# Discovering process models by rule set induction

Laura Măruşter[1], A.J.M.M. (Ton) Weijters[1],
Antal van den Bosch[2], and Walter Daelemans[2]

[1] Eindhoven University of Technology,
PO Box 513, 5600 MB Eindhoven, The Netherlands
{l.maruster,a.j.m.m.weijters}@tm.tue.nl
[2] Tilburg University, Tilburg 5000 LE, The Netherlands
{antal.vdnbosch,walter.daelemans}@uvt.nl

**Abstract.** Effective information systems require the existence of explicit process models; a completely specified process design needs to be developed in order to enact a given business process. This development is time consuming and the resulting models are often subjective and incomplete. We propose a method that discovers the process model from process logs where process events are recorded as they have been executed over time. We induce rules that predict causal, exclusive, and parallel relations between process events. These relation information are used by an already developed $\alpha$-algorithm to construct a process model in the Petri-net formalism. The rules are induced from simulated process log data that are generated by varying process characteristics (e.g. noise, log size). Tests reveal that the induced rules has a high performance on new data.

## 1 Introduction

The managing of complex business processes calls for the development of powerful information systems, able to control and support the underlying process. In order to support a structured business process, an information system has to offer generic modelling capabilities. However, many problems are encountered when designing and employing information systems. One of the problems is that these systems presuppose the existence of the process design, i.e. a designer has to construct a detailed model accurately describing the whole process. The drawback of such an approach is that the process model designing requires considerable effort from the process designers, workers and management, is time consuming and often subjective and incomplete.

As an alternative to hand-designing the process, we propose to collect the sequence of events produced over time by that process, and discover the underlying process model from these sequences. We assume that it is possible to record events such that (i) each event refers to a task, (ii) each event refers to a case (i.e. process instance) and (iii) events are totally ordered. We call a set of such recorded sequences the *process log*. We call the method of distilling a structured process description from a process log *process discovery* or *process mining* [1].

To illustrate the idea of process discovery, consider the process log from Table 1. In this example, there are seven cases that have been processed; twelve different tasks occur in these cases.

**Table 1.** A process log example

| Case number | Executed tasks |
|---|---|
| Case 1 | a f g h i k l |
| Case 2 | a b c e j l |
| Case 3 | a f h g i k l |
| Case 4 | a f g i h k l |
| Case 5 | a b c e j l |
| Case 6 | a b d j l |
| Case 7 | a b c e j l |

Using the information shown in Table 1, we try to discover the process model shown in Figure 1. We represent the model using the Petri nets formalism [13]. After executing *a*, either task *b* or task *f* can be executed. If task *f* is executed, tasks *h* and *g* can be executed in parallel. A parallel execution of tasks *h* and *g* means that they can appear in any order. In this simple example, the construction of the Petri net was straightforward. However, in the case of real-world processes where much more tasks are involved and with a high level of parallelism, the problem of discovering the underlying process becomes very complex.

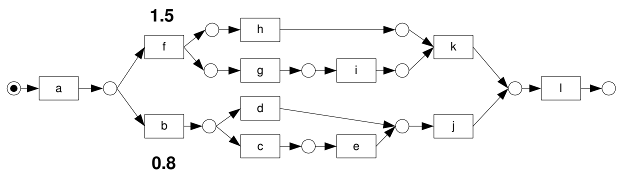

**Fig. 1.** A process model for the process log shown in Table 1

The idea of discovering models from process logs was previously investigated in contexts such as software engineering processes and workflow management [3], [5], [9], etc. Cook and Wolf propose different methods for process discovery in the case of sequential [5] and concurrent [6] software engineering processes. Herbst and Karagiannis used a hidden Markov model in the context of workflow management, in the case of sequential processes [7, 9] and concurrent processes [8]. In [10], a technique for discovering the underlying process from hospital data is presented, under the assumption that the workflow log does not contain any noisy data. A heuristic method that can handle noise is presented in [14]; however, in some situations, the used metric is not robust enough for discovering the complete process. Theoretical results are presented in [1], being proven that for certain subclasses it is possible to find the right process model. In [2] the method used in [1] is extended to incorporate timing information. In [11], a logistic regression model that uses a global threshold has been developed to detect the direct successors from a process logs, in the presence of noise and imbalance of task execution probabilities.

In this paper we try to define some useful log-based relations and to use inductive learning techniques on them to induce rule sets that can be used for finding the causal, parallel and exclusive relations between tasks. The problem of process discovery from process logs is defined as a three-step method: (i) find the causal relations (i.e., for each task, find its direct successor tasks), (ii) find the parallel/exclusive relations (i.e. for tasks that share the same cause or the same direct successor, detect if they can be executed in parallel or there is a choice between them) and (iii) use the $\alpha$ algorithm (introduced in [1]) to build the Petri net. We aim to use an experimental approach for inducing the rule sets required in the three-step method.

In practical situations it seems realistic to assume that process logs contain noise. Noise can have different causes, such as missing registration data or input errors. Moreover, the log can be incomplete. In case of a complex process, the process log will contain not enough information to detect all causal relations between tasks. Another source of problems is the existence of imbalances between the task execution frequency. If the frequency of some tasks is very low, there is a higher chance that the log does not contain enough information about these tasks. Given the fact that in practical situations the process logs are incomplete, contain noise and can exist imbalances between the task execution priorities, the discovery problem becomes more problematic.

The content of this paper is organized as follows: in Section 2 the types of relations that can exist between two tasks are presented. The methodology of generating experimental data that serves to induce the rule sets is presented in Section 3. In Section 4 the metrics used to induce the rule set are introduced. Section 5 contains the description of the three-step process discovery method. In Section 6 we discuss the results obtained. We end with conclusions and issues for further research.

## 2 The log-based relations

Our method of discovering the process model from a log file starts with finding the relations that can exist between tasks. For example, if a task is always followed by another task, it is likely that there is a causal relation between both tasks. In order to find these relations, we use a so-called dependency/frequency (D/F) table [14].

**The dependency/frequency table.** An excerpt from the D/F table for a process log larger than the log presented in Table 1 is shown in Table 2 (this process log contains traces for 1800 cases, not only for 7 cases). For each pair of tasks $x$ and $y$, the following information is abstracted out of the process log: (i) the identifiers for tasks $x$ and $y$, (ii) the overall frequency of task $x$ (notation $|X|$ [1]), (iii) the overall frequency of task $y$ $|Y|$, (iv) the frequency of task $x$ directly preceded by another task $y$ $|Y > X|$, (v) the frequency of task $x$ directly succeeded by another task $y$ $|X > Y|$, (vi) the frequency of $x$ directly or indirectly preceded by another task $y$, but before the next appearance of $x$ $|Y >_n X|$, (vii) the frequency of $x$ directly or indirectly succeeded by another task $y$, but before the next appearance of $x$ $|X >_n Y|$. The frequencies (ii)-(vii) are used to find the log-based ordering relations, which are presented in Section 4. In order to discover the log-based relations, we have to introduce their definitions.

**Table 2.** An excerpt from the D/F table

| $x$ | $y$ | $\|X\|$ | $\|Y\|$ | $\|Y > X\|$ | $\|X > Y\|$ | $\|Y >_n X\|$ | $\|X >_n Y\|$ |
|---|---|---|---|---|---|---|---|
| a | f | 1800 | 850 | 0 | 850 | 0 | 850 |
| f | g | 850 | 850 | 0 | 438 | 0 | 850 |
| c | d | 446 | 504 | 0 | 0 | 0 | 0 |
| g | h | 850 | 850 | 412 | 226 | 412 | 438 |
| b | f | 950 | 850 | 0 | 0 | 0 | 0 |
| i | h | 850 | 850 | 226 | 212 | 638 | 212 |

**Definition 1** ***(Succession relation)*** *Let $W$ be a process log over $T$, i.e. $W \in \mathcal{P}(T^*)$. Let $a$, $b \in T$. Then between $a$ and $b$ there is a* ***succession*** *relation (notation $a > b$), i.e. $b$ succeeds $a$ if and only if there is a trace $\delta = t_1 t_2 ... t_n$ and $i \in \{1, ..., n-1\}$ such that $\delta \in W$ and $t_i = a$ and $t_{i+1} = b$.*

The relation $>$ describes which tasks appeared in sequence, one directly following the other. In the log from Table 1, $a > f$, $f > g$, $h > g$, $g > h$, etc.

**Definition 2** ***(Causal, exclusive and parallel relations)*** *Let $W$ be a process log over $T$, i.e. $W \in \mathcal{P}(T^*)$ and $a, b \in T$ . If we assume that there is no noise in $W$, then between $x$ and $y$ there is:*

1. *a* ***causal*** *relation (notation $x \rightarrow y$), i.e. $x$* ***causes*** *$y$ if and only if $x > y$ and $y \not> x$.*
2. *an* ***exclusive*** *relation (notation $x \# y$) if and only if $x \not> y$ and $y \not> x$;*
3. *a* ***parallel*** *relation (notation $x \parallel y$) if $x > y$ and $y > x$.*

[1] We use a capital letter between | | when referring to the number of occurrences of some task.

We call task $x$ the *cause* of task $y$ and task $y$ is the *direct successor* of task $x$. We consider the inverse of the causal relation $\rightarrow^{-1}$, i.e. $\rightarrow^{-1} = \{(y,x) \in T \times T \mid x \rightarrow y\}$. The relations $\rightarrow, \rightarrow^{-1}, \#$ and $\parallel$ are mutually exclusive and partition $T \times T$ [1].

To illustrate the above definitions, let's consider again the process log from Table 1 corresponding to the Petri net from Figure 1. If there is no noise, there are three possible situations in which a pair of events can be: (i) events $c$ and $e$ are in sequence, then $c > e$, $e \not> c$, i.e., $c \rightarrow e$, (ii) there is a choice between events $b$ and $f$, then $b \not> f, f \not> b$, i.e., $b\#f$ (and $f\#b$) and (iii) events $h$ and $i$ are in parallel, then $h > i$, $i > h$, i.e., $h \parallel i$ (and $i \parallel h$).

However, in case of noise, the notions presented in Definition 2 can conduce to wrong conclusions. If we want to investigate the relation between $c$ and $e$, we find that $c > e$. However, because of some noisy sequences, we may see also that $e > c$. Applying Definition 2, we could conclude that events $c$ and $e$ are parallel, which is wrong, because they are actually in a causal relation. Similarly, looking at events $b$ and $f$, it can happen that $b > f$ and $f > b$, because of noise. Investigating the relation between $h$ and $i$, we can see that $h > i$ and $i > h$, in situations with and without noise.

Suppose now that we are aware of the existence of noise in a process log (which is a realistic assumption) and for two generic tasks $x$ and $y$ we have $x > y$ and $y > x$. What is the relation between $x$ and $y$: causal, exclusive or parallel?

In the rest of our paper we plan to induce decision rules that are used to detect the relations between events, from noisy process logs. Next, we can construct the Petri net process model, using the $\alpha$ algorithm described in [1]. This algorithm considers first all tasks that stand in a causal relation. Then for all tasks that share locally the same input (or output) task, the exclusive/parallel relations are included to build the Petri net. Based on the choice for the $\alpha$ algorithm to build the Petri net, we plan to develop a method that adopts its sequence of actions: first detects the causal relations, second, determines the exclusive/parallel relations for all tasks that share the same local input (or output) task and third, builds the Petri net.

In order to induce such decision rule sets, we use an artificial simulated learning material, as described in the next section.

## 3 Experimental setting and data generation

The learning material that we use to induce the rule sets should resemble realistic process logs. Of the possible elements that vary from process to process and subsequently affect the process log, we identified four: (i) the total number of events types, (ii) the amount of available information in the process log, (iii) the amount of noise and (iv) the execution priorities in OR-splits and AND-splits.

We generate Petri nets with 12, 22, 32 and 42 event types. The *amount of information* in the process log or log size is expressed by varying the number of lines (one line or trace represents the processing of one case). We consider logs with 200, 400, 600, 800 and 1000 lines. To vary the *amount of noise*, we generate noise performing four different operations, (i) delete the head of a event sequence, (ii) delete the tail of a sequence, (iii) delete a part of the body and (iv) interchange two randomly chosen events. We select 5%, 10%, 20% and respectively 50% of the original event sequences and we apply one of the four above described noise generation operations. We assume that tasks can be executed with *imbalanced execution priorities* between 0 and 2. In Figure 1, after executing the event $a$ (which is an OR-split), it is possible to exist an imbalance between executing task $b$ and task $f$. For example, task $b$ can have an execution priority of 0.8 and task $f$ 1.5. This implies that after $a$, in 35 percent of the cases task $b$ is selected (because $0.8/(0.8+1.5)*100 \simeq 35$), and in 65 percent ($1.5/2.3*100 \simeq 65$) of the cases, task $f$ is executed. The execution imbalance is produced at four levels, i.e., no imbalance, small, medium and high imbalance.

Varying the amount of information, the amount of noise and unbalancing the Petri nets with 12, 22, 32 and 42 event types, we end up with 400 log files. For each of this log file, a D/F table is built and finally all the 400 D/F tables are combined into one big file used to induce the rule sets for detecting the relations between tasks. In the next section we see how the information contained in the D/F table is used to detect the log-based relations.

## 4 The relational metrics

Based on the information in the D/F table, we develop three derived measures to predict the causal relations, i.e., the causality metric $CM$, the local metric $LM$ and the global metric $GM$ and two measures to predict exclusive/parallel relations, i.e., $YX$ and $XY$.

**The causality metric CM.** The causality metric $CM$ was first introduced in [14]. If for a given process log it is true that when task $x$ occurs, shortly later task $y$ also occurs, it is possible that task $x$ causes the occurrence of task $y$. The $CM$ metric is computed as follows: if task $y$ occurs after task $x$ and $n$ is the number of events between $x$ and $y$, then $CM$ is incremented with a factor $(\delta)^n$, where $\delta$ is a causality factor, $\delta \in [0.0, 1.0]$. We set $\delta = 0.8$. The contribution to $CM$ is maximally 1, if task $y$ appears right after task $x$ and consequently $n = 0$. Conversely, if task $x$ occurs after task $y$ and again the number of events between $x$ and $y$ is $n$, $CM$ is decreased with $(\delta)^n$. After processing the whole log, $CM$ is divided with the minimum of the overall frequency of $x$ and $y$.

**The local metric LM.** Considering tasks $x$ and $y$, the local metric $LM$ is expressing the tendency of the succession relation $x > y$ by comparing the magnitude of $|X > Y|$ versus $|Y > X|$. The formula for the local metric $LM$ is $LM = P - 1.96\sqrt{\frac{P(1-P)}{N+1}}$, where $P = \frac{|X>Y|}{N+1}$ and $N = |X > Y| + |Y > X|$. The idea of this measure is borrowed from statistics and it is used to calculate the confidence intervals for errors. For more details, see [12]. In our case, we are interested to know with a probability of 95% the likelihood of causality relation, by comparing the magnitude of $|X > Y|$ versus $|Y > X|$. For example, if $|A > B| = 30$, $|B > A| = 1$ and $|A > C| = 60$, $|C > A| = 2$, what is the most likely: $a$ causes $b$ or $a$ causes $c$? Although both ratios $\frac{|A>B|}{|B>A|}$ and $\frac{|A>C|}{|C>A|}$ equal 30, $a$ is more likely to cause $c$ than $b$. Our $LM$ measure for tasks $a$ and $b$ gives a value of $LM = 0.85$ and for tasks $a$ and $c$ gives a value of $LM = 0.90$, which is in line with our intuition.

Let's now consider again the Petri net from Figure 1. If we suppose that the number of lines in the log corresponding to this Petri net is equal to 1000 (i.e. $\#L$=1000), we can have the following three situations: (i) $|C > E|$=1000, $|E > C|$=0, $LM$=0.997, (ii) $|H > G|$=600, $|G > H|$=400, $LM$=0.569, (iii) $|F > B|$=0, $|B > F|$=0, $LM$=0. In the sequential case (situation (i)), because $e$ always succeeds $c$, $LM \cong 1$. When $h$ and $g$ are in parallel, in situation (ii), $LM = 0.569$, i.e. a value much smaller than 1. In the case of choice between $f$ and $b$, in situation (iii), $LM = 0$. In general, we can conclude that the $LM$ measure has a value close to 1 when there is a clear tendency of causality between tasks $x$ and $y$. When the $LM$ measure is close to 0, there is no causality relation between tasks $x$ and $y$. When the $LM$ measure has a value close to 0.5, then $x > y$ and $y > x$, but a clear tendency of causality cannot be identified.

**The global metric GM.** The previous measure $LM$ was expressing the succession tendency by comparing the magnitude of $|X > Y|$ versus $|Y > X|$ at a local level. Let us now consider that the number of lines in our log is $\#L$=1000 and the frequencies of tasks $a$, $b$ and $c$ are $|A|$=1000, $|B|$=1000 and $|C|$=1000. We also know that $|A > B| = 900, |B > A| = 0$ and $|A > C| = 50$ and $|C > A| = 0$. The question is: $a$ is the most likely cause of $b$ or $c$ or both? For $a$ causes $b$, $LM = 0.996$ and for $a$ causes $c$, $LM = 0.942$, so we can conclude that $a$ causes both $b$ and $c$. However, one can argue that $c$ succeeds $a$ less frequently, thus $a$ should be considered the cause of $b$. Therefore, we build a second measure, the global metric $GM$, defined as $GM = ((A > B) - (B > A))\frac{\#L}{(A)*(B)}$. The value for the $GM$ in case of $a$ causes $b$ is $GM = 0.90$ and for $a$ causes $c$, $GM = 0.05$.

In conclusion, for determining the likelihood of causality between two events $x$ and $y$, the $GM$ metric is indeed a global metric because it takes into account the overall frequencies of tasks $x$ and $y$, while the $LM$ metric is a local metric because it only takes into account the magnitude of $|X > Y|$ versus $|Y > X|$.

**The $YX$ and $XY$ metrics.** $|X > Y|$ and $|Y > X|$ frequencies can be also used to decide between exclusive and parallel relations. When between $x$ and $y$ there is an exclusive relation, both $|X > Y|$ and $|Y > X|$ frequencies should be zero or a small value, while for the parallel case both should be relatively high. Because the rule set that will be induced using these metrics as predictors must be general, we have to take into account also the frequencies of tasks $x$ and $y$. Therefore we divide $|X > Y|$ and $|Y > X|$ with the minimum of $|X|$ and $|Y|$. Thus, $YX$ and $XY$ are defined as $YX = |Y > X|/min\{|X|, |Y|\}$ and $XY = |X > Y|/min\{|X|, |Y|\}$. In Table 3 the values for the relational metrics of some task pairs for a process log (with similar traces as the log presented in Table 1, containing traces for 1800 cases) are presented.

## 5 The three-step process discovery method

We aim to develop a three-step method for discovering process models. This method (i) determines the causal relations, then (ii) determines the parallel/exclusive relations and (iii) use the $\alpha$ algorithm (introduced in [1]) to build the Petri net.

**Step1: Determining causal relations.** In Section 2 we introduced five relational metrics $CM$, $GM$, $LM$, $YX$ and $XY$ to be used in determining the causal and exclusive/parallel relations. The idea is to use the learning material generated in Section 3 and to compute for each data set the relational metrics.

We are interested in a fast and efficient algorithm and we want to obtain a model that can be easily understood. Ripper [4] is an algorithm that induces rule sets and seems to meet our requirements.

For inducing a rule set, we have to provide a set of examples, each of which has been labelled with a *class*. We are interested to induce rule sets for detecting the log-based relations, therefore our examples are classified as "c" (if $a \rightarrow b$), "e" (if $a\#b$) and "p" (if $a \parallel b$). As we mentioned before, we start with searching for rules that detect the "c" relation.

An excerpt of the table with the class labelling is presented in Table 3. Note the pairs ($c$,$d$) and ($g$,$h$) which are labelled in Step 1 with an "n" (in the first step they are used as non-causal examples), while in Step 2 they are labelled "e" and "p" respectively, being selected to induce rules that distinguish between the exclusive and the parallel relation.

**Table 3.** Excerpt from the learning materials used to induce the rule set for detecting in Step 1 the causal relations and in Step 2, the exclusive/parallel relations, from a log with similar traces as the log presented in Table 1

| Step | $x$ | $y$ | $CM$ | $GM$ | $LM$ | $YX$ | $XY$ | $Rel$ |
|---|---|---|---|---|---|---|---|---|
| 1 | a | f | 1.000 | 1.000 | 0.998 | 0.000 | 1.000 | c |
| 1 | a | b | 1.000 | 1.000 | 0.998 | 0.000 | 1.000 | c |
| 1 | f | g | 0.903 | 1.091 | 0.996 | 0.000 | 0.515 | c |
| 1 | f | h | 0.857 | 1.026 | 0.995 | 0.000 | 0.485 | c |
| 1 | b | a | -1.000 | -1.000 | 0.000 | 1.000 | 0.000 | n |
| 1 | c | d | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | n |
| 1 | g | h | -0.019 | -0.436 | 0.317 | 0.485 | 0.266 | n |
| 2 | b | f | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | e |
| 2 | c | d | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | e |
| 2 | g | h | -0.019 | -0.436 | 0.317 | 0.485 | 0.266 | p |

To obtain the rule set for detecting the causal relations, we use only the instances labelled with "c" or "n". Using Ripper, we obtain 33 ordered rules for class "c" ("n" is the default class); we refer this rule set as RIPPER_CAUS. The training error rate for RIPPER_CAUS is 0.08% (the training error rate represents the rate of incorrect predictions made by the model over the training data set). Below is presented a selection of rules that have a coverage higher than 100 positive instances and less than 7 negative instances. We can remark that these rules cover quite a lot of positive instances and have few negative counterexamples.

```
Rule1: IF LM>=0.949 AND XY>=0.081 THEN class c [10797 pos, 0 neg]
Rule2: IF LM>=0.865 AND YX=0 AND GM>=0.224 THEN class c [1928 pos, 6 neg]
Rule3: IF LM>=0.844 AND CM>=0.214, CM<=0.438 THEN class c [525 pos, 1 neg]
Rule4: IF LM>=0.741 AND GM>=0.136 AND YX<=0.009 AND
CM>=0.267 AND CM<=0.59 THEN class c [337 pos, 0 neg]
Rule5: IF XY>=0.6 AND CM<=0.827 THEN class c [536 pos, 0 neg]
Rule6: IF LM>=0.702 AND YX<=0.009 AND GM>=0.36 THEN class c [273 pos, 0 neg]
Rule7: IF LM>=0.812 AND CM<=0.96 AND GM>=0.461 THEN class c [142 pos, 0 neg]
```

Let us interpret these rules. Rule1 has the highest coverage of positive examples, i.e. almost 70% of “c” instances match this rule. E.g., if the $LM$ measure has a very high value (i.e. there is a big difference in magnitude between $|X > Y|$ and $|Y > X|$ frequencies) and the $XY$ measure is exceeding a small value, there is a high chance there to be a causal relation between $x$ and $y$. The first condition of Rule2 specifies $LM$ to be high, while the second requires the global measure $GM$ to exceed 0.2 (i.e., the difference between $|X > Y|$ and $|Y > X|$ frequencies accounted by the overall frequencies of $x$ and $y$ should be sufficiently high). The third condition specify that the $YX$ measure must be 0, i.e. $|Y > X| = 0$. In general, the rules require the $LM$ measure to exceed a high value, $YX$ to be a value close to zero, while $XY$ should be bigger than 0. Also, $CM$ and $GM$ measures should be sufficient large.

**Step 2: Detecting exclusive/parallel relations.** In order to induce the second rule set for detecting the exclusive/parallel relations, from the whole material generated in Section 3, we select only the pairs of tasks which share the same cause or the same direct successor task. In Table 3, at Step 2, the pairs of tasks in exclusive and parallel relations and the corresponding relational measures are shown. We see that tasks $g$ and $h$ have as same common cause the task $f$ and tasks $b$ and $f$ have as same common cause the tasks $a$. The pairs in exclusive relation are labelled with “e” (e.g. the pair of tasks $(b, f)$) and those in parallel relations with “p” (e.g. the pair $(g, h)$). We induce the second rule set for detecting exclusive and parallel relations. We obtain the RIPPER_ANDOR rule set with 15 unordered rules, 7 for class “e” and 8 for class “p”, with the training error rate 0.38%. Doe to space limitations, we do not show the rule set for detecting exclusive/parallel relations.

***Rule sets evaluation.*** To check how well our rule sets will generalize to new data, we use the well-known $k$-fold cross validation technique (with $k$=10). In order to compare the performance of the 10 obtained models, we compare three averaged performance indicators: the error rate, precision and recall. In the case of identifying the relations between tasks, we are interested to see an aggregate of the cost of false positives and false negatives, expressed in terms of recall and precision. In case of causal relations, false positives are false causal relations found, i.e. linking tasks which are not causally related. False negative are actual causal relations that are omitted from the Petri net. Asserting that precision and recall are equally important, we use the combined F-measure, $F = \frac{2*TP}{2*TP+FP+FN}$. $TP$ are class members classified as class members, $FP$ are class non-members classified as class members and $FN$ are class members classified as class non-members.

We check the performance of a model based on predicted data, i.e., we use the first rule set, RIPPER_CAUS, to predict the causal relations. From this new learning material, we select the task pairs that share a common cause or a common direct successor and we induce with Ripper a new rule set that detects exclusive/parallel relations. The 10-fold averaged error rate of this new second rule set is 0.36% and the averaged F-measure for "e" and "p" classes is 99.83 and 99.85, respectively.

Based on the 10-fold cross validations experiments, we can say that both rule sets (i.e. the rule set that detects causal relations and the rule set that detects exclusive/parallel relations) seem to have a high performance on new data.

**Step 3: Building the Petri net.** The basic idea of the $\alpha$ algorithm [1] is to connect (i) all events $a \rightarrow b$ and to add a place between $a$ and $b$ and (ii) to merge those places if $a\#b$. For the second step, we use the relations $a\#b$ and $a \parallel b$. To illustrate the $\alpha$ algorithm, we consider the log presented in Table 1. Applying the rule set for detecting causal relations on the log information presented in Table 1, resulted the following causal relations: $a \rightarrow f$, $a \rightarrow b$, $f \rightarrow g$, $f \rightarrow h$, $b \rightarrow d$, $b \rightarrow c$, $g \rightarrow i$, $h \rightarrow k$, $i \rightarrow k$, $c \rightarrow e$, $e \rightarrow j$, $d \rightarrow j$, $k \rightarrow l$, $j \rightarrow l$. According to our algorithm, we add on each arc one place (represented as a small circle), as shown in Figure 2. The next step is to apply the rule set for detecting exclusive/parallel relations and merge those places whenever $a\#b$. Thus, we have to perform four merge tasks, i.e. to merge: the two places from event $a$ to $f$ and from $a$ to $b$, $(b\#f)$, the two places from event $b$ to $d$ and from $b$ to $c$, $(c\#d)$, the two places from event $d$ to $j$ and from $e$ to $j$, $(d\#e)$ and the two places from event $k$ to $l$ and from $j$ to $l$, $(k\#j)$. In Figure 2, these places that need to be merged are marked by bold dotted circles. After merging, we recover the Petri net from Figure 1.

## 6 Discussion

Finding the causal, exclusive and parallel relations with our method does not necessarily result in Petri nets equivalent with the original Petri nets used to generate the learning material. It was already formally proven which class of Petri nets it is possible to rediscover the original net, assuming log completeness and no noise in the process log [1]. The method presented in this paper pro-

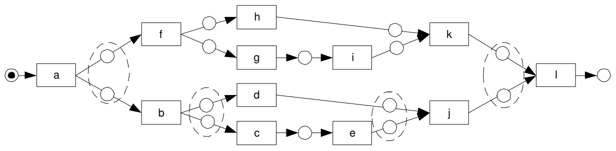

**Fig. 2.** The directed graph that contains the events in relation $a \rightarrow_W b$, after adding places.

vides a solution to construct the Petri net model from a process log when the log is incomplete and noisy. However, the degree of incompleteness and noise is affecting in a certain extent the quality of the discovered process model. Namely, more noise, less balance and less cases, each have a negative effect on the quality of the results. Our experiments show that causal relations can be predicted more accurately if there is less noise, more balance and more cases. There is no clear evidence that the number of event types has an influence on the performance of predicting causal relations. However, causal relations in a structurally complex Petri net can be more difficult to detect. Because the detection of exclusive/parallel relations depends on the detection of the causal relations, it is difficult to formulate specific conclusions for the quality of exclusive/parallel relations. It appears that noise is affecting exclusive and parallel relations in a similar way as the causal relations, e.g., if the level of noise is increasing, the accuracy of finding the excusive/parallel relations is decreasing.

When discovering real process data, the above conclusions can play the role of useful recommendations. Usually it is difficult to know the level of noise and imbalance beforehand. However, during the discovery process it is possible to collect data about these metrics. This information can be used to motivate additional efforts to collect more data.

## 7 Conclusions and future directions

Based on artificial experimental data, where the number of event types, noise, execution imbalance and log size are varied, we developed a three-step method that discovers the underlying process from a process log. In the first step, the method employs a rule set to detect the causal relations; after the causal relations are found, the second rule set detects the exclusive/parallel relations between tasks that share the same cause or the same direct successor. In the third step, knowing the causal and exclusive/parallel relations, the Petri net is built to obtain the process model. Our three-step method has a very high performance in classifying new data, being able to find almost all relations in the presence of parallelism, imbalance and noise.

The causal relations can be predicted more accurately if there is less noise, more balance and more cases. However, causal relations in a structurally complex Petri net can be more difficult to detect. The current experimental setting shows that noise, imbalance and log size are factors that affect the quality of the discovered model. We plan as future work to perform real-world case studies and to adapt our method by considering other possible factors that may influence the characteristics of the process logs.

## References

1. Aalst, W.M. P., Weijters, A.J.M. M., Maruster., L.: Workflow Mining: Which Processes can be Rediscovered? BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven, 2002.
2. Aalst, W.M. P., Dongen, B. F.: Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, *International Conf. on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, volume 2480 of Lecture Notes in Computer Science, pages 45-63, Springer-Verlag, Berlin, 2002.
3. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process models from Workflow Logs. In Sixth International Conference on Extended Database Technology. In Sixth International Conf. on Extended Database Technology, pg. 469–483, 1998.
4. Cohen, W. W.: Fast Effective Rule Induction. Proc. of the Twelfth Int. Conf. of Machine Learning ICML95, 1995.
5. Cook, J. E., Wolf, A. L.: Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology, 7(3):215-249, 1998.
6. Cook, J. E., Wolf, A. L.: Event-Based Detection of Concurrency. In Proc. of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6), Orlando, FL, pp. 35-45, November, 1998.
7. Herbst, J.: A Machine Learning Approach to Workflow Management. In 11th European Conf. on Machine Learning, volume 1810 of Lecture Notes in Computer Science, Springer, Berlin, Germany, pp. 183-194, 2000.
8. Herbst, J.: Dealing with Concurrency in Workflow Induction. In U. Baake, R. Zobel, and M. Al-Akaidi, editors, European Concurrent Engineering Conf.. SCS Europe, 2000.
9. Herbst. J, Karagiannis, D.: Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. Int. Journal of Intelligent Systems in Accounting, Finance and Management, 9:67–92, 2000.
10. Maruster, L., Aalst, W.M. P, Weijters, A.J.M. M, Bosch, A., Daelemans, W.: Automated Discovery of Workflow Models from Hospital Data. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, Proc. of the 13th Belgium-Netherlands Conf. on Artificial Intelligence (BNAIC 2001), pages 183–190, 2001.
11. Maruster, L., Weijters, A.J.M. M, Aalst, W.M. P, Bosch, A.: Process Mining: Discovering Direct Successors in Process Logs. In Proc. of the 5th International Conf. on Discovery Science (Discovery Science 2002), Lecture Notes in Computer Science 2534, S. Lange, K. Satoh, C. Smith (Eds.), pages 364-373, Springer-Verlag, Berlin, 2002.
12. Mitchell, T.: Machine Learning. Mc-GrawHill, 1995.
13. Reisig, W., Rosenberg, G. (eds.): Lectures on Petri nets II. Basic models, Springer 1998.
14. Weijters, A.J.M.M., Aalst, W.M.P.: Process Mining: Discovering Workflow Models from Event-Based Data. In B. Kröse et al. (eds.), Proc. of the 13th Belgium-Netherlands Conf. on Artificial Intelligence (BNAIC 2001), pages 283–290, 2001.