (Springer Texts in Statistics). Springer (September 2004)
25. Weijters, A.J.M.M., van der Aalst, W.M.P.: Workflow mining: Discovering workflow models from event-based data. In: Dousson, C., Höppner, F., Quiniou, R. (eds.) Proceedings of the ECAI Workshop on Knowledge Discovery from Temporal and Spatial Data. pp. 78–84. Lyon, France (2002)
26. Xiong, H., Pandey, G., Steinbach, M., Kumar, V.: Enhancing data analysis with noise removal. IEEE Transactions on Knowledge and Data Engineering 18(3), 304–319 (2006)