Discovering Process Model from Event Logs by Considering Overlapping Rules

Yutika Amelia Effendi, Riyanarto Sarno
Department of Informatics
Faculty of Information Technology, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
e-mail: yutika.effendi@gmail.com, riyanarto@if.its.ac.id

***Abstract*—Process Mining is a technique to automatically discover and analyze business processes from event logs. Discovering concurrent activities often uses process mining since there are many of them contained in business processes. Since researchers and practitioners are giving attention to the process discovery (one of process mining techniques), then the best result of the discovered process models is a must. Nowadays, using process execution data in the past, process models with rules underlying decisions in processes can be enriched, called decision mining. Rules defined over process data specify choices between multiple activities. One out of multiple activities is allowed to be executed in existing decision mining methods or it is known as mutually-exclusive rules. Not only mutually-exclusive rules, but also fully deterministic because all factors which influence decisions are recorded. However, because of non-determinism or incomplete information, there are some cases that are overlapping in process model. Moreover, the rules which are generated from existing method are not suitable with the recorded data. In this paper, a discovery technique for process model with data by considering the overlapping rules from event logs is presented. Discovering overlapping rules uses decision tree learning techniques, which fit the recorded data better than the existing method. Process model discovery from event logs is generated using Modified Time-Based Heuristics Miner Algorithm. Last, online book store management process model is presented in High-level BPMN Process Model.**

Keywords—Decision Mining; Process Mining; Overlapping Rules; Process Discovery; BPMN; Petri Net; Modified Time-based Heuristics Miner

I. INTRODUCTION

Nowadays, organizations represent their business processes through process models. These process models are used for multiple reasons, such as, to specify, document, analyze processes, and extract information [1]. Generally, process models show activities, their dependencies, and their relation in a graph representation. There are a lot of ways to represent process model, such as UML, Causal Net, BPEL, BPMN, EPC, PNML, etc [2]. Each type of model has different characteristic, such as Petri Net uses token to connect the activity in the business process model, while in Causal Net, the activity can be connected directly [3].

Apart from the sequence of activities, during the execution of non-common processes, there are some decisions which need to be made, such as decisions between multiple activities. Decision points mean the choices explicitly shown in process model [4]. The alternatives available are specified by a decision point. These days, when using process models, there are some important challenges, namely understanding the conditions of particular alternative activities are executed and the decision which need to be made in a process.

Process Mining is a technique to automatically discover and analyze business processes with decision points from event logs [5]. Nowadays, all of information systems have event logs which contain information, such as Case ID, activity, timestamp, resources [6]. Therefore, the main purpose of decision mining is to discover the rules which use data recorded in information systems of organization as supporting process [7]. Different from original event logs, decision mining uses event logs which contain process data, such as amount and attributes. The data was available when the activity was performed. Exclusive choices in existing decision mining techniques depend on the rules attached to the alternative activities of a exclusive choice which must be mutually exclusive. Non-deterministic becomes the characteristic of business rules which are often happened in fact, so solving this condition is a must in a particular situation. Missing contextual information or conflicting rules and incomplete event logs can cause ambiguity [8]. The mutually exclusive rules underlying decision making cannot be discovered with having incomplete information in the event log. Hence, this assumption is rare to happen in reality. This paper presents a discovery technique for process model with data by considering the overlapping rules from event logs. Before we discover the overlapping rules, we need to mine the process model in Petri Net using modified time-based heuristics miner algorithm [9]. After process model is discovered, we replay the event logs with data attributes amount and status. Because of replaying event logs, Petri Net with Data or well known as Data Petri Net (DPN) is able to be discovered. Using DPN, overlapping rules which are contained in process model can be mined well. Last, we convert the Petri Net with Data into BPMN process model as higher-level process modeling notations [10].

There are three components to build an initial decision tree, namely using decision mining technique, event logs, and

978-1-5386-0549-3/17/$31.00 ©2017 IEEE

based on observations. After that, new rules for each leaf of decision tree which are led from a new decision tree is used to learn the wrongly classified instances [4]. In this experiment, we use a real-life event log data which are taken from online book store management process to evaluate our technique. We organize this paper into several sections. In Section II, we review Data Petri Net (DPN) and Modified Time-based Heuristics Miner algorithm. We present our proposed method in Section III. The evaluation setup, results, and discussion are explained in Section IV. Last, this paper is concluded with conclusions and sketched future works in Section V.

II. RELATED WORK

Section II gives a short summary of Data Petri Net (DPN) and Modified Heuristics Miner algorithm.

A. Data Petri Net (DPN)

DPN *(P; T; F; V; U; W; G)* is an extended Petri Net. The additional components describe the data perspective of the process model*:*

– (*P; T; F*) *is a Petri Net;*
– *U the universe of possible variable values;*
– *W* : *T* ! 2^{V} *for each transition, there is a set of write operations;*
– *V a set of variables;*
– *G* : *T* ! *Formulas*(*V*) *a set of guard expressions (guards).*

Through write operations (*W*), the values of variable are updated by transitions. Furthermore, when transitions may be executed, the variables of the DPN further constrain are defined by guards. There are two conditions that must be met in order to execute a transition in a DPN. Firstly, at least one token should be contained in input places. Secondly, the current variable assignment should satisfy the guard. To define the state of a DPN, there should be at least two components, namely the current values of all variables and the marking of all variables. From initial to final state, a DPN has the behavior which fits to all sequences of transition [11].

B. Modified Time-based Heuristics Miner Algorithm

A modification of process mining algorithm, Heuristics Miner based on time interval is known as Modified Time-Based Heuristics Miner algorithm. This algorithm relies on the information of time interval in the event logs for all activities, so that the algorithm enable to produce better process models than that of original algorithm. Based on their evaluation using yarn manufacturing process [9], the algorithm can effectively distinguish short loops, sequence and parallel relation (XOR, OR, and AND). Meanwhile, the original Heuristics Miner algorithm is not able to discover parallel relation OR in the process model. In dependency graph, the average value of dependency measure become the main calculation of the threshold intervals. The main steps of this modified algorithm are mining dependency graph, checking short loops, and mining parallel activities [9].

a. Mining dependency graph

Mining dependency graph is a first step which will be done in Heuristics Miner algorithm, because this step focus on construction and calculation of the dependency graph. In Heuristics Miner, a frequency-based metric can be generated from the dependency between two activities. The direct successor frequency matrix will be generated from the frequency of activities. In this step, the dependency measure will be determined using RBT, POT, and DT as thresholds.

- Relative-to-best threshold (RBT)
  Selecting the edges which are going to be accepted into the process model are the purpose of RBT.
- Positive observations threshold (POT)
  POT is used to control the minimum frequencies of dependency between activities.
- Dependency threshold (DT)
  Choosing the dependency measure which has higher value than the value of the threshold in matrix dependency is the main purpose of DT.

b. Checking short loop

A loop means there is possibility that one activity may be executed more than once in process model. Using the dependency measure, the original heuristics miner algorithm can detect long distance loops. But, the dependency measure is not able to detect short loops. After the algorithm is modified, short loops in process model can be detected well. Length-one-loop and length-two-loop are two types of short loops.

c. Mining parallel activities

Mining parallel activities focus on obtaining the frequency of parallel activities. One of the many ways which are generally used is the number of concurrent relation of all parallel activities. The frequency of parallel activities is needed in parallel measure calculation. A common event log uses single timestamp [6]. However, a double timestamp event log is extended form of the single timestamp event log which is used in concurrent relation discovery. Using the double timestamp, the parallel relation (AND, OR, and XOR) in process model is discovered.

III. THE PROPOSED METHOD

This subsection presents an integrated discovery approach for discovering overlapping rules as well as the algorithms which are used in this research.

A. Integrated Discovery Approach

- A localized event log *L* is obtained and log *L1* corresponding to regions are extracted.
- Process model discovery from event log *L1* is generated using Modified Time-Based Heuristics Miner Algorithm. The result of this step is Labeled Petri Net *PN1*.

- After process model is discovered, we replay the event logs with data attributes amount and status. The results of this step are labeled Petri Net *PN1* and replay result *R1*.
- We use the event log *L1* with data attributes amount and status to discover Petri Net with Data. So, from this step we produce Petri Net with Data DPN.
- Next, overlapping rules in DPN are needed to discover. Input of this step is Petri Net with Data DPN, so is the output.
- At last, converting Petri Net with Data DPN into High-level BPMN Process Model will be presented.

B. *Overall Discovery Procedure for Overlapping Rules*

The overall discovery procedure for the entire process model is presented in Algorithm 1. The algorithm uses input, namely an event log *L1* and a Petri Net without Data (*P; T; F*). The algorithm returns the guard function *G* of the DPN. As described in Algorithm 1, the guard function ψp is constructed for each decision point $p \in P$. To construct them, we use the observation instances I_p. After we acquire the guard function, each transition needs to be assigned the conjunction of all rules which are acquired from the input places. This technique is different in terms of how the actual rules are obtained and it can discover guards which may be partially overlapping.

C. *Discovering Overlapping Rules*

This step introduces a decision tree builder which will be presented in Algorithm 2. By using given the observation instances *I*(*p*), we discover overlapping guards for place *p*. Not only discovering overlapping guards, but also defining two parameters, namely the merge ratio ϵ and the minimum number of instances *n*.

Algorithm 1: discoverGuardFunction

Input: Petri Net (*P; T; F*), Event Log (*L1*), Merge Ratio (ϵ), Minimum Instances (*v*)
Result: Guard function of the DPN *G*

```
1   foreach p ∈ P s.t. |p•|> 1 do
2       ψp ← constructEvaluator(p, I, v .|I(p)|, ϵ )
3   end
4   foreach t ∈ T do
5       G(t) ← true
6       foreach p ∈•t do
7           G(t) ← G(t) ∧ ψp(t)
8       end
9   end
10  return G
```

- C4.5 Decision Tree Builder

In the decision tree induction, for the splitting criterion, the minimum number of instances on a leaf is defined as $n \in N$. Meanwhile, a multi-set of observation instances over a set *V* of variables is declared as *O*. By using set of instances which -

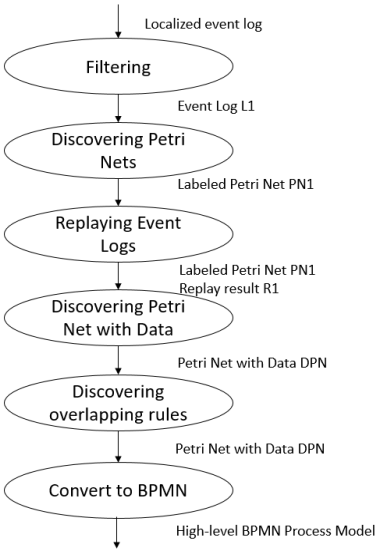

Fig. 1. The integrated discovery approach presented in this paper

have obtained before, function $buildTree_n(O) \in 2^{Formulas(V)} \times T$ is built to return the leaves of a C4.5 decision tree builder. Transition $t \in T$ under condition $expr \in Formulas(V)$ is going to be predicted by a leaf of a C4.5 decision tree builder. Initially all transitions are assigned placeholder guards (t) = *void* [6].

Algorithm 2: constructEvaluator

Input: Place (*p*), Observation Instances (*I*), Merge Ratio (ϵ), Minimum Number of Instances (*n*)
Result: Guard Function (ψ)

```
1   ψ : T → Formula ∪ {void} s.t. ∀t ∈ p• : ψ(t) = void
2   foreach leaf: (expr, t) ∈ buildTree_n (I(p)) do
3       ψ(t) ← ψ(t) ∨ expr
4       I ← [(x,t') ∈ I(p) | t≠t' ∧ expr evaluates to true for x]
5       subTree ← buildTree_{n |I|/|I|} (I)
6       if |subTree| > 1 then
7           foreach subLeaf: (subExpr, t') ∈ subTree do
8               ψ(t') ← ψ(t') ∨ (expr ∧ subExpr)
9           end
10      else
11          {(true, t')} ← subTree
12          if |I| > n and |{(x, t) ∈ I| t≠t'}| / |I| < ϵ then
13              ψ(t') ← ψ(t') ∨ (expr)
14          end
15      end
16  end
17  foreach {t∈ p• | ψ(t) = void} do ψ(t) = true
18  return ψ
```

D. *Converting Petri Nets with Data to BPMN Models with Data*

In this subsection we will introduce an approach for transforming Petri Nets with data to High-level BPMN process model with data [10].

Suppose that DPN = *(PN; V; U; R; W; G)* is an initial Petri Net with data. In order to transform this model to a target

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors*).

BPMN model with data $BPMN_{data}$ = $(BPMN_{core}; DO; DA; Expr; SF_{default})$, where $BPMN_{core}$ = $(FN; A; G_{XOR}; G_{OR}; G_{AND}; e_{start}; E_{end}; E_{cancel}; SF; \lambda A)$, the following steps are to be performed:

1. labeled Petri Net PN = $(P; T; F; M_{init}; M_{final}; l)$ is converted to a core BPMN model $BPMN_{core}$, using the algorithm described in [16] (without removing activities, which correspond to invisible transitions); by $M_A : T \rightarrow A$, we denote a function, which maps transitions to activities;
2. for each variable from V a data object do is created and added to the target BPMN model, the mapping of variables to data objects is defined as a function $M_D : V \rightarrow DO$;
3. for each non-invisible $t \in T$, for each $v \in R(t)$, a data input association $(M_D(v), M_A(t))$ is added to DA; similarly, for each $t \in T$, for each variable v from W (t), a data output association $(M_A(t), M_D(v))$ is created and added to DA;
4. for each $t \in T$ with a guard $G(t)$ an expression for each incoming sequence flow of activity $M_A(t)$ is set to $G(t)$, if the source node of this sequence flow is an exclusive or inclusive gateway, default condition expressions are chosen in an arbitrary way;
5. each activity corresponding to an invisible transition is removed from the target BPMN diagram along with incoming and outgoing sequence flows in accordance with the simplification technique presented in [16]; all outgoing sequence flows of exclusive gate-ways added instead of removed sequence flows inherit condition expressions.

Variables, read and write functions of a data Petri Net are trivially transformed to data objects, input and output data associations respectively [10].

IV. EVALUATION

In this subsection, a real-life data set is used to evaluate our technique and we present the final process model.

A. Evaluation Setup

Event Log and Process Model

In this research, a real-life event log data which are taken from online book store management process is used to evaluate our technique. The event logs contain the information, such as the ID, the case ID, the activities, and the start time, and the complete time (double timestamp).

Next, process model discovery from event log is generated using modified time-based heuristics miner algorithm. The process model shows the sequence and parallel relation. We present the discovered process model in Petri Net.

Experimental Design

Firstly, event logs of oline book store management process (*L1*) are obtained. Then, process model discovery from event – log *L1* is generated using Modified Time-Based Heuristics Miner Algorithm. The result of this step is Labeled Petri Net *PN1*. After process model is discovered, we replay the event logs with data attributes amount and status. The results of this step are labeled Petri Net *PN1* and replay result *R1*. After that, we use the event log *L1* with data attributes amount and status to discover Petri Net with Data. So, from this step we produce Petri Net with Data DPN. Next, overlapping rules in DPN are needed to discover. Input of this step is Petri Net with Data DPN, so is the output. At last, converting Petri Net with Data DPN into High-level BPMN Process Model will be presented.

TABLE I FRAGMENT OF EVENT LOGS FROM ONLINE BOOK STORE MANAGEMENT PROCESS

ID	Case ID	Activity	Start Time	Complete Time
e1	PP1	Choose Books	6/20/2014 8:32	6/20/2014 13:42
e2	PP1	Check price	6/20/2014 13:42	6/20/2014 23:41
e3	PP1	Order to seller	6/20/2014 23:41	6/21/2014 9:30
e4	PP1	Create Bill	6/21/2014 9:30	6/21/2014 10:46
e6	PP1	Send the notification	6/21/2014 10:46	6/21/2014 16:57
e7	PP2	Choose Books	6/21/2014 16:57	6/21/2014 18:09
e8	PP2	Check price	6/21/2014 18:09	6/22/2014 4:14
e9	PP2	Order to seller	6/22/2014 4:14	6/22/2014 10:51
e10	PP2	Create Bill	6/22/2014 10:51	6/22/2014 16:28
e11	PP2	Send the ordered books	6/22/2014 16:28	6/22/2014 23:42
e12	PP3	Choose Books	6/22/2014 23:42	6/23/2014 4:48
e13	PP3	Check price	6/23/2014 4:48	6/23/2014 16:44
e14	PP3	Order to seller	6/23/2014 16:44	6/23/2014 23:26
e15	PP3	Create Bill	6/23/2014 23:26	6/24/2014 5:40
e17	PP3	Send the Ordered Books	6/24/2014 5:40	6/24/2014 7:48
e18	PP4	Choose Books	6/24/2014 7:48	6/25/2014 1:08
e19	PP4	Check price	6/25/2014 1:08	6/25/2014 3:02
e20	PP4	Order to seller	6/25/2014 3:02	6/25/2014 5:11
e21	PP4	Create Bill	6/25/2014 5:11	6/25/2014 8:25
e22	PP4	Send the Notification	6/25/2014 8:25	6/25/2014 12:45
	…	...	….	…
	PP18	….	….	….

B. Results and Discussion

Table I presents the fragment of event logs which is used for the experiment. The event logs contain the information, such as the ID, the case ID, the activities, the start time, and the complete time (double timestamp).

Next step, the event log *L1* is mined using Modified Time-Based Heuristics Miner algorithm. Table II presents causal matrix of the process model. Causal matrix contains the dependency relation in activities in the form of input and output.

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors).*

TABLE II CAUSAL MATRIX

INPUT	ACTIVITY	OUTPUT
{}	Choose books	Check price
Choose books	Check price	Order to seller
Check price	Order to seller	Create bill
Order to seller	Create bill	Send the ordered books
Order to seller	Create bill	Send the notification
Create bill	Send the ordered books	{}
Create bill	Send the notification	{}

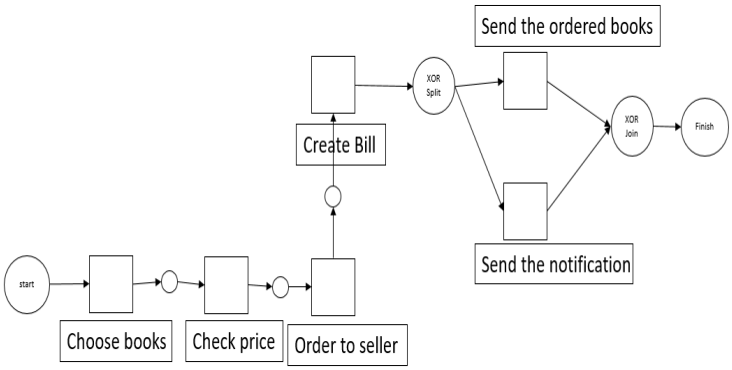

Fig. 2. The discovered process model

Since there are input which is same activity but outputs are different and more than one activity, we can conclude that there is a parallel activity, namely *Create Bill* → *w Send the Ordered Books* ∧ *Send the Notification*. Using equations in [9], the split and join pattern which are used in this process model are XOR. Hence, Fig.2 shows the discovered process model.

We replay the event logs with data attributes amount and status after process model is discovered. The results of this step are labeled Petri Net *PN1* and replay result *R1*. Table III presents the fragment of event logs with data attributes amount and status. To discover Petri Net with Data, we use the event log *L1* with data attributes amount and status. In the discovered process model, the attributes amount and status are shown. The discovered process model (Petri Net with Data) is shown in Fig.3.

The guards are placed on all three transitions in Data Petri Net (DPN) which are shown in Fig.3. The enablement of guards rely on the current task of attribute status. According to Fig.3, when status is *unpaid,* then transition Send the Notification can be performed. The same condition also applies to transition Payment which can be performed when status is *unpaid.* The choice between both transitions Send the Notification and Payment is categorized as non-deterministic if the guards are overlapping. We may assume the transition Create Bill assigned the value *paid* to status. For example in Fig.3, if the bill is paid directly, then we can only execute transition Send the Ordered Books. The process stops when the final marking of the DPN has been reached.

TABLE III FRAGMENT OF EVENT LOGS WITH DATA ATTRIBUTES AMOUNT AND STATUS

ID	Case ID	Activity	Status	Amount
e1	PP1	Choose Books		
e2	PP1	Check price		
e3	PP1	Order to seller		
e4	PP1	Create Bill	Unpaid	239000
e6	PP1	Send the notification	Unpaid	
e7	PP2	Choose Books	Unpaid	
e8	PP2	Check price		
e9	PP2	Order to seller		
e10	PP2	Create Bill		
e11	PP2	Send the ordered books	Unpaid	144000
e12	PP3	Choose Books	Paid	
e13	PP3	Check price		
e14	PP3	Order to seller		
e15	PP3	Create Bill		
e17	PP3	Send the Ordered Books	Unpaid	353200
e18	PP4	Choose Books	Paid	
e19	PP4	Check price	Paid	
e20	PP4	Order to seller		
e21	PP4	Create Bill		
e22	PP4	Send the Notification		
	…	…	….	…
	PP18	….	….	….

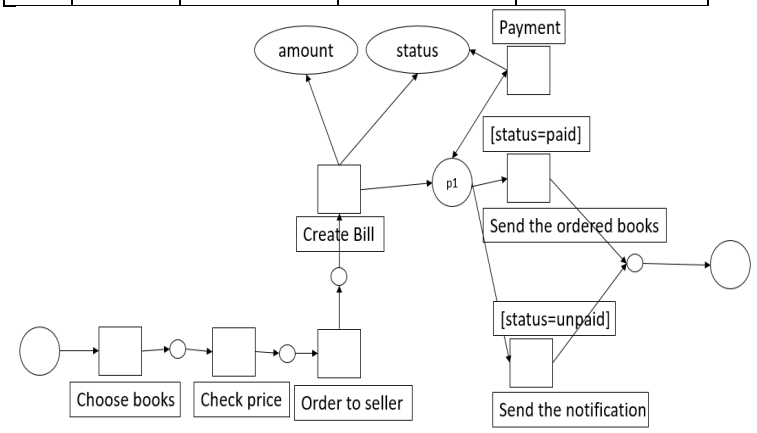

Fig. 3. The discovered process model with data

From the steps which are performed before, Petri Net with Data DPN is generated. Next, overlapping rules in DPN are needed to discover. Input of this step is Petri Net with Data DPN, so is the output.

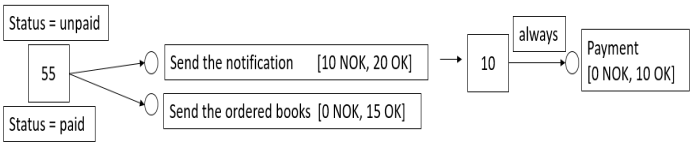

Fig. 4. The discovered decision trees in the event log.

By using event logs given in Table III, we obtain the multi-set of observations for place p_1. Using Algorithm 2 (constructEvaluator), the guard estimation function $_{p1}$ is built. Firstly, we build a decision tree and assume that there are two leafs in this initial decision tree. They are l_1 = (status = *paid ;* Send the Ordered Books) and l_2 = (status = *unpaid ;* Send the Notification). Root node in rectangle form defines the number of instances. Meanwhile, next to leaf nodes in circle form defines the number of correctly (OK) and wrongly (NOK) predicted instances. There are ten of the thirty instances classified as Send the Notification which are wrongly classified. Transition Payment was observed in all those instances. When status is *unpaid,* the transition Payment is performed and those wrongly classified instances give proof for it. We build an additional decision tree for those ten instances which are wrongly classified as shown in Fig.4 (on the right-hand side). Instances for transition Payment are contained a set of wrong classification instances I_{l2}. Moreover, only one leaf is included in the additional decision tree. This leaf assumes transition Payment using majority vote. And the condition status = *unpaid* will be added to the guard function of transition Payment when the number of wrongly classified instances $jI_{l2}\,j$ = 10 are above the threshold n.

At last, converting Petri Net with Data DPN into High-level BPMN Process Model will be presented. The steps explained in Section III are used to convert the Petri Net with Data into High-level BPMN Process Model with Data. Fig. 5 shows High-level BPMN Process Model with Data from online book store management process model.

V. CONCLUSION

This paper presents a discovery technique for process model with data by considering the overlapping rules from event logs. Only rules which assume completely deterministic decisions are returned by existing techniques. This assumption is rare to happen in reality. The proposed technique focuses on creating process model which prioritize the fitness of overlapping rules rather than the precision of mutually-exclusive rules. This is happened when the rules are given proof by the behavior.

The work was tested on a real-life event log. The result shows that our technique is able to discover process model with Data from event logs by considering overlapping rules which fit the behavior better than existing techniques and without loosing too much precision. The process model is also converted from Petri Net with Data into High-level BPMN with Data as higher-level process modeling notations.

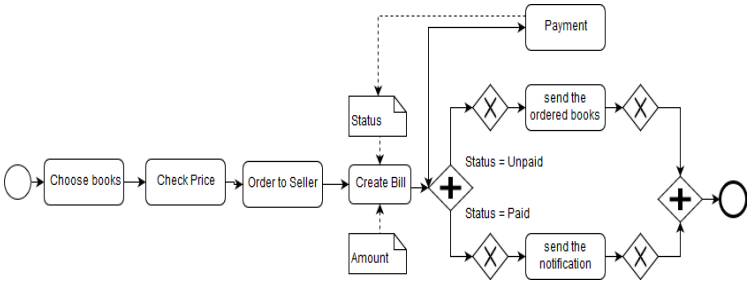

Fig. 5. High-level BPMN Process Model with Data from online book store management process model

Future work aims at finding the various business process model with overlapping rules and moreover, we want to learn and do research about other techniques in decision mining.

ACKNOWLEDGMENT

We would like to thank Department of Informatics, Faculty of Information Technology, Institut Teknologi Sepuluh Nopember for supporting the research.

REFERENCES

[1] W.M.P. van der Aalst. Process Mining, "Discovery, Conformance and Enhancement of Business Processes", Netherlands: Springer, pp. 95-125, 2011.

[2] R. Sarno and K. R. Sungkono, "Hidden Markov Model for Process Mining of Parallel Business Processes," *International Review on Computers and Software (IRECOS)*, vol. 11, no. 4, pp. 290–300, 2016.

[3] R. Sarno, E. W. Pamungkas, D. Sunaryono, and Sarwosri, "Business process composition based on meta models," International Seminar on Intelligent Technology and Its Applications (ISITIA), 2015. http://doi.org/10.1109/isitia.2015.7219998

[4] F. Mannhardt, M. De Leoni, H. A. Reijers, and W.M.P. van der Aalst, "Decision Mining Revisited – Discovering Overlapping Rules". bpmcenter.org.

[5] R. Sarno, W. A. Wibowo, Kartini, F. Haryadita, Y. Effendi, and K. Sungkono, "Determining Model Using Non-Linear Heuristics Miner and Control-Flow Pattern," TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 14 (1), 2016.
http://doi.org/10.12928/telkomnika.v14i1.3257

[6] R. Sarno, F. Haryadita, Kartini, Sarwosri, and A. Solichah, "Business Process Optimization from Single Timestamp Event Log," Computer, Control, Informatics and Its Applications (IC3INA), 2015.
https://doi.org/10.1109/ic3ina.2015.7377745

[7] R. Sarno and K. R. Sungkono, "Coupled Hidden Markov Model for Process Mining of Invisible Prime Tasks," *International Review on Computers and Software (IRECOS)*, vol. 11, no. 6, pp. 539–547, 2016.

[8] A. Rozinat and W.M.P. van der Aalst, "Decision mining in ProM. In: Business Process Management". Volume 4102 of LNCS. Springer Berlin Heidelberg (2006) 420–425

[9] R. Sarno, Y. Effendi, and F. Haryadita, "Modified Time-Based Heuristics Miner for Parallel Business Processes," International Review on Computers and Software (IRECOS), vol. 11 (3), pp. 249-260, March 2016. http://doi.org/10.15866/irecos.v11i3.8717

[10] A. A. Kalenkova, A. Burattin, M. De Leoni, W.M.P. van der Aalst, and A. Sperdutti. "Discovering High-level BPMN Process Models from Event Log Data". bpmcenter.org.

[11] F. Mannhardt, M. De Leoni, H. A Reijers, and W.M.P. van der Aalst, "Balanced multi-perspective checking of process conformance". Computing (2015) doi:10.1007/s00607-015-0441-1.