International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **5277–5293**

PROCESS DISCOVERY FROM THE LOG OF BUSINESS RULE ENGINE

Jianwei Yin, Bin Cao, Shuiguang Deng* and Zhaohui Wu

College of Computer Science
Zhejiang University
No. 38, Zheda Road, Hangzhou 310027, P. R. China
{ zjuyjw; caobin }@zju.edu.cn; wzh@cs.zju.edu.cn; *Corresponding author: dengsg@zju.edu.cn

Received March 2011; revised August 2011

Abstract. *Process/workflow mining aims at discovering the underlying processes to help in improving or rebuilding business processes. Most of the current practices of process mining are based on event logs from Transactional Information Systems (TIS) (such as WFM, ERP, CRM, SCM and B2B systems). However, with the popular deployment and use of business rule engine with TIS, a great number of rule logs are generated, but they are rarely utilized for discovering processes. This paper intends to propose a different perspective for process discovery as compared with the traditional way based on the event logs. Firstly, it illustrates a motivation scenario about process mining from rule logs and then brings forward a framework for process discovery based on rule logs. After that, the mining algorithm called Alpha-r with a case study is introduced to discover a process through mining the relations of traces in rule flow log. Finally, some experiments show the effectiveness and performance of the method.*
Keywords: Process/Workflow mining, Business rule engine, Rule flow, Log

1. Introduction.

In general, enterprise systems usually consist of three layers which are, from bottom to top, persistence layer, business logic layer and presentation layer. The business logic layer represents the core of the application where all of the business processes and business rules take place. Separately, business processes represent what the business does, and business rules represent decisions that the business does.

1.1. Business rule engine.

With the quick development of modern business, the market changes so rapidly that the requirements for the business logic layer are forced to change more frequently than the requirements for the rest of the application. In order to reduce the cost of development and maintenance, the IT companies have to design software systems that can adapt to the quick change very well. So, people tend to use rule engine, derived from the inference engine of Rule-based Expert System [1], to achieve the goal by separating the business logics from the application. A business rules engine is a software system that executes one or more business rules in a runtime production environment [2]. The business rules which are written by a pre-defined language might come from legal regulation, company policy, or other sources. The rule engine accepts the data inputs which is also called facts, builds the business rules and matches them with corresponding facts using algorithms like Rete [3] and LEAPS [4], then fires the matched rules to make decisions. The rules change as soon as the business logics change; however, the other parts of the system including the designs and codes remain intact. As a result, the business rules can be maintained independently according to the business requirements according to the business requirements.

1.2. **Workflow mining.** Today, the workflow management technology has been widely applied in many enterprise information systems. Besides pure WFM (Workflow Management) systems many other software systems have adopted and used workflow technology, such as ERP (Enterprise Resource Planning) systems, CRM (Customer Relationship Management) software, SCM (Supply Chain Management) systems, B2B (Business to Business) applications [5]. The focus of workflow related systems is to design and configure the workflow models which are conformed to the requirements of business. However, constructing the model not only is far from trivial but also requires deep domain knowledge. Naturally, analyzing the dependencies among the business activities and generating the process automatically which can support the workflow design through the events log of workflow related systems become the goal of workflow mining [6]. The simplified process of workflow mining can be seen in Figure 1. Workflow mining assumes the existence of structured logs from information systems to store relevant events for the business. These events are a good source of information and make it possible to discover business processes from log data.

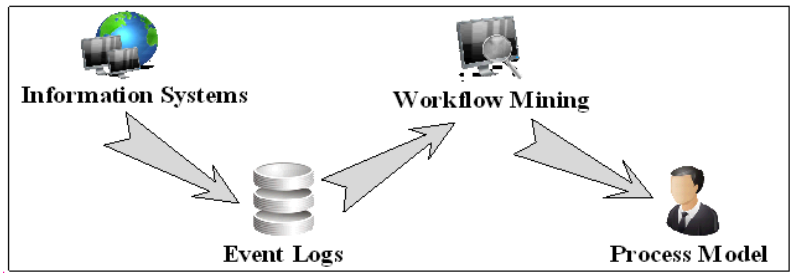

FIGURE 1. The simplified process of workflow mining

Since the workflow model mainly referred to process control flow model, the workflow mining could also be called "*Process Mining*" or "*Process Discovery*". There are at least two significant uses with process mining [7]. First of all, it is often used when no formal description of the process can be obtained by other means, or when the quality of an existing documentation is questionable. For instance, the audit trails of a WFM system, the transaction logs of an ERP system, and the electronic patient records in a hospital can be used to discover workflow models describing actual processes, organizations, and products. Moreover, such event logs can also be utilized by process mining for *Delta analysis* [6], i.e., comparing the actual model with some a priori model to see whether the observed real process conforms to the predefined process.

1.3. **Discovering processes from rules.** Nowadays, more and more people are trying to automate all kinds of business processes and implement complex business decisions since business rules and processes can improve their business by providing a level of agility and flexibility. Integrating business rules into WFM systems can help people to implement the dynamic workflows which are modifiable in run-time. Besides, it is regarded as an important means of increasing the productivity of enterprises since it crosses organizational boundaries and facilitates exchange of information among units and people [8]. Consequently, the integration of rules and workflows has attracted lots of attention from researchers and enterprises, also, many applications are already available such as Drools [9] and ILOG [10].

In this paper, we focus on the situation that workflows use business rules to imperatively make decisions during the flow. These rules are all bound to corresponding activities in the workflow and an embedded rule engine is used (when invoked) to process over the current

state of the business objects to see if the conditions of any rules are met and subsequently execute any resulting action. In this situation, as soon as the integrated systems begin to execute, not only the workflow engine logs the whole instances information which traditional workflow mining depends on, as already noted, but also the rule engine does. The log generated by business rule engine is about the trail of rule engine, i.e., the execution instances of rule engine. Since the business rules are fired following the process which is being executed, it seems that there exist some certain flow relations, which could be discovered, among the business rules. Besides, we can utilize the conditions and actions within rules to remark the discovered flow which means that people can obtain more knowledge to reason the exact execution route of the workflow model. Unfortunately, this kind of log is seldom used for the purpose of mining actual process. For these reasons, we decide to do some work on it and proposed a novel approach for process mining by analyzing the log of rule engine. The approach is based on some theories and foundations which are the same with the classic *Alpha* algorithm (α *algorithm*) [7], so we call it *Alpha-r* algorithm (α-r *algorithm*).

Using α-r algorithm in workflow that integrated with business rules can discover a process model which might be different from the model mined by traditional process mining technique, i.e., the α algorithm. The reason why it happens is that the rules which could be parallel processed in rule firing perspective are serialized within original workflow. In another view, the process discovered from rules can reflect something about real execution of flow which could benefit the redesign of workflow model. Anyway, it can be regarded as another useful choice for supporting workflow design, and combining with traditional workflow mining perspective, it is more overall in improvement of the existed workflow model.

Moreover, in fact, the α-r algorithm can work as long as there exists rules set, that is to say, workflow is not the compulsory. We can use α-r algorithm directly based on rules set to discover a process which never met before.

The remainder of this paper is organized as follows. We show a motivation scenario in Section 2, and introduce our framework for discovering in Section 3. In Section 4, we introduce some preliminaries, give the details of α-r algorithm, and present a case study. Section 5 discusses about the experimental evaluation. Then, we present an overview of related work in Section 6. Section 7 concludes the paper.

2. **A Motivation Scenario.**

In order to make our motivation and proposed approach clear, we illustrate a scenario in bank domain. It is quite necessary and efficient that using rule engine to maintain the business rules which are changed frequently in bank.

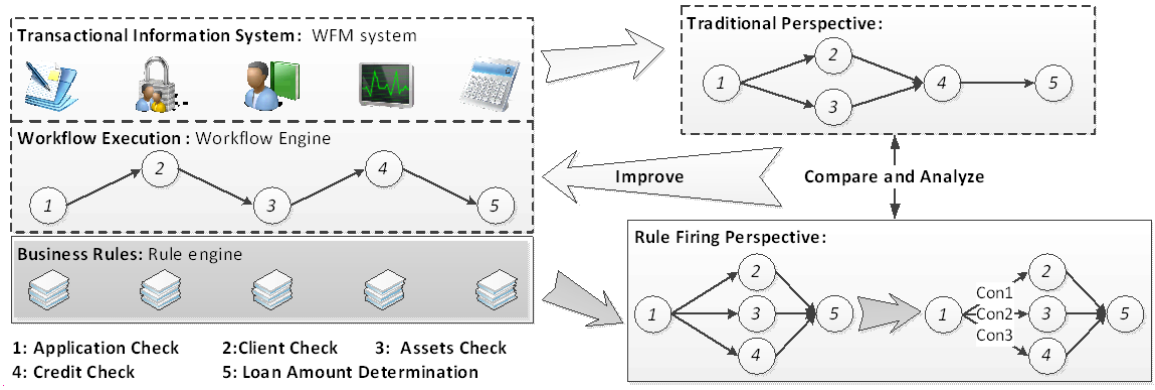

FIGURE 2. A motivation scenario: bank loan

Let us consider the typical scenario of the bank loan business. Assume that someone wants to get a loan from a bank in order to buy something or invest. After submitting the application, the bank will begin to execute the following steps which form a workflow to make correct decision. Typically, the application would firstly be checked since the bank has to find out what the content is, which classification it belongs to (i.e., the large or small amount money) and whether it is illegal. Secondly, in case of any fraud behavior, the identity of applicant would be verified to decide whether he is qualified to get a loan. Thirdly, the bank would have the applicant's assets checked to see if he is potentially capable of repay his loan. Then for the purpose of reducing the risk, the credit of applicant's would be checked by bank. At last, on the basis of previous steps, the bank would determine the loan amount for applicant.

In the above scenario as Figure 2 shows, we notice that there are business rules, in each step, which facilitate the bank to make decision with the help of rule engine. And the whole workflow consist of rules is executed by workflow engine within WFM system. From traditional workflow mining perspective, the actual workflow model might be the one in the upper right corner of Figure 2. However, from rule firing perspective, the workflow model probably different. As shown in the lower right corner of Figure 2, there are three parallel activities (i.e., Client Check, Assets Check and Credit Check), moreover, we can get conditions of each activity. Under the circumstances, we can compare and analyze the two different models, then combine with realities to improve the original workflow. At last, the bank will obtain a more objective workflow model which can be more appropriate for bank loan business.

3. **A Framework for Process Discovery Based on Rule Logs.** Workflow has grown to be a major approach to assist automation of business processes in quite diverse domains. At present, for the purpose of controlling rule execution, some applications use the term "rule flow" for specifying the order in which rule sets should be evaluated [11]. In fact, rule flow can be regarded as a special kind of workflow which includes a set of rules together with the control and data flow among the rules. The current achievements on workflow modeling, execution and cross-enterprise integration provide the means to compose rules in a practicable and convenient way.

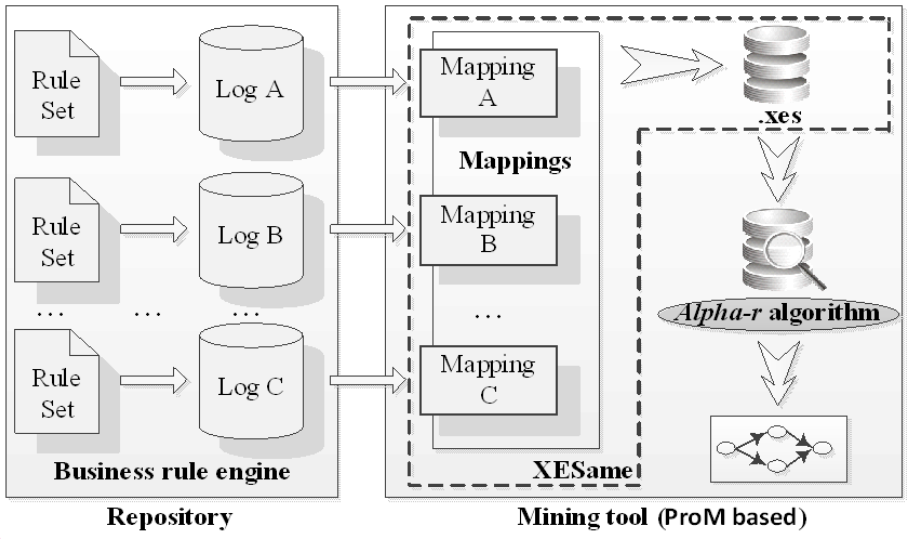

FIGURE 3. A framework for discovering process from rules

As stated before, while there are plenty of systems available for supporting the execution of such rules, the current practices for monitoring and analyzing this execution in the organizational reality still leaves a lot to be desired. The motivation and the α-r algorithm we proposed which concerned with the extraction of knowledge about a (business) process from rules through the execution logs of rule engine are able to fill that gap. In this section, we propose a framework shown in Figure 3 for discovering process from rules based on **ProM** [12] mining tool. **ProM** is an extensible framework, issued under an open source license, which supports a wide variety of process mining techniques in the form of plug-ins. Currently, the **ProM** not only supports several process modeling languages, such as Petri-net [13-15], BPMN [16], etc., but also supports in the extraction of an event log from non-event log data sources with the help of **XESame** tool.

As Figure 3 shows, the framework supports both mapping logs and discovering process. It consists of the following two main parts:

- **Repository:** This part of framework not only provides the input source for mining tool but also is responsible for maintaining the rules and corresponding execution logs which are generated by business rule engine. The format of logs can be various, such as ".csv", ".xls", ".txt" and so on.
- **Mining tool:** This part is based on **ProM**, and there are two main components collaborating with each other. The **XESame** component is used to map and transform the execution logs of rule engine to the standard and general format, i.e., ".xes". This step is meaningful since it loosely coupled with logs, which can reduce the implementation effort and to promote the use of mining techniques in multiple contexts. After finishing the transformation, the α-r algorithm component can execute on the log of ".xes" format for discovery. At last, the discovered process will be presented in the form of diagram which in this paper is a Petri model.

The target of the framework is to discover process from rules through the log of rule engine. The α-r algorithm we used in framework guarantees the whole process. The details about the α-r algorithm for discovering will be presented in the following section.

4. Algorithm for Discovering.

Note that, how to deal with noise is not our concern, we assume that there is no noise and the rule flow log contains sufficient information. Under these ideal circumstances, we could be able to discover potential process based on α-r algorithm.

Within this section, we introduce the details about the α-r algorithm for discovering. First of all, we introduce preliminaries including some basic concepts and definitions which are both related to the approach we proposed. Then, we present the α-r algorithm in form of pseudo code. At the end of this section, we verify the algorithm through a case study.

4.1. Preliminaries.

The α-r algorithm we proposed shares a few same theories and foundations with the classical α algorithm [7]. For example, the definitions of Workflow nets, Implicit place, SWF-net, etc. We won't introduce them all since the author of [7] has described at full length, instead, we just list the different and some same but necessary concepts and definitions for the purpose of accurate description. As previously mentioned, to discover a process from rules through the log of rule engine is our aim. So, choosing an appropriate model method for process (i.e., workflow) is the first step we should do. In our paper, we use the classic Petri-net [13-15] model, namely, Place/Transition nets. Concisely, we introduce the basic concept of the Petri-net as follows.

Definition 4.1. (P/T-nets) *A Place/Transition net, or simply P/T-net, is a tuple (P,T,F), where:*

1. *P is a finite set of places, equals to $\{p_1, p_2, \ldots, p_n\}$.*
2. *T is a finite set of transitions, equals to $\{t_1, t_2, \ldots, t_n\}$, such that $P \cap T = \varnothing$.*
3. *$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the flow relation.*

From above, we can see that the basic elements of a P/T-net are place, transition and arc. Besides, token is another important element which stays in the place to mark the execution of a P/T-net. Normally, we use the following notations to represent these elements.

FIGURE 4. Basic elements of P/T-net

Definition 4.2. (Marked P/T-nets) *A marked P/T-net is a tuple (N, μ), where:*

1. *$N = (P, T, F)$ is a P/T-net.*
2. *μ is a function from P (a finite set of places) to natural numbers N, denoting the marking of a P/T-net, μ: $P \to N$, i.e., $P \to \{0, 1, 2, \ldots\}$.*

Definition 4.3. (Nodes, Inputs, Outputs) *Let $N = (P, T, F)$ is a P/T-net, we have:*

1. *Elements in $P \cup T$ are called nodes.*
2. *If $x, y \in P \cup T$ and $(x, y) \in F$, then x is the input node of y.*
3. *If $x, y \in P \cup T$ and $(y, x) \in F$, then x is the output node of y.*
4. *$\forall x \in P \cup T$, the inputs of x is $\bullet x = \{y | (y, x) \in F\}$.*
5. *$\forall x \in P \cup T$, the outputs of x is $x\bullet = \{y | (x, y) \in F\}$.*

For example, Figure 5 shows a P/T-net consisting of five places and four transitions. Transition C has two input places and one output place. The tokens in first place denote the initial marking, i.e., $\mu = \{2, 0, 0, 0, 0\}$. When P/T-net in Figure 5 is in operation, the number and places of tokens are changing.

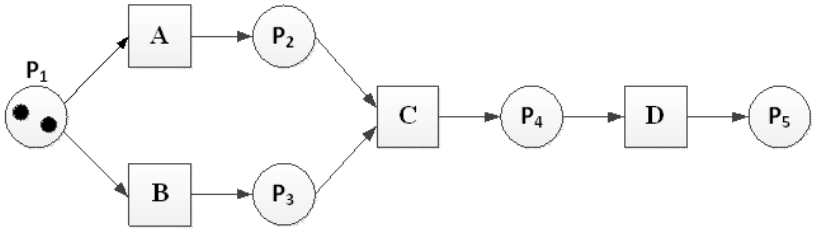

FIGURE 5. A P/T-net example

The dynamic behavior of a P/T-net is controlled by the number and the distribution of tokens, and the tokens in places control the operation of the transitions. Next, we give the definition about the enabled transition.

Definition 4.4. (Enabled) *Let $N = (P, T, F)$ is a P/T-net, $M = (N, \mu)$ is a marked P/T-net, and $\mu(p_i)$ represents the number of tokens in place $p_i (p_i \in P)$. Assume that there is a transition $t_i (t_i \in T)$, if $\forall p_i \in \bullet t_i$, $\mu(p_i) \geq \delta(p_i, t_i)$, then transition t_i is enabled, where $\delta(p_i, t_i)$ represents the number of directed arcs between p_i and t_i.*

The execution of a P/T-net depends on the firing of enabled transitions. The firing of a transition is to remove the tokens from its input places and generate new tokens

to its output places. As in Figure 5, transition A and B are both enabled since there are two tokens in place P_1, after firing A, one token would be removed from P_1, and P_2 would get a new token. When A and B are all fired, the marking of current P/T-net is $\mu = \{0, 1, 1, 0, 0\}$.

At present, most workflow systems offer and support standard building blocks such as the AND-split, AND-join, OR-split, and OR-join [17-20], which are used to model sequential, conditional, parallel, and iterative routing. Apparently, these routing of cases can be specified by P/T-nets. Tasks, also referred to activities in workflow, are modeled by transitions and casual dependencies are modeled by places and arcs [7]. Figure 6 shows the standard building blocks modeled by P/T-nets. An AND-split corresponds to a transition with more than two output places, and an AND-join corresponds to a transition with more than two input places. OR-split/OR-join corresponds to place with multiple outgoing/ingoing arcs.

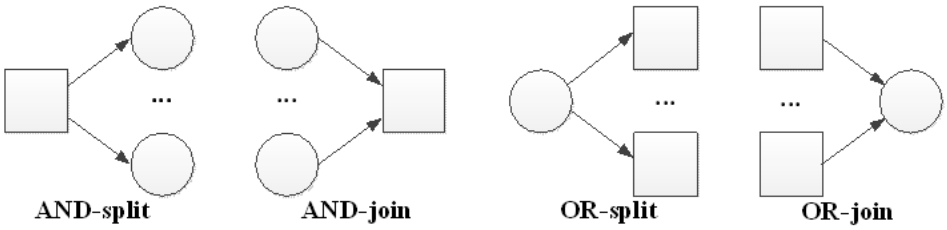

FIGURE 6. Standard blocks specified by P/T-net

In fact, a place corresponds to a condition which can be pre and/or post condition for tasks. In our work, we consider a rule task (i.e., rule set which used as a node in workflow to make decisions) as a subset of task mentioned above, in other words, a rule can be seen as a combination of place and transition, additionally, the conditions of a rule correspond to the place and the actions of a rule corresponds to the transition. Following definition is about what we call a rule.

Definition 4.5. (Rule) *A rule, denoted r, is a tuple (LHS, RHS), where:*

1. *LHS is a finite set of conditions in a rule, called the left hand side.*
2. *RHS is a finite set of actions in a rule, called the right hand side.*

In general, rules can be classified into different categories when they applied to different domain. What we concerned in this paper are business rules, and they are defined by organizations according to their business behaviors. For instance, a bank may define a rule, stating that if the client's credit risk is greater than 50% then refuse his loan.

Usually, both ***LHS*** and ***RHS*** involve some facts which are application data, the business rule engine function by having a working memory of them. By adding, removing or changing facts in the working memory, the rule engine can see which (if any) rules need triggering. Rules will be re-evaluated whenever the effect of a rule changes a state in the working memory. In α algorithm, we use the following definition of fact.

Definition 4.6. (Fact) *A fact, denoted f, is viewed as a tuple (τ, ε, π), where:*

1. *τ represents for the fact type ID.*
2. *ε represents for the ID of itself.*
3. *π represents for the updated frequency in a same case (i.e., a set of rule tasks used together to make decision in form of workflow).*

We regard case as the same concept of workflow instance. The fact types are user defined data types, like Java objects with fields or properties. A fact can be viewed as a

concrete assignment of values to the properties for a given fact type. The reason why we define fact as Definition 4.2 is that, the form of $f = (\tau, \varepsilon, \pi)$ can help us to analyze the firing relations of rules in rule flow log. Next, we define a rule flow log as follows.

Definition 4.7. (Rule flow trace, Rule flow log) *Let T be a set of rule tasks. $\sigma \in T^*$ is a rule flow trace and $\vartheta \in \rho(T^*)$ is a rule flow log, where $\rho(T^*)$ is the power set of T^*.*

The rule flow log contains some traces and each trace has at least one case. Note that the ordering of rules within a case is relevant, while the ordering of rules among cases is of no importance. In this paper, we abstract from the identity of cases and study the details of each rule task within them. There are lots of knowledge can be mined from the rule flow log, such as the attributes of a case and frequency of trace. In this paper, we mine the conditions which trigger the next business task. A key issue we must emphasize is that when dealing with noise, frequencies are of the utmost importance. However, we assume that there is no noise in the log, which is the same with α algorithm.

To find a workflow model on the basis of a rule flow log, the log should be analyzed for causal dependencies, e.g., if the fact used for firing a rule is the same fact updated by the rule before it in the same case, it is likely that there is a causal relation between both rules. We introduce these relations as follows.

Definition 4.8. (Fact-based firing relations) *Let ϑ be a rule flow log over T (i.e., $\vartheta \in \rho(T^*)$), FL(r)/FR(r) represents the facts set involved in LHS/RHS of rule r and $\pi(f)$ represent the updated frequency of fact f.*

Assume that there is a trace $\sigma = r_1 r_2 r_3 \ldots r_n$ that $\sigma \in \vartheta$ and $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n\}$.

1. $r_i \preccurlyeq_\vartheta r_j$ iff $i < j$ and $\forall k \in \{1, 2, \ldots, j\}, f \in FL(r_k)$: $\exists k = i \wedge \pi(f)$ is same.

2. $r_i \rightarrow_\vartheta r_j$ iff $i < j$ and $FR(r_i) \cap FL(r_j) \neq \varnothing$.

3. $r_i \prec_\vartheta r_j$ iff $i < j$ and $FL(r_i) \cap FL(r_j) \neq \varnothing$.

4. $r_i \succcurlyeq_\vartheta r_j$ iff $r_i, r_j \in \{r | (FL(r) = \varnothing) \vee (\forall k \in \{1, 2, \ldots, n\} : \neg\exists r \rightarrow_\vartheta r_k)\}$.

Assume that there is another trace σ' which shares some same rule tasks with σ'. Let $r_s \in \sigma' \cap \sigma$, $r_i \in \sigma$, $r_j \in \sigma$.

5. $r_i \ltimes_\vartheta r_j$ iff $(r_s \rightarrow_\vartheta r_i) \wedge (r_s \rightarrow_\vartheta r_j)$.

Note that, first four relations of above are based on an identical trace σ and the last one is based on two different traces (i.e., σ', σ) in rule flow log ϑ, that means we can find most relations through analyzing the facts of each rule in a single trace. In α-r algorithm, we rely on the above main relations. Relation $\preccurlyeq_\vartheta$ describes which rule tasks start first (i.e., rules triggered concurrently at beginning), we call this relation an AND-start block and it requires that the facts in LHS of first several rules shares same updated frequencies. Relation $\succcurlyeq_\vartheta$ is called AND-end and it meets two different circumstances: there is no fact involved in rule task or the facts involved in RHS of rule can't trigger any other rules. Relation $\rightarrow_\vartheta$ means the direct causal dependency which could be inferred by judging whether there is more than one same fact between RHS of a rule and LHS of another rule, if there is at least one same fact there is a direct causal relation between these two rules, or else there is not. The third relation $\prec_\vartheta$ gives pairs of rule tasks which are the start nodes for parallel branches and from a workflow point of view, these rules can be viewed as the nodes that generated by AND-split transition. As for the last relation $\ltimes_\vartheta$, since it is based on two different traces we call it OR-split block.

In order to reason about the quality of a process mining algorithm, it is necessary for us to make assumptions about the completeness of a rule flow log. For a complex process, lots of the traces will not suffice to discover the exact behavior of the process [7]. And fact-based firing relations are crucial information and principles for discovering process

from rule flow log, as a result of which, we assume the log to be complete with each relation.

Definition 4.9. (Complete rule flow log) *Let $N = (P, T, F)$ be a sound WF-net [7], and ϑ is a rule flow log of N iff $\vartheta \in \rho(T^*)$ and every trace $\sigma \in \vartheta$ is a firing sequence [7] of N starting in state [i] and ending in [o]. ϑ is a complete rule flow log of N iff:*

1. *For any rule flow log ϑ' of N, $(\preccurlyeq_{\vartheta'}, \rightarrow_{\vartheta'}, \prec_{\vartheta'}, \succcurlyeq_{\vartheta'}, \ltimes_{\vartheta'}) \subseteq (\preccurlyeq_{\vartheta}, \rightarrow_{\vartheta}, \prec_{\vartheta}, \succcurlyeq_{\vartheta}, \ltimes_{\vartheta})$.*
2. *For any rule task $r \in T$, there is a $r \in \vartheta$ such that $r \in \sigma$.*

This is the second important assumption besides no noise made in this paper. A rule flow log of a sound WF-net only contains rule executions behaviors that can be exhibited by the corresponding process. And from the above completeness definition, we can see that if all the rule tasks in T can be fired by some certain facts, the rule flow log is complete. If there is more than one rule task in T which could not be fired by any fact, the log will be incomplete. In order to discover the entire process structure, we need to conduct analysis over all possible traces in the log (i.e., complete) and merge and infer the final structure, however, as long as one real rule task is never be fired to execute or executed but was not logged (i.e., the log is incomplete), we can't measure the quality of the mining algorithm used on this kind of incomplete log. As far as our knowledge goes, the current process mining algorithms are mostly based on completeness of log. Since we have had many definitions about how to mining process from a complete rule flow log, we should consider the opposite, that is to say how we can validate the quality of the mined process model for a certain mining algorithm. Next, we define what accuracy means for the discovered process.

Definition 4.10. (Accuracy) *Let W be a workflow/process model discovered from rule flow log ϑ by mining algorithm, $Num(\sigma_\vartheta)$ represents the number of traces in log ϑ, $Num(\sigma_\vartheta^W)$ represents the number of traces of log ϑ which can be executed in W. The accuracy of W is:*

$$Accuracy = \frac{Num(\sigma_\vartheta^W)}{Num(\sigma_\vartheta)} \times 100\%$$

As mentioned in introduction section, process mining aims at discovering the underlying processes to help improving or rebuilding business processes. But validating the quality of the mined process should be done firstly before improving the current process. An important criterion for the quality of a mined process model is the consistency between the mined model and the traces in the rule flow log. Therefore, a standard check for a mined model is to try to execute all traces of the rule flow log in the discovered model [6]. If a trace that could not be executed successfully, there is a discrepancy between the log and the model, which means that the mined model is not accurate. From above formula, it is easy to infer that the accuracy is between 0 and 1which represents the worst (i.e., incorrect) and the best (i.e., correct) respectively. We can use the accuracy to measure the quality of a mining algorithm.

4.2. **Algorithm.** On the foundation of the aforementioned preliminaries, we now present our α-r algorithm for discovering process from rules through the log of rule engine. Specially, according to the fact-based firing relations (Definition 4.8), the algorithm uses the basic ideas showed in Figure 7.

The result of α-r algorithm, as mentioned before, is a process which modeled by P/T-net in this paper and the discovered process is composed of blocks mostly showed in Figure 7. On the whole, based on Definition 4.8, there are two main phases in the core of algorithm for mining the process. One is called the analysis phase, it analyzes the cases

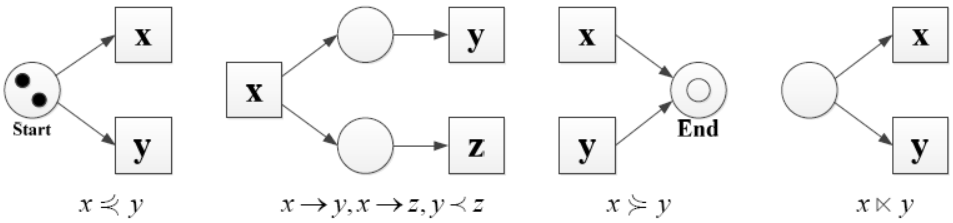

FIGURE 7. Basic ideas modeled by P/T-net

TABLE 1. The α-r algorithm

Input: rule flow log ϑ
Output: process β modeled by P/T-net
For each trace σ in rule flow log ϑ, do
For each rule task t in trace σ, do
• If t is the first rule task in σ then t is the start of process β
• Get the facts involved in LHS of rule r in t: *FL(r)*
• Get the facts involved in RHS of rule r in t: *FR(r)*
• If the *FL(r)* is empty
• Rule task t is the end of process β
For each rule task t' after t, do
• Get the facts involved in LHS of rule r' in t': *FL(r')*
• Get the facts involved in RHS of rule r' in t': *FR(r')*
• If the updated frequencies in *FL(r)* and *FL(r')* are all the same
• There is $\preccurlyeq$ relation between t' and t
• If *FR(r)* and *FL(r')* share at least one same fact f
• There is $\to$ relation between t' and t
• If *FL(r)* and *FL(r')* share at least one same fact f
• There is $\prec$ relation between t' and t
• If *FR(r)* never trigger any rules after t then t is one of the ends of process β
Mine $\succcurlyeq$ relation
Merge the results mined by above steps and discover $\ltimes$ relation
Model the process β above by P/T-net

within rule flow log to find out the fact-based firing relations (i.e., $\preccurlyeq, \to, \prec, \succcurlyeq$). The other is called the merging phase, the algorithm merges the results of the former phase to mine the (i.e., $\ltimes$) relation. Table 1 is the pseudo code of α-r algorithm.

The detail of α-r algorithm showed in Table 1 is the whole process for discovering. The input for α-r algorithm is the rule flow log, and the output is a process modeled by P/T-net. Note that, the analysis phase is the most important and complicated since it focuses on the inner relations among rule tasks within each trace. As each rule task in each trace within the rule flow log is processed by fact, the firing relations can be mined and discovered. At first, we should find out the start rule tasks of the process, then, we analyze the following rule tasks to find out the "$\succcurlyeq, \to, \prec$" relations. The last step in this phase is to identify the end rule tasks and mine "$(O(p) \bullet O(q) \bullet O(r))$" relation. After all traces have been considered and analyzed, the OR-split and OR-join blocks are discovered and integrated into the ultimate process (workflow) model in the merging phase. Apparently, because of the iterations of traces and facts, the analysis phase consumes more time than the merging phase in α-r algorithm. Totally, the complexity of the algorithm is equal to $(O(p) \bullet O(q) \bullet O(r))$, where $O(p)$ represents the time consumed by iterating traces to

mine "$\ltimes$" relation, $O(q)$ represents the second iteration of rule tasks in one trace, $O(r)$ represents the iteration of every rule task behind the current one. In the following case study of this algorithm, we will see how it works in real applications and what specific result it generates.

4.3. **Case study.** To illustrate the whole process of α-r algorithm, we consider the rule flow log shown in Table 2. And in order to drive it home, we sort the content of the log by case ID. From Table 2, we can see that there are twelve rule tasks (refer to event in XES file [12]) and two cases (traces) which represent two instances of a workflow form. In each rule task, there is a corresponding rule with its LHS and RHS involved with facts which are most important information. The fact is represented by fact type ID, ID of itself and the updated frequency. For example, f(3, 2, 1) means the fact type ID is 3, fact ID is 2 and it is updated once.

TABLE 2. An example rule flow log

Case	Rule	LHS (Conditions)	RHS (Actions)
1	A	f(1,1,0), f(2,1,0)	Update f(2,1,1)
1	B	f(2,1,1)	Update f(3,1,1)
1	C	f(3,1,0), f(2,1,1)	Update f(1,1,1)
1	D	f(3,1,1)	Update f(3,1,2)
1	F	f(3,1,2), f(2,1,1)	Update f(2,1,2), f(3,1,3)
1	G	f(1,1,1), f(2,1,2), f(3,1,3)	Return true
2	A	f(1,2,0), f(2,2,0)	Update f(2,2,1)
2	B	f(2,2,1)	Update f(3,2,1)
2	C	f(3,2,0), f(2,2,1)	Update f(1,2,1)
2	E	f(3,2,1)	Update f(3,2,2)
2	F	f(3,2,2), f(2,2,1)	Update f(2,2,2), f(3,1,3)
2	G	f(1,2,1), f(2,2,2), f(3,2,3)	Return true

As for case 1, there are three fact types totally. Rule A is matched with facts f(1,1,0) and f(2,1,0), then the RHS (actions) of the rule is fired which means it will update the fact f(2,1,0) to f(2,1,1). Next, f(2,1,1) fires rule B and with the help of f(3,1,0) rule C is also fired in parallel. As a result, f(1,1,0) and f(3,1,0) are updated to f(1,1,1) and f(3,1,1) respectively. And then, rule D is fired by f(3,1,1) which updated to f(3,1,2). In 5th rule task, f(3,1,2) and f(2,1,1) together fire the rule F, and they are updated to f(2,1,2), f(3,1,3). At last, the combination of f(1,1,1), f(2,1,2) and f(3,1,3) fire rule G which is the end of this case since there is no fact involved in RHS of it.

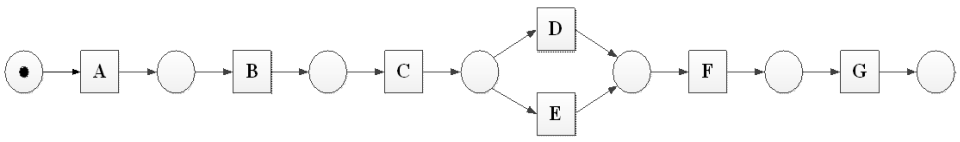

FIGURE 8. The process model discovered by α algorithm

We mined this log (Table 2) by classic α algorithm and α-r algorithm. Figure 8 and Figure 9 respectively show the discovered process model. In Figure 8, the relation between rule B and rule C is direct causal dependency, but in Figure 9 there is a parallel relation between them. This is because α algorithm merely depends on the log based ordering relations [7] without analyzing the facts. This phenomenon suggests that the process

model can be quite different in different perspective (i.e., traditional perspective and rule firing perspective). But it is meaningful in real life, as mentioned in introduction and motivation section of this paper, we can combine and analyze these different models with actual circumstances to design or redesign a workflow model which can be more objective to real worlds. Besides, since each rule consists of LHS and RHS, we can obtain the conditions for routing of a process easily, and this is more specific for users to understand the details of the whole process.

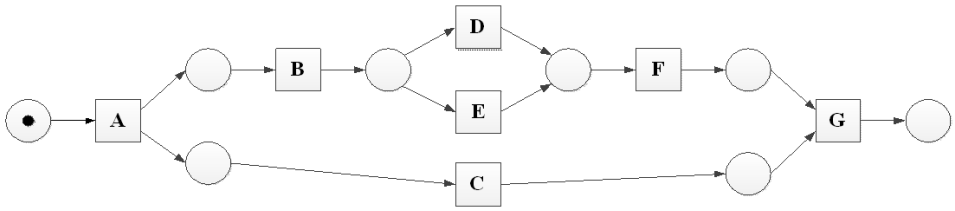

FIGURE 9. The process model discovered by α-r algorithm

5. **Experimental Evaluation.** Since α-r algorithm is based on the log of rule engine and it analyzes each rule in each trace, there are few factors that influence the whole performance. The main goal for our evaluation is to see the performance of the α-r algorithm, in terms of rule task amount and fact type amount.

5.1. **Implementation and setup.** Base on the framework mentioned in third section of this paper, we implement the α-r algorithm and evaluate the performance of whole procedure for mining a workflow from the log of rule engine. Figure 10 shows the implementation result for case study in Section 4.3.

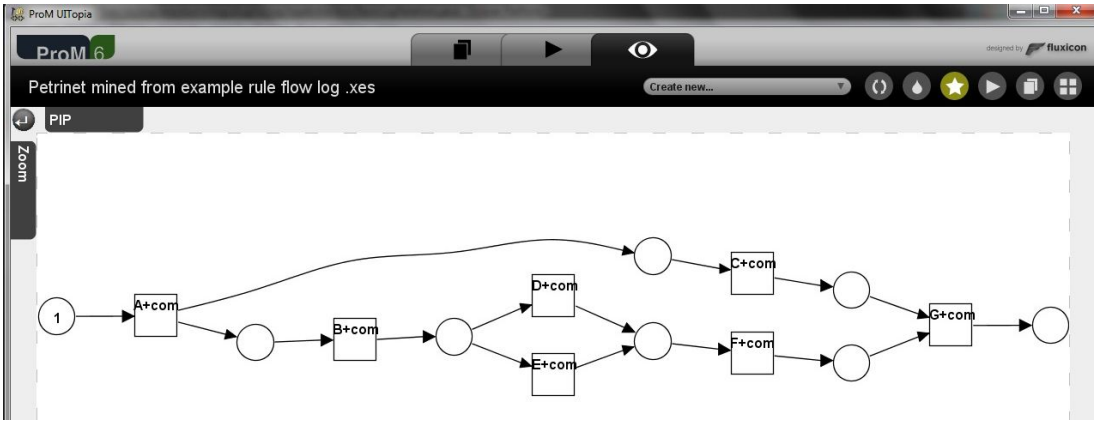

FIGURE 10. The implementation result for case study

In order to evaluate the performance, we simulate the rule tasks and create the rule flow log automatically through a simulation program. In this way, we can control the different factors so that it is much easier for us to analyze the performance. The detail of test environment is showed in Table 3.

Table 4 summarizes the execution time of the mining procedure with varying number of rule tasks which represented by $\#\boldsymbol{T}$. Here $\#\boldsymbol{D}$ means the duration of the procedure, and it is measured in milliseconds. The amount of fact type in each test is same, here we base on three fact types.

TABLE 3. The test environment configuration

SAMSUNG	***P408-DA0H***		
CPU:	Pentium(R) Dual-Core CPU T4200 @2.00 GHz	HD:	160 GB 5400 rpm
Memory:	2 GB, DDRII 800MHz	OS:	Windows XP Professional 2002 Service Pack 3

TABLE 4. Duration with varying number of rule tasks

#T	12	24	48	102	198	498	1002	1998
#D (ms)	47	62	79	94	125	196	321	484

TABLE 5. Duration with varying number of fact types

#F	10	20	50	100	200	500	1000	1200
#D (ms)	63	78	96	178	512	2203	8199	11391

Since the α-r algorithm must mine several relations which are facts based in traces of the log, the fact should be counted as a factor for performance. Table 5 shows the duration of mining procedure with varying number of fact types in totally twelve rule tasks. Here, ***#F*** is used to represent the number of fact types.

5.2. **Performance analysis.** Table 4 and Table 5 respectively correspond to Figure 11 and Figure 12. From Figure 11, we can find that the α-r algorithm's execution time for mining is growing as the number of rule tasks increases. But the trend of increasing is not linear, when rule tasks number increase to 500 around, the duration of mining becomes longer. And this is likely because of the loops for mining those relations in the traces.

As for Figure 12, we can also see that the execution time for mining is growing as the number of fact types increases. For the similar reason, the trend is not linear. But, the duration is much longer than that in Figure 11, which means the more fact types there are the more time would be cost for mining processes.

From the experimental evaluation, we can infer that the number of rule tasks contributes less for the performance comparing with the number of facts types. And in general, since there are not too many fact types in real business and the number of rule tasks is typically less than 100 which does not depend on the size of the log [7], considering the acceptable duration showed in above tables and figures, it is clear that the α-r algorithm is suitable for practical uses and its complexity is not a bottleneck for large-scale application.

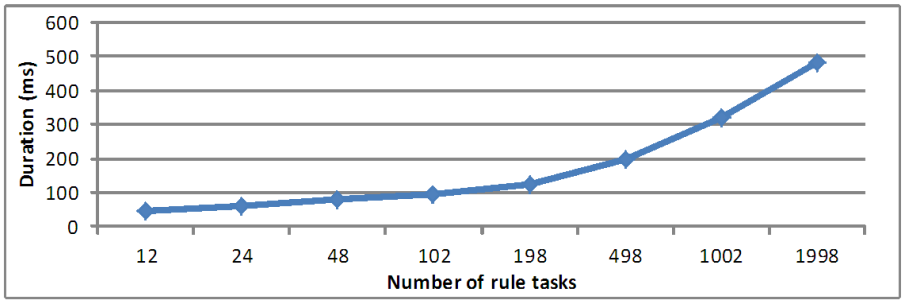

FIGURE 11. Duration for different number of rule tasks

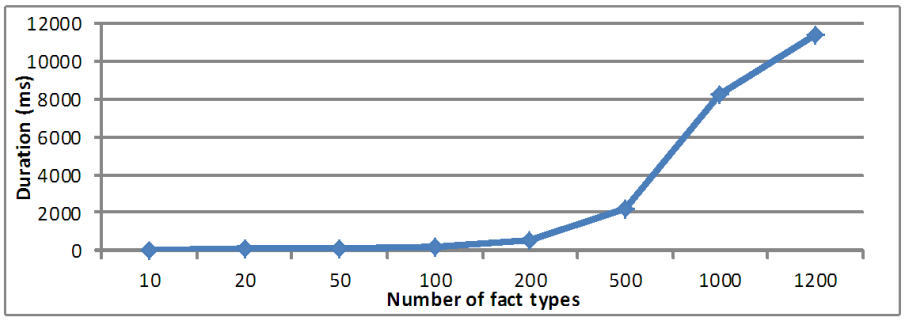

FIGURE 12. Duration for different number of rule tasks

5.3. **Accuracy analysis.** We performed the accuracy test on JTangFlow [21] which is a workflow product integrated with rule engine. Beforehand, we prepare some sample complete rule execution logs and their corresponding process mined by α-r algorithm. In order to validate the accuracy of mined process, we have to execute all traces of log in mined process first, and then we compute the accuracy according to Definition 4.10. Since we must use facts set to test the process execution, how to select the facts becomes a problem. Generally, there are two different cases:

- **Sample logs facts:** We can get the facts logged in sample logs since there is enough useful information recorded when the real process executed. For example, the fact ID in fact base. In this case, we use the sample logs facts to test the mined process, and it turns out to be accurate for α-r algorithm and the value of accuracy is 1.
- **Random facts:** In real applications, people are not so easy to get the sample logs facts even if they know the fact ID. They are probably refused to query from fact base or there is no access at all for them. In this case, in order to fully execute the entire structure of the mined processes, we have to automatically produce the facts set in large scale with random attribute value and try them for many times. Though this is a time consuming and boring job for executing all traces in sample logs, fortunately, given the sample logs and mined process, we find it is accurate for α-r algorithm and the value of accuracy is 1.

The reasons why both cases can prove the value of accuracy to be 1 for α-r algorithm are assumptions we made in this paper: there is no noise in rule flow log and the rule flow log is complete. They ensure the correctness and entireness of the discovered process model. However, it is unrealistic to validate all potential traces for complex process models since there are not only lots of different blocks but also the combinations for these blocks is in a large scale. Anyway, we can infer the trend and to induce the conclusion. As for the second case above, especially, from Figure 13 we can see that the more attempts conducted for executing the mined process, the value of accuracy for α-r algorithm is inclined to 1.

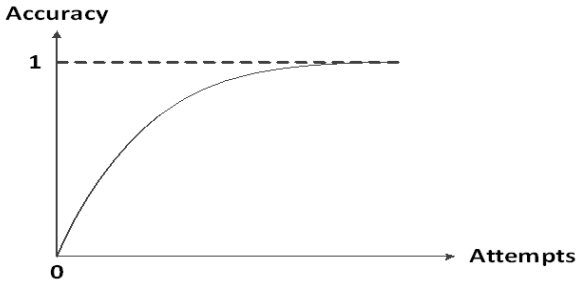

FIGURE 13. Variation of accuracy with execution attempts

6. **Related Work.** The idea of process mining was previously investigated in contexts such as workflow management and software engineering [22-24]. In the context of workflow management, Agrawal et al. [22] introduced the first algorithm for mining workflow logs. This work is based on workflow graphs, which are inspired by workflow products such as IBM MQSeries workflow and InConcert. Two problems are defined in this paper, the first is to find a workflow graph generating events appearing in a given workflow log, the second problem is to find the definitions of edge condition. The author proposed an approach for solving the first problem, and the approach is quite different from others since it does not identify the nature (AND or OR) of joins and splits because of the nature of the workflow. Cook and Wolf propose alternative methods for process discovery in case of software engineering, focusing on sequential [23] and concurrent processes [25]. In [23], they described three methods for process discovery: neural networks, purely algorithmic approach and Markovian approach. The purely algorithmic approach builds a finite state machine where states are fused if their futures are identical, and the Markovian approach uses a mixture of algorithmic and statistical methods and is able to deal with noise. As a result, the authors in [23] consider these two methods the most promising approaches. Herbst and Karagiannis use a hidden Markov model in the context of workflow management, focusing on sequential [26,27] and concurrent processes [24]. In Mauster, Aalst, et al. [28], a technique for discovering workflow processes in hospital data is presented.

α-r algorithm in this paper is related to and share some certain theories or foundations with the α algorithm presented in [7,29-33]. In [7] it is proven that the α algorithm is able to construct a corresponding P/T-net. In [29-33], in order to address the noise in logs, a heuristic approach is used to construct "dependency/frequency tables" and "dependency/frequency graphs" as an intermediate step for mining correct processes. The authors of [34] proposed a so called "β-*algorithm*" which can be seen as an extension of the α algorithm. In this algorithm, two kinds of event types, i.e., START and COMPLETE, are considered. Using these two types of events it is possible to see if occurrences of tasks overlap. The methods for process mining are various, machine learning tools that can mine workflow models are of great interest and relatively unexplored. In Ricardo, et al. [35], the author presented an algorithm for learning workflow graphs that makes use of a coherent probability model. Another work for dealing with the noise is presented in [36]. In this paper, the authors proposed a method that constructs the process model from process log data, by determining the relations between process tasks. And to predict these relations, they employed machine learning technique to induce rule sets which are from process logs. Greco et al. [37] approach the problem using clustering techniques. All the work above is based on the process logs, and we didn't find any other methods are special for the circumstances presented in our paper.

7. **Conclusion and Future Work.** We have presented an approach called α-r algorithm in this paper. The distinguishing feature of the α-r algorithm is that it exploits the execution logs of business rule engine, which integrated to a workflow to make decisions for organizations. In order to implement the whole procedure for mining, we used the **Prom** [12] tools and designed a framework. During the implementation, we elaborately analyzed the relations $(\preccurlyeq, \rightarrow, \prec, \succcurlyeq, \ltimes)$ and illustrated the algorithm with a case study. From the experimental evaluation we can conclude that the α-r algorithm is appropriate for practical uses.

Although our work is based on the rule flow(i.e., rules organized in a workflow form) scenario, the α-r algorithm aims at discovering process from rules through the log of rule engine, it can be used in the circumstances that where business rule engine adopted only

or integrated to workflow related systems. To the best of our knowledge, this is the first approach with such a property.

In general, the α-r algorithm can be seen as an extension of the α algorithm. They share some same theories and foundations, likewise, some of the known problems of the α algorithm, e.g., short-loops and noises also exist in α-r algorithm. However, we didn't resolve or consider them which make our algorithm imperfect. Additionally, there is room for improvement of the performance since the duration for mining is not a linear increase. So our future work will focus on these aspects. Dig into the known and unknown problems of α-r algorithm and try to perfect the theories and foundation of it. Simultaneously, optimize the algorithm to get a better performance. Last but not least, referring some existing research work on dealing noise and do best to make α-r algorithm be capable of mining.

Acknowledgments. This work has been supported by following fundations: National Key Science and Technology Research of China (2011ZX01039-001-002), National Natural Science Foundation of China (61170033), Research Fund for the Doctoral Program by Ministry of Education of China(No. 20110101110066), Science and technology Program of Zhejiang Province (Nos. 2011C14004, 2009C11027).

REFERENCES

[1] P. Jackson, *Introduction to Expert Systems*, 3rd Edition, Addison-Wesley Publishing Company, 1990.
[2] *Business Rules Engine*, http://en.wikipedia.org/wiki/Business_rules_engine, 2011.
[3] C. L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem, *Artificial Intelligence*, vol.19, no.1, pp.17-37, 1982.
[4] D. Batory, The LEAPS algorithms, *Technical Report 94-28 of Department of Computer Sciences*, University of Texas at Austin, 1994.
[5] W. M. P. van der Aalst and A. J. M. M. Weijters, Process mining: A research agenda, *Special Issue of Computers in Industry*, vol.53, no.3, 2004.
[6] 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*, vol.47, no.2, pp.237-267, 2003.
[7] W. M. P. van der Aalst, A. J. M. M. Weijters and L. Maruster, Workflow mining: Discovering process models from event logs, *QUT Technical Report, FIT-TR-2003-03*, Queensland University of Technology, Brisbane, 2003 (Accepted for publication in *IEEE Trans. on Knowledge and Data Engineering*).
[8] A. Goh, Y.-K. Koh and D. S. Domazet, ECA rule-based support for workflows, *Artificial Intelligence in Eng.*, vol.15, no.1, pp.37-46, 2001
[9] *Drools*, http://en.wikipedia.org/wiki/Drools, 2011.
[10] *ILOG*, http://en.wikipedia.org/wiki/ILOG, 2011.
[11] *What is a Rule Flow*, http://publib.boulder.ibm.com/infocenter/brjrules/v7r0m2/index.jsp?topic=/com.ibm.websphere.ilog.jrules.doc/Content/Business_Rules/Documentation/_pubskel/JRules/ps_JRules_Global409.html, 2011.
[12] *ProM 6*, http://www.promtools.org/prom6/, 2011.
[13] J. Desel and J. Esparza, Free choice petri nets, *Cambridge Tracts in Theoretical Computer Science*, vol.40, 1995.
[14] T. Murata, Petri nets: Properties, analysis and applications, *Proc. of IEEE*, vol.77, no.4, pp.541-580, 1989.
[15] W. Reisig and G. Rozenberg (eds.), Lectures on petri nets I: Basic models, in *LNCS*, vol.1491, Springer-Verlag, Berlin, 1998.
[16] *Business Process Modeling Notation*, http://en.wikipedia.org/wiki/Business_Process_Modeling_Notation, 2011.
[17] W. M. P. van der Aalst and K. M. van Hee, *Workflow Management: Models, Methods, and Systems*, MIT Press, Cambridge, 2002.
[18] L. Fischer, *Workflow Handbook*, Workflow management coalition, 2001.

[19] S. Jablonski and C. Bussler, *Workflow Management: Modeling Concepts, Architecture, and Implementation*, Int'l Thomson Computer Press, London, 1996.
[20] F. Leymann and D. Roller, *Production Workflow: Concepts and Techniques*, Prentice-Hall PTR, New Jersey, 1999.
[21] *JTang*, http://www.jtang.com.cn/index.html, 2011.
[22] R. Agrawal, D. Gunopulos and F. Leymann, Mining process models from work-flow logs, *Proc. of the 6th International Conference on Extending Database Technology*, pp.469-483, 1998.
[23] J. E. Cook and A. Wolf, Discovering models of software processes from event-based data, *ACM Trans. on Software Engineering and Methodology*, vol.7, no.3, pp.215-249, 1998.
[24] J. Herbst, Dealing with concurrency in workflow induction, *European Concurrent Engineering Conference*, Society of Computer Simulation Europe, 2000.
[25] J. E. Cook and A. Wolf, Event-based detection of concurrency, *Proc. of the 6th International Symposium on the Foundations of Software Engineering*, pp.35-45, 1998.
[26] J. Herbst and D. Karagiannis, Integrating machine learning and workflow management to support acquisition and adaptation of workflow models, *International Journal of Intelligent Systems in Accounting, Finance and Management*, vol.9, pp.67-92, 2000.
[27] J. Herbst, Inducing workflow models from workflow instance, *Proc. of the 6th European Concurrent Engineering Conference*, Society of Computer Simulation Europe, pp.175-182, 2000.
[28] L. Maruster, W. van der Aalst, T. Weijters, A. van den Bosch and W. Daelemans, Automated discovery of workflow models from hospital data, *Proc. of the EDAI Workshop on Knowledge Discovery from Temporal and Spatial Data*, pp.32-37, 2001.
[29] W. M. P. van der Aalst and B. F. van Dongen, Discovering workflow performance models from timed logs, *International Conference on Engineering and Deployment of Cooperative Information Systems, LNCS*, vol.2480, pp.45-63, 2002.
[30] A. J. M. M. Weijters and W. M. P. van der Aalst, Process mining: Discovering workflow models from event-based data, *Proc. of the 13th Belgium-Netherlands Conference on Artificial Intelligence*, pp.283-290, 2001.
[31] A. J. M. M. Weijters and W. M. P. van der Aalst, Rediscovering workflow models from event-based data, *Proc. of the 11th Dutch-Belgian Conference on Machine Learning*, pp.93-100, 2001.
[32] A. J. M. M. Weijters and W. M. P. van der Aalst, Workflow mining: Discovering workflow models from event-based data, *Proc. of the ECAI Workshop on Knowledge Discovery and Spatial Data*, pp.78-84, 2002.
[33] A. J. M. M. Weijters and W. M. P. van der Aalst, Rediscovering workflow models from event-based data using little thumb, *Integrated Computer-Aided Engineering*, vol.10, no.2, pp.151-162, 2003.
[34] L. J. Wen, J. Wang, W. M. P. van der Aalst, Z. Wang and J. Sun, A novel approach for process mining based on event types, *Journal of Intelligent Information Systems*, vol.32, no.2, pp.163-190, 2009.
[35] R. Silva, J. Zhang and J. G. Shanahan, Probabilistic workflow mining, *Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp.275-284, 2005.
[36] L. Maruster, A. J. Weijters, W. M. Aalst and A. Bosch, A rule-based approach for process discovery: Dealing with noise and imbalance in process logs, *Data Mining and Knowledge Discovery*, vol.13, no.1, pp.67-87, 2006.
[37] G. Greco, A. Guzzo, L. Pontieri and D. Sacca, Mining expressive process models by clustering workflow traces, *Proc. of the 8th PAKDD*, 2004.