Decision Mining in Business Processes

A. Rozinat and W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{a.rozinat,w.m.p.v.d.aalst}@tm.tue.nl

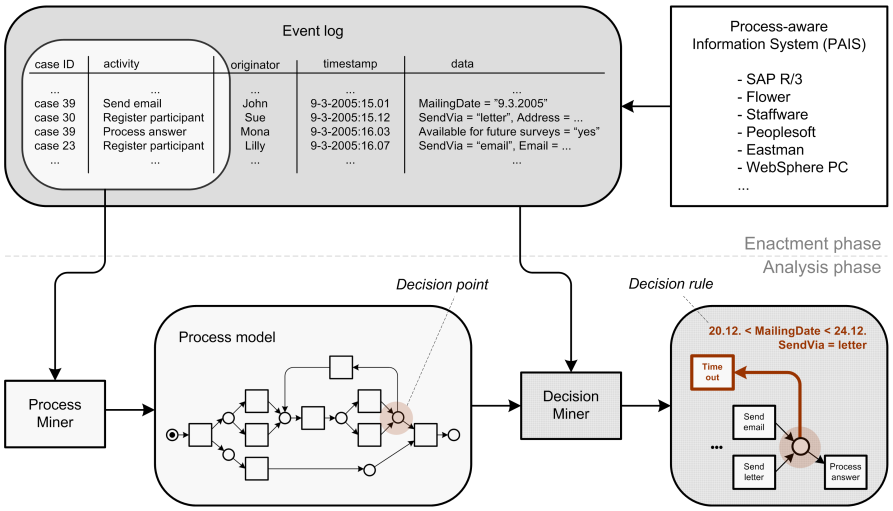
Abstract. Many companies have adopted Process-aware Information Systems (PAIS) for supporting their business processes in some form. These systems typically log events (e.g., in transaction logs or audit trails) related to the actual business process executions. Proper analysis of PAIS execution logs can yield important knowledge and help organizations improve the quality of their services. Starting from a process model as it is possible to discover by conventional process mining algorithms we analyze how data attributes influence the choices made in the process based on past process executions. Decision mining, also referred to as decision point analysis, aims at the detection of data dependencies that affect the routing of a case. In this paper we describe how machine learning techniques can be leveraged for this purpose, and discuss further challenges related to this approach. To verify the presented ideas a *Decision Miner* has been implemented within the ProM framework.

Keywords: Business Process Intelligence, Process Mining, Petri Nets, Decision Trees.

1 Introduction

Process mining techniques have proven to be a valuable tool in order to gain insight into how business processes are handled within organizations. Taking a set of real process executions (the so-called "event logs") as the starting point, these techniques can be used for *process discovery* and *conformance checking*. Process discovery [2, 3] can be used to automatically construct a process model reflecting the behavior that has been observed and recorded in the event log. Conformance checking [1, 11] can be used to compare the recorded behavior with some already existing process model to detect possible deviations. Both may serve as *input* for a design and improvement of business processes, e.g., conformance checking can be used to find problems in existing processes and process discovery can be used as a starting point for process analysis and system configuration. While there are several process mining algorithms dealing with the control flow perspective of a business process [2] *less attention has been paid to how the value of a data attribute may affect the routing of a case.*

Most information systems, cf. WFM, ERP, CRM, SCM, and B2B systems, provide some kind of *event log* (also referred to as transaction log or audit trail) [2] where an event refers to a case (i.e., process instance) and an activity, and, in most systems, also a timestamp, a performer, and some additional data. Nevertheless, many process mining techniques only make use of the first two attributes in order to construct a process model which reflects the causal dependencies that have been observed among the activities. In addition, machine learning algorithms have become a widely adopted means to extract knowledge from vast amounts of data [9, 13]. In this paper we explore the potential of such techniques in order to gain insight into the data perspective of business processes. The well-known concept of *decision trees* will be used to carry out a *decision point analysis*, i.e., to find out which properties of a case might lead to taking certain paths in the process. Starting from a discovered process model (i.e., a model discovered by conventional process mining algorithms), we try to enhance the model by integrating patterns that can be observed from data modifications, i.e., every choice in the model is analyzed and, if possible, linked to properties of individual cases and activities.



Event log

case ID	activity	originator	timestamp	data
...	...	...	...	...
case 39	Send email	John	9-3-2005:15.01	MailingDate = ”9.3.2005”
case 30	Register participant	Sue	9-3-2005:15.12	SendVia = “letter”, Address = ...
case 39	Process answer	Mona	9-3-2005:16.03	Available for future surveys = “yes”
case 23	Register participant	Lilly	9-3-2005:16.07	SendVia = “email”, Email = ...
...	...	...	...	...

Fig. 1. The approach pursued in this paper

Figure 1 illustrates the overall approach. First of all, we assume some *Process-aware Information System* (PAIS) [5] (e.g., a WFM, CRM, ERP, SCM, or B2B system), that records some event log. Note that this PAIS may interact with humans, applications, or services in order to accomplish some business goal. The top-half of Figure 1 sketches the content of such an event log and lists some example systems we have used to obtain this data from. In the context of the ProM framework we have analyzed such event logs from a wide variety of systems in application domains ranging from health care to embedded systems. As Figure 1 shows the event log may contain information about the people executing activities (cf. originator column), the timing of these activities, and the data involved. However, classical process mining approaches (e.g., the α-algorithm [3]) tend to only use the first two columns to obtain some process model. (Note that in Figure 1 only a subset of the whole event log is depicted.) The Petri net shown in the middle of the lower half of Figure 1 illustrates the result of applying the α-algorithm to the event log of some survey process. Many other process mining algorithms could have been used to discover this process. Moreover, the fact that the result is a Petri net is not essential, it could have been any other process model (e.g., an EPC or UML activity diagram). What is essential is that the process mining algorithm identifies decision points. Figure 1 highlights one decision point in the survey process. (There are two additional decision points but for decision mining we focus on one decision point at a time.) The decision point considered is concerned with the choice between a *Timeout* activity that leads to the repetition of certain steps in the process (such as re-sending the survey documents) and a *Process answer* activity that concludes the process. The Petri net shows no information regarding this choice. The goal of *decision mining* is to find "rules" explaining under which circumstances the *Timeout* activity is selected rather than the *Process answer* activity. The result is a rule as shown in Figure 1. This decision rule indicates that survey documents sent by letter to the participant are very often not returned in time, just like documents sent shortly before Christmas. Consequently, an extension of the time limit in these cases could help to reduce mailing expenses.

Clearly, the application of (existing) data mining techniques for detecting frequent patterns in the context of business processes has the potential to gain knowledge about the process, or to make tacit knowledge explicit. Moreover, the type of dependency which may be discovered is very general. Besides data attributes, resource information, and timestamps, even more general quantitative (e.g., key performance indicators like waiting time derived from the log) and qualitative (i.e., desirable or undesirable properties) information could be included in the analysis if available. To directly support data analysis for business processes we have implemented a *Decision Miner* in the context of the ProM framework[1], which offers a wide range of tools related to process mining and process analysis.

The paper is organized as follows. First, Section 2 introduces a simple example process that is used throughout the paper. Then, the use of machine learning techniques in the context of the decision point analysis is discussed in Section 3, and the challenges with respect to this application area are highlighted in Section 4. Section 5 presents the Decision Miner plug-in of the ProM framework. Finally, related work is discussed in Section 6, and the paper concludes by pointing out future research directions.

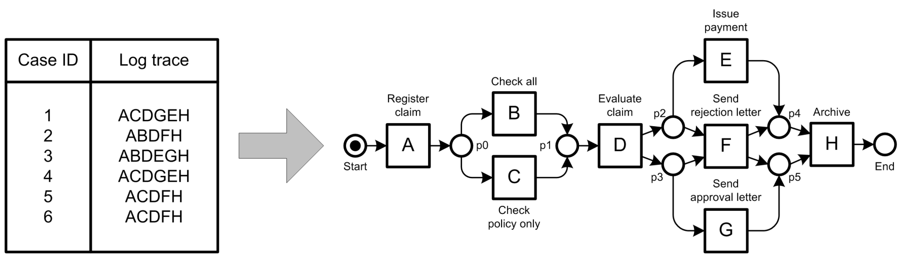
2 Running Example

As depicted in Figure 1, the first step comprises the application of some process mining algorithm in order to obtain a process model. Figure 2(a) shows an event log in a schematic way. It has been grouped by instances (according to the Case ID), and all information except the executed activities has been been discarded. Based on this log the α-algorithm [3] induces the process model shown in Figure 2(b), which in turn serves as input for the decision mining phase.

The example process used throughout the paper sketches the processing of a liability claim within an insurance company: at first there is an activity where data related to the claim gets registered (A), and then either a full check or a policy-only check is performed (B or C). Afterwards the claim is evaluated (D), and then it is either

[1] Both documentation and software (including the source code) can be downloaded from *www.processmining.org*.

rejected (F) or approved and paid (G and E). Finally the case is archived and closed (H)[2].



Case ID	Log trace
1	ACDGEH
2	ABDFH
3	ABDEGH
4	ACDGEH
5	ACDFH
6	ACDFH

(a) Event log (b) Mined process model

Fig. 2. Process mining phase

Entering the second stage of the approach we keep the mined process model in mind and take a closer look at the event log, now also taking data attributes into account. Figure 3 depicts a screenshot of the log in MXML[3] format. One can observe that only activities A and D have data items associated. Here a data item within an audit trail entry is interpreted as a case attribute that has been written, or modified. So during the execution of activity "Register claim" information about the amount of money involved (*Amount*), the corresponding customer (*CustomerID*), and the type of policy (*PolicyType*) are provided, while after handling the activity "Evaluate claim" the outcome of the evaluation is recorded (*Status*). Semantically, *Amount* is a numerical attribute, the *CustomerID* is an attribute which is unique for each customer, and both *PolicyType* and *Status* are enumeration types (being either "normal" or "premium", or either "approved" or "rejected", respectively).

As illustrated in Figure 1 the discovered process model *and* the detailed log are the starting point for the decision mining approach, which is explained in the next section.

3 Using Decision Trees for Analyzing Choices in Business Processes

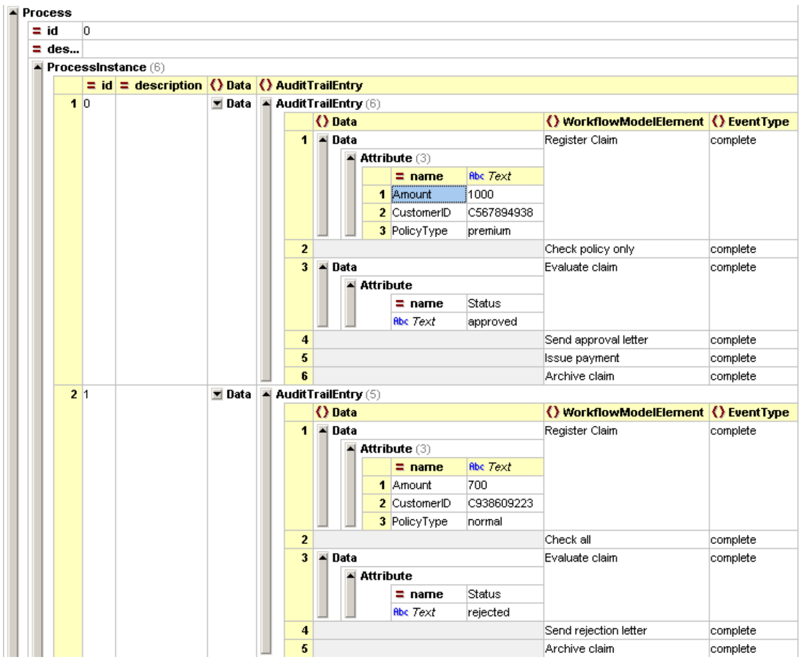
In order to analyze the choices in a business process we first need to identify those parts of the model where the process is split into alternative branches, also called decision points (Section 3.1). Subsequently, we want to find rules for following one way or the other based on data attributes associated to the cases in the event log (Section 3.2).

3.1 Identifying Decision Points in a Process Model

In terms of a Petri net, a decision point corresponds to a place with multiple outgoing arcs. Since a token can only be consumed by one of the transitions connected to these

[2] Note that the letters only serve as a shorthand for the actual activity names.

[3] Both the corresponding schema definition and the ProMimport framework, which converts logs from existing (commercial) PAIS to the XML format used by ProM, can be downloaded from *www.processmining.org*.



```
Process
  = id    0
  = des...
  ProcessInstance (6)
       = id  = description  () Data  () AuditTrailEntry
    1  0                    Data     AuditTrailEntry (6)
                                        () Data                                   () WorkflowModelElement  () EventType
                                     1  Data                                      Register Claim           complete
                                          Attribute (3)
                                               = name      Abc Text
                                            1  Amount      1000
                                            2  CustomerID  C567894938
                                            3  PolicyType  premium
                                     2                                            Check policy only        complete
                                     3  Data                                      Evaluate claim           complete
                                          Attribute
                                            = name    Status
                                            Abc Text  approved
                                     4                                            Send approval letter     complete
                                     5                                            Issue payment            complete
                                     6                                            Archive claim            complete
    2  1                    Data     AuditTrailEntry (5)
                                        () Data                                   () WorkflowModelElement  () EventType
                                     1  Data                                      Register Claim           complete
                                          Attribute (3)
                                               = name      Abc Text
                                            1  Amount      700
                                            2  CustomerID  C938609223
                                            3  PolicyType  normal
                                     2                                            Check all                complete
                                     3  Data                                      Evaluate claim           complete
                                          Attribute
                                            = name    Status
                                            Abc Text  rejected
                                     4                                            Send rejection letter    complete
                                     5                                            Archive claim            complete
```

Fig. 3. Fragment of the example log in MXML format viewed using XML Spy

arcs, alternative paths may be taken during the execution of a process instance. The process model in Figure 2(b) exhibits three such decision points: *p0* (if there is a token, either B or C can be performed), *p2* (if there is a token, either E or F can be executed) and *p3* (if there is a token, either F or G may be carried out).

In order to analyze the choices that have been made in past process executions we need to find out which alternative branch has been taken by a certain process instance. Therefore, the set of possible decisions must be described with respect to the event log. Starting from the identification of a choice construct in the process model a decision can be detected if the execution of the activity in the respective alternative branch of the model has been observed, which requires a mapping from that activity to its "occurrence footprint" in the event log. So if a process instance contains the given "footprint" it means that there was a decision for the associated alternative path in the process. The example model in Figure 2(b) has been mined from the given event log, and therefore all the activity names already correspond to their log event labels. For example, the occurrence of activity "Issue payment" is recorded as "Issue payment" in the log[4], which can be directly used to classify the decision made by that process instance with respect to decision point *p2*. So, for the time being it is sufficient to consider the occurrence of the first activity per alternative branch in order to classify

[4] Note that the two labels match. This is not always the case (e.g., multiple activities may be logged using the same label or activities may not be logged at all). Initially, we assume that the activity label denotes the associated log event but later we will generalize this to also allow for duplicate and invisible activities.

the possible decisions, and we know enough to demonstrate the idea of such decision point analysis. However, in order to make decision mining operational for real-life business processes additional complications such as loops need to be addressed. They are discussed in Section 4.

3.2 Turning a Decision Point into a Learning Problem

Having identified a decision point in a business process we now want to know whether this decision might be influenced by case data, i.e., whether cases with certain properties typically follow a specific route. Machine learning techniques [9] can be used to discover structural patterns in data based on a set of training instances. For example, there may be some training instances that either do or do not represent a table, accompanied by a number of attributes like height, width, and number of legs. Based on these training instances a machine learning algorithm can "learn" the *concept* table, i.e., to classify unknown instances as being a table or not based on their attribute values. The structural pattern inferred for such a classification problem is called a *concept description*, and may be represented, e.g., in terms of rules or a decision tree (depending on the algorithm applied). Although such a concept description may be used to predict the class of future instances, the main benefit is typically the insight gained into attribute dependencies which are "explained" by the explicit structural representation.

Using decision point analysis we can extract knowledge about decision rules as shown in Figure 4. Each of the three discovered decision points corresponds to one of the choices in the running example. With respect to decision point *p0* the extensive check (activity B) is only performed if the *amount* is greater than 500 and the *policyType* is "normal", whereas a simpler coverage check (activity C) is sufficient if the *amount* is smaller than or equal to 500, or the *policyType* is "premium" (which may be due to certain guarantees by "premium" member corporations). The two choices at decision point *p2* and *p3* are both guided by the *status* attribute, which is the outcome of the evaluation activity (activity D). In the remainder of this section we describe how these rules can be discovered.

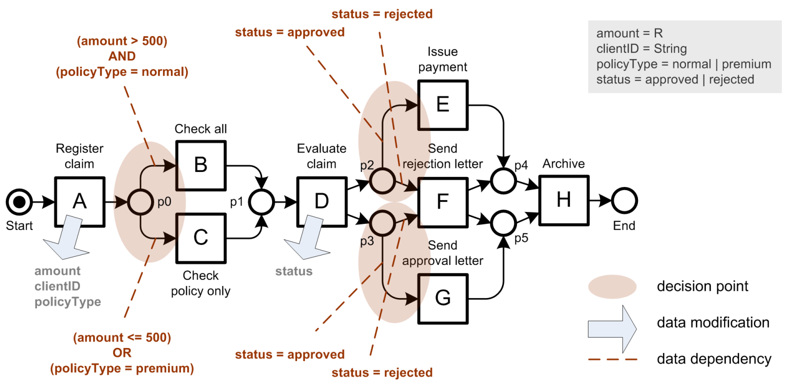
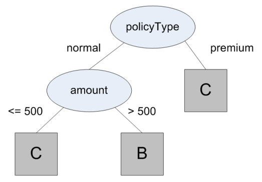


Fig. 4. Enhanced process model

The idea is to convert every decision point into a *classification problem* [9, 13, 10], whereas the *classes* are the different decisions that can be made. As training examples we can use the process instances in the log (for which it is already known which alternative path they have followed with respect to the decision point). The attributes to be analyzed are the case attributes contained in the log, and we assume that all attributes that have been written *before* the considered choice construct may be relevant for the routing of a case at that point. So for decision point *p0* only the data attributes provided by activity *A* are considered (i.e., *amount*, *clientID*, and *policyType*), and in Figure 5(a) the corresponding values contained in the log have been used to build a training example from each process instance (one training example corresponds to one row in the table). The last column represents the (decision) class, which denotes the decision that has been made by the process instance with respect to decision point *p0* (i.e., whether activity *B* or *C* has been executed). Similarly, Figure 5(c) and (e) represent the training examples for decision point *p2* and *p3*, respectively. Here, an additional attribute (i.e., *status*) has been incorporated into the data set because it is provided by activity *D*, which is executed before *p2* and *p3* are reached. Furthermore, the *class* column reflects the decisions made with respect to decision point *p2* and *p3* (i.e., *E* or *F*, and *G* or *F*, respectively).

amount	clientID	policyType	class
1000	C567894938	premium	C
700	C938609223	normal	B
550	C135697567	normal	B
500	C568120443	normal	C
50	C493823084	normal	C
200	C945675110	premium	C

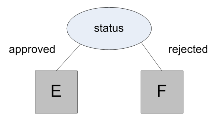
(a) training examples for decision point “p0”



(b) decision tree for decision point “p0”

amount	clientID	policyType	status	class
1000	C567894938	premium	approved	E
700	C938609223	normal	approved	E
550	C135697567	normal	rejected	F
500	C568120443	normal	approved	E
50	C493823084	normal	rejected	F
200	C945675110	premium	rejected	F

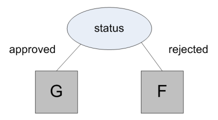
(c) training examples for decision point “p2”



(d) decision tree for decision point “p2”

amount	clientID	policyType	status	class
1000	C567894938	premium	approved	G
700	C938609223	normal	approved	G
550	C135697567	normal	rejected	F
500	C568120443	normal	approved	G
50	C493823084	normal	rejected	F
200	C945675110	premium	rejected	F

(e) training examples for decision point “p3”



(f) decision tree for decision point “p3”

Fig. 5. Decision points represented as classification problems

In order to solve such a classification problem there are various algorithms available [9, 13]. We decided to use decision trees (such as C4.5 [10]), which are among the most popular of inductive inference algorithms and provide a number of extensions that are important for practical applicability. For example, they are able to deal with continuous-valued attributes (such as the *amount* attribute), attributes with many values (such as the *clientID* attribute), attributes with different costs, and missing attribute values. Furthermore, there are effective methods to avoid *overfitting* the data (i.e., that the tree is over-tailored towards the training examples). A decision tree classifies instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Figure 5(b), (d), and (f) show the decision trees that have been derived for decision point *p0*, *p2*, and *p3*, respectively. Because of the limited space we cannot provide further details with respect to the construction of decision trees. But since we rely on existing techniques the interested reader is kindly referred to [9, 13].

From the decision trees shown in Figure 5(b), (d), and (f) we can now infer the logical expressions that form the decision rules depicted in Figure 4 in the following way. If an instance is located in one of the leaf nodes of a decision tree, it fulfills all the predicates on the way from the root to the leaf, i.e., they are connected by a boolean AND operator. For example, class B in Figure 5(b) is chosen if ($policyType =$ "normal") AND ($amount > 500$). When a decision class is represented by multiple leaf nodes in the decision tree the leaf expressions are combined via a boolean OR operator. For example, class C in Figure 5(b) is chosen if (($policyType =$ "normal") AND ($amount \leq 500$)) OR ($policyType =$ "premium"), which can be reduced to ($policyType =$ "premium") OR ($amount \leq 500$).

4 Challenges for Decision Mining in Business Processes

If we want to apply decision mining to real-life business processes, two important challenges need to be addressed.

The *first challenge* relates to the quality of data, and the correct interpretation of their semantics. For example, there might be a loss of data or incorrectly logged events, which is typically referred to as *noise*. The analysis (and thus the applied data mining algorithms) must be sufficiently robust to deal with "noisy" logs. Moreover, the interpretation of a data attribute, e.g., whether it is relevant, what it actually means, in what quantities it is measured etc., still needs human reasoning. In fact, human involvement is inherent to all data mining applications, no matter in what domain. The techniques cannot be put to work until the problem has been formulated (like a concrete classification problem) and learning instances have been provided (properly preprocessed, so that the results are likely to exhibit real patterns of interest). For this reason it will remain a semi-automatic analysis technique, and for a software tool that "intelligently" supports decision mining it is crucial to offer the full range of adjustment parameters to the business analyst. These parameters include tuning parameters with respect to the underlying algorithms (such as the degree of noise, which kind of validation is to be used etc.) and the possibility to include/exclude certain attributes.

The *second challenge* relates to the correct interpretation of the control-flow semantics of a process model when it comes to classifying the decisions that have been made. The example process from Figure 2(b) is rather simple and does not show more advanced issues that must be addressed in order to make decision mining operational for real-life business processes. In fact, providing a correct specification of the possible choices at a decision point, which can be used to classify learning examples, can be quite difficult. In the remainder of this section we highlight problems related to the control-flow semantics of real-life business processes, namely *invisible activities*, *duplicate activities*, and *loops*, and we point out how they can be solved in order to provide proper decision mining support through a software tool.

As a first step, we need to elaborate on the mapping of an activity in the process model onto its corresponding "occurrence footprint" in the log. This mapping is provided by the labeling function l, which is defined as follows.

Definition 1 (Labeling Function). *Let T be the set of activities in a process model and L a set of log events, then $l \in T \not\rightarrow L$ is a partial labeling function associating each activity with at most one (i.e., either zero or one) log event.*

As stated in Section 3.1, with respect to the example model in Figure 2(b) all the activity names already correspond to their log event labels, and no two activities have the same label. Furthermore, there is no activity in the model that has no log event associated. However, real-life process models may contain activities that have no correspondence in the log, e.g., activities added for routing purposes only. These activities are called *invisible activities*.

Definition 2 (Invisible Activity). *An activity $t \in T$ is called invisible activity iff $t \notin dom(l)$.*

Figure 6 shows a fragment of a process model that contains a decision point at the place *p1* where each of the alternative paths starts with an invisible activity (denoted as small activities filled with black color). Since these activities cannot be observed in the log it means that considering the occurrence of the first activity in each alternative branch is not always sufficient in order to classify the possible choices relating to a decision point.

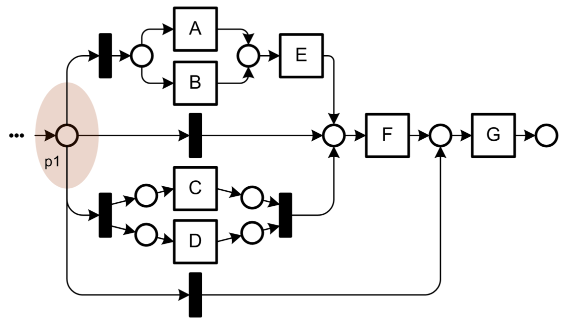


Fig. 6. A decision point involving invisible activities

Instead, the occurrence of each of the activities A or B indicates that the first (i.e., upmost) alternative branch has been chosen during the process execution. Similarly, the occurrence of each of the activities C or D indicates the decision for the third branch[5]. So, invisible activities need to be traced until the next visible activities have been found, which may lead to a set of activities whose occurrences each indicate the decision for the respective alternative branch.

However, this tracking can also reach too far. Looking for visible successors of the invisible activity which starts the second branch (see Figure 6) results in finding activity F, whose occurrence, however, does not necessarily indicate that the second branch had been chosen at *p1*. Since F is preceded by a join construct the first or third path might have been followed as well. Similarly, the occurrence of G is not sufficient to conclude that the fourth branch has been followed. Therefore, we stop tracking invisible activities as soon as a join construct is encountered, and those alternative paths that cannot be specified in the described way are discarded from the analysis.

An additional challenge in classifying the possible choices at a decision point with respect to the log is posed by *duplicate activities*. They emerge from the fact that real-life process models often contain multiple activities that have the same log event associated (i.e., l is not an injective function), which means that their occurrences cannot be distinguished in the log.

Definition 3 (Duplicate Activity). *An activity $t \in T$ is called duplicate activity iff $\exists_{t' \in T}\ t \neq t' \wedge l(t) = l(t')$.*

Figure 7 shows a fragment of a process model that contains a decision point where each of the alternative paths starts with a duplicate activity A, which in the first branch is followed by another duplicate activity B. Although duplicate activities (highlighted in grey color) have an associated log event, its occurrence cannot be used to classify the possible choices related to a decision point as it could also stem from another activity.

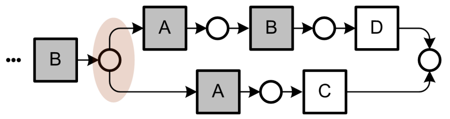


Fig. 7. A decision point involving duplicate activities

A possible solution to deal with duplicate activities is to treat them in the same way as invisible activities, that is, to trace their succeeding activities until either an unambiguous activity (i.e., a non-duplicate visible activity) or a join construct has been encountered. With respect to Figure 7, therefore, only C and D can be used to determine which path has been taken by a specific process instance.

Algorithm 1 summarizes how a decision point found in a Petri net process model can be expressed as a set of possible decisions with respect to the log. The starting point is a place with more than one outgoing arc, i.e., a decision point. Then, each of the outgoing arcs is considered as an alternative branch, and thus as a potential decision class. If the first transition found in such a branch is neither invisible nor duplicate,

[5] Note that, although the activities C and D are in parallel, and therefore will both be executed, observing the occurrence of one of them is already sufficient.

Algorithm 1 Recursive method for specifying the possible decisions at a decision point in terms of sets of log events

determineDecisionClasses :

```
 1: $decisionClasses \leftarrow$ new empty set
 2: while outgoing edges left do
 3:    $currentClass \leftarrow$ new empty set
 4:    $t \leftarrow$ target transition of current outgoing edge
 5:    if $(t \neq$ invisible activity$) \wedge (t \neq$ duplicate activity$)$ then
 6:       add $l(t)$ to $currentClass$
 7:    else
 8:       $currentClass \leftarrow traceDecisionClass(t)$
 9:    end if
10:    if $currentClass \neq \emptyset$ then
11:       add $currentClass$ to $decisionClasses$
12:    end if
13: end while
14: return $decisionClasses$
```

traceDecisionClass :

```
 1: $decisionClass \leftarrow$ new empty set
 2: while successor places of passed transition left do
 3:    $p \leftarrow$ current successor place
 4:    if $p =$ join construct then
 5:       return $\emptyset$                                          // (a)
 6:    else
 7:       while successor transitions of $p$ left do
 8:          $t \leftarrow$ current successor transition
 9:          if $(t \neq$ invisible activity$) \wedge (t \neq$ duplicate activity$)$ then
10:             add $l(t)$ to $decisionClass$
11:          else
12:             $result \leftarrow traceDecisionClass(t)$
13:             if $result = \emptyset$ then
14:                return $\emptyset$                                 // (a)
15:             else
16:                $result \cup decisionClass$
17:             end if
18:          end if
19:       end while
20:    end if
21: end while
22: return $decisionClass$                                        // (b)
```

the associated log event can be directly used to characterize the corresponding decision class. With respect to the example model in Figure 2(b) this is the case for all three decision points. Following the described procedure decision point *p0* yields $\{\{B\},\{C\}\}$, *p2* yields $\{\{E\},\{F\}\}$, and *p3* yields $\{\{F\},\{G\}\}$.

However, if the first transition found in such a branch is an invisible or duplicate activity, it is necessary to trace the succeeding transitions until either a join construct has been encountered and the whole decision class is discarded (a) or all the succeeding transitions (or recursively all their succeeding transitions) could be used for the specification of that class (b). With respect to the decision point *p1* in Figure 6 the described procedure yields $\{\{A,B\},\{C,D\}\}$. So the second and the fourth branch are not represented as a decision class since a join construct was encountered before a visible activity had been reached, and the first and the third branch are described as a set of log events whose occurrence each indicates the respective decision class. Finally, the decision point in Figure 7 results in $\{\{D\},\{C\}\}$, i.e., the duplicate activities were traced as they could not be used for an unambiguous decision class specification.

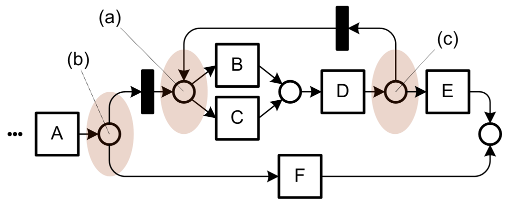


Fig. 8. Loop semantics affect the interpretation of decision occurrences

Another obstacle to be overcome can be seen in the correct interpretation of the loop semantics of a process model. Figure 8 shows a fragment of a process model containing three decision points that can all be related to the occurrence of activity B and C. However, as discussed in the remainder, the corresponding interpretations differ from each other.

Decision points contained in a loop (a) Multiple occurrences of a decision related to this decision point may occur per process instance, and every occurrence of B and C is relevant for an analysis of this particular choice. This means that, opposed to the procedure described in Section 3.2, one process instance can result in more than one training example for the decision tree algorithm.

Decision points containing a loop (b) Although a process instance may contain multiple occurrences of activity B and C, only the first occurrence of either of them indicates a choice related to this decision point.

Decision points that *are* loops (c) This choice construct represents a post-test loop (as opposed to a pre-test loop), and therefore each occurrence of either B or C except the first occurrence must be related to this decision point.

This example demonstrates that in the presence of loops it is not sufficient to consider the mere occurrence of activity executions in order to correctly classify the training examples (i.e., the past process executions that are used to derive knowledge about data dependencies). Instead, it may be important that a log event X is observed *after* log event Y but *before* log event Z. Similarly, the non-occurrence of a log event can be as important as its occurrence. Therefore, a more powerful specification language (e.g., some temporal logic) must be developed in order to express such constraints. Finally, the possibility to express non-occurrence also enables the treatment of alternative paths that are discarded by the current approach. For example, the second branch in Figure 6 can be specified if we are able to say that F happened but E, C, and D did not.

5 Decision Mining with the ProM Framework

The approach presented in this paper has been implemented as a plug-in for the ProM Framework. The *Decision Miner* plug-in[6] determines the decision points contained in a Petri net model[7], and specifies the possible decisions with respect to the log while being able to deal with invisible and duplicate activities in the way described in Section 4. Figure 9(a) shows the model view of the Decision Miner, which provides a visualization of each decision point with respect to the given process model.

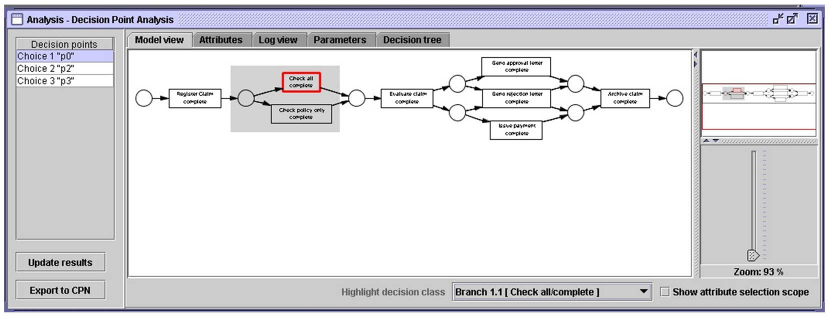
The attribute view shown in Figure 9(b) allows for the selection of those attributes to be included in the analysis of each decision point. Here the advantage of a tool suite like ProM becomes visible. The tight integration of further analysis components available in the framework can be used to add meta data to the event log before starting the actual decision point analysis. For example, a previous performance analysis evaluating the timestamps of each log event (see Figure 1) can provide additional attributes, such as the flow time and waiting time, to specific activities or the whole process instance. These attributes then become available for analysis in the same way as the initial attributes.

While the Decision Miner formulates the learning problem, the actual analysis is carried out with the help of the decision tree algorithm J48 provided by the Weka software library [13], which is their implementation of an algorithm known as C4.5 [10]. The Parameters view offers the modification of the full range of parameters that are available for the used decision tree algorithm from the Weka library.

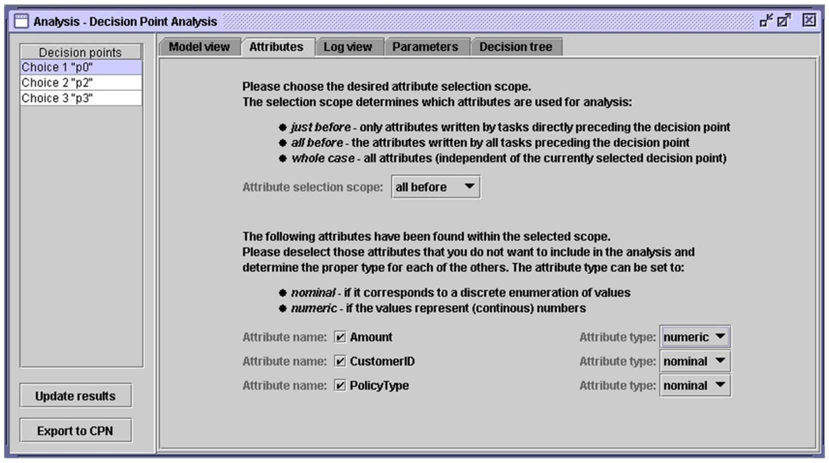
In addition, the log view provides a means to manually inspect the process instances categorized with respect to the decisions made at each decision point in the model. Finally, there is the possibility to export the enhanced process model as a Colored Petri net (CPN) to a tool called CPN Tools [7, 12], which, e.g., enables the subsequent use of the simulation facilities that are available in CPN Tools. However, a detailed description of the CPN representation is beyond the scope of this paper.

[6] Both the Decision Miner, which is embedded in the ProM framework, and the log file belonging to the example process used in this paper can be downloaded from *www.processmining.org*.

[7] Note that although only Petri net process models are directly supported by the Decision Miner, various other process model types (EPC, YAWL, etc.) are indirectly supported via conversion tools available in ProM.



(a) Model view visualizes decision points in the process model



Please choose the desired attribute selection scope.
The selection scope determines which attributes are used for analysis:

- *just before* - only attributes written by tasks directly preceding the decision point
- *all before* - the attributes written by all tasks preceding the decision point
- *whole case* - all attributes (independent of the currently selected decision point)

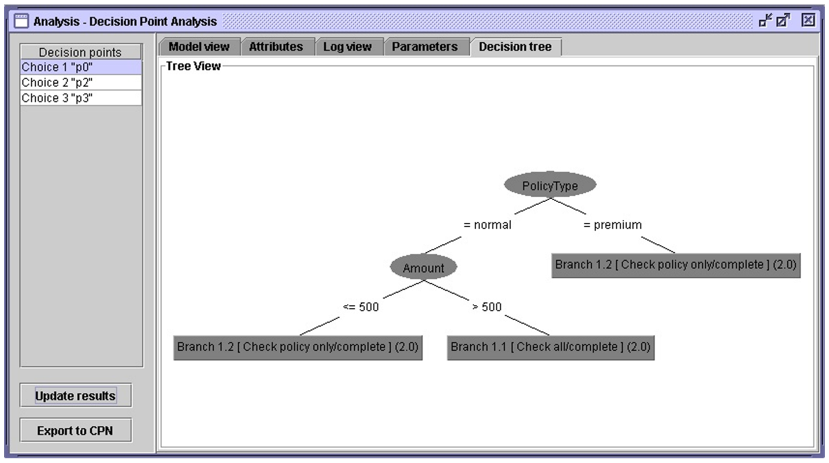
Attribute selection scope: all before

The following attributes have been found within the selected scope.
Please deselect those attributes that you do not want to include in the analysis and determine the proper type for each of the others. The attribute type can be set to:

- *nominal* - if it corresponds to a discrete enumeration of values
- *numeric* - if the values represent (continous) numbers

Attribute name: ☑ Amount Attribute type: numeric
Attribute name: ☑ CustomerID Attribute type: nominal
Attribute name: ☑ PolicyType Attribute type: nominal

(b) Attribute view allows for selection of those attributes to be used for analysis



(c) Decision tree result for analysis of decision point “p0”

Fig. 9. Screenshots of the the Decision Miner in ProM

6 Related Work

The work reported in this paper is closely related to [6], where the authors describe the architecture of the *Business Process Intelligence* (BPI) tool suite on top of the *HP Process Manager* (HPPM). Whereas they outline the use of data mining techniques for process behavior analysis in a broader scope, we show in detail how a decision point analysis can be carried out also in the presence of duplicate and invisible activities. In [8] decision trees are used to analyse staff assignment rules. Additional information about the organizational structure is incorporated in order to derive higher-level attributes (i.e., roles) from the actual execution data (i.e., performers). In [4] the authors aim at the integration of machine learning algorithms (neural networks) into EPC process models via fuzzy events and fuzzy functions. While this approach may support, e.g., a concrete mortgage grant decision process, we focus on the use of machine learning techniques as a general tool to analyze business process executions.

7 Conclusion

In this paper we have highlighted the challenges that underlie the application of machine learning techniques in order to support the analysis of choices in the context of business processes. For such an analysis tool it is crucial to provide the greatest possible flexibility to the business analyst (e.g., with respect to the modification of algorithm parameters, and to the selection and interpretation of data attributes) when applying these techniques. Furthermore, the control flow semantics of the given process model need to be respected in order to provide meaningful results. Finally, a close integration of the results provided by other analysis techniques (such as performance analysis) is expected to increase the potential of decision mining in real-life business processes. A *Decision Miner* that analyzes the choice constructs of Petri net process models using decision trees has been developed within the ProM Framework.

Future research plans include the support of further types of process models (such as EPCs), and the provision of alternative algorithms already available in the data mining field (and related software libraries). For example, sometimes a concept description can be better directly captured in rules than in a decision tree. The reason for this is a problem known as the replicated subtree problem, which may lead to overly large decision trees.

Finally, the application of data mining techniques in the context of business processes can be beneficial beyond the analysis of decisions that have been made. Instead, a free specification of the learning problem on the available data can be used to, e.g., mine association rules, or to assess potential correlations to the fact that a case does or does not comply with a given process model (whereas process compliance has been previously examined by a technique called conformance checking [11]).

Acknowledgements

This research is supported by EIT and the IOP program of the Dutch Ministry of Economic Affairs. The authors would also like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song, Laura Maruster, Christian Günther, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, Huub de Beer, Peter van den Brand, et al. for their on-going work on process mining techniques.

References

1. W.M.P. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*, pages 138–145. Riga Technical University, Latvia, 2004.
2. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
3. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
4. O. Adam, O. Thomas, and P. Loos. Soft Business Process Intelligence — Verbesserung von Geschäftsprozessen mit Neuro-Fuzzy-Methoden. In F. Lehner et al., editor, *Multikonferenz Wirtschaftsinformatik 2006*, pages 57–69. GITO-Verlag, Berlin, 2006.
5. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology.* Wiley & Sons, 2005.
6. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan. Business Process Intelligence. *Computers in Industry*, 53(3):321–343, 2004.
7. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Springer-Verlag, 1997.
8. L. T. Ly, S. Rinderle, P. Dadam, and M. Reichert. Mining Staff Assignment Rules from Event-Based Data. In C. Bussler et al., editor, *Business Process Management 2005 Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 177–190. Springer-Verlag, Berlin, 2006.
9. T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
10. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
11. A. Rozinat and W.M.P. van der Aalst. Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In C. Bussler et al., editor, *Business Process Management 2005 Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2006.
12. A. Vinter Ratzer, L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen, and K. Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In W.M.P. van der Aalst and E. Best, editors, *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 450–462. Springer Verlag, 2003.
13. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, 2005.