Discovery of Cancellation Regions within Process Mining Techniques*

A. A. Kalenkova and I. A. Lomazova

[1] National Research University Higher School of Economics, Moscow, Russia
{akalenkova, ilomazova}@hse.ru
[2] Program Systems Institute of the Russian Academy of Sciences,
Pereslavl-Zalessky, Russia

Abstract. Process mining is a relatively new field of computer science which deals with process discovery and analysis based on event logs. In this work we consider the problem of discovering workflow nets with cancellation regions from event logs. Cancellations occur in the majority of real-life event logs. In spite of huge amount of process mining techniques little has been done on cancellation regions discovery. We show that the state-based region algorithm gives labeled Petri nets with overcomplicated control flow structure for logs with cancellations. We propose a novel method to discover cancellation regions from the transition systems built on event logs and show the way to construct equivalent workflow net with reset arcs to simplify the control flow structure.

1 Introduction

Process mining technology [1] provides us with a variety of methods for discovering business processes from event logs. These methods are commonly used when formal process description is not available or description does not correspond to the real-life process behavior. One of the goals of the process discovery is the retrieving of readable process models. The majority of real-life event logs contain information about cancellations which occur during the process execution. These cancellations can be expressed by means of workflow languages (BPMN [2], YAWL [3]) and formal models such as Reset workflow nets (RWF-net) [4]. In [5] an approach for discovery of cancellations from event log has been presented. This approach constructs a workflow net (WF-net) with the state-based region algorithm [6]. After that it replays the log on this model, for all remaining tokens reset arcs are added to the WF-net, as a result RWF-net is produced.

State-based region algorithm [6] used within process mining techniques was developed on the basis of well-known algorithms for the construction of a Petri net (PN) from a transition system (TS) [7–9], taking into account that minor transformations of TS will allow to retrieve WF-net instead of arbitrary PN. An algorithm was given by [7] to synthesize a PN from the elementary transition

* This study was carried out within the National Research University Higher School of Economics' Academic Fund.

system (ETS) in such a manner that reachability graph (RG) of the PN is isomorphic to the TS (or the minimized TS if it is not minimal). Algorithm proposed in [8, 9] generate a labeled PN for an arbitrary TS such that RG of the PN is isomorphic or split-isomorphic [8] to the TS or its minimization. This algorithm effectively checks whether TS is elementary (by verifyng the excitation closure property [8]), if TS is not elementary then TS and target PN are splitted.

In this paper we propose a method which not just adds reset arcs, but makes a target model more compact and readable. We prove that the straightforward applying of a state-based region algorithm to an event logs with cancellations leads to the generation of a labeled Petri net with overcomplicated control flow structure. Then we present an algorithm for discovering cancellations and constructing an RWF-net with a more compact and transparent structure. We prove correctness of the proposed algorithm.

The paper is organized as follows. In Section 2 a motivating example of the booking process with cancellations is presented, it gives a ground for the development of a novel cancellation discovery method. Section 3 contains some basic definitions and notions, including logs, Petri nets and transition systems. In Section 4 we formally prove that cancellations in a log in the presence of parallel branches lead to the generation of labeled Petri nets with complicate control-flow structure. We also present an algorithm for construction of a RWF-net from TS and give the proof of the algorithm correctness. Section 5 contains some conclusions.

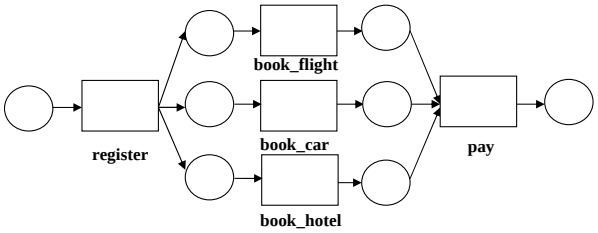

Fig. 1. A WF-net for a regualer booking process (no cancellations)

2 Motivating example

Let us consider a simple model of booking the trips process from [5]. One needs to book a hotel, a car and a flight (Fig. 1). This model formalizes the booking process which can't be canceled, while a real-life process might be canceled in consequence of internal booking errors. If we take a look at a log of some real-life booking process, we may notice that it contains traces of process execution

failures along with traces of standard booking process executions. A sample of such a real-life log L is presented in Fig. 2.

$$
\begin{aligned}
L = \{ & < register, book\_flight, book\_hotel, book\_car, pay >, \\
& < register, book\_flight, book\_car, book\_hotel, pay >, \\
& < register, book\_car, book\_flight, book\_hotel, pay >, \\
& < register, book\_car, book\_hotel, book\_flight, pay >, \\
& < register, book\_hotel, book\_car, book\_flight, pay >, \\
& < register, book\_hotel, book\_flight, book\_car, pay >, \\
& < register, book\_flight, book\_hotel\_NOK, cancel >, \\
& < register, book\_flight, book\_car, book\_hotel\_NOK, cancel >, \\
& < register, book\_car, book\_flight, book\_hotel\_NOK, cancel >, \\
& < register, book\_car, book\_hotel\_NOK, cancel >, \\
& < register, book\_hotel\_NOK, cancel >\}.
\end{aligned}
$$

Fig. 2. An event log for booking process with cancellations

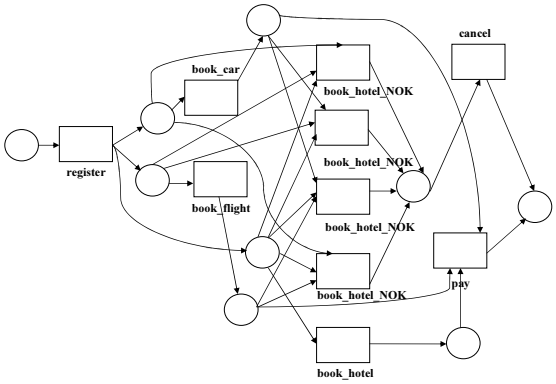

Fig. 3. A WF-net for booking process with cancellations discovered from event log in Fig. 2 by the standard region-based method.

In this log an additional event with the label 'book_hotel_NOK' occurs and denotes the situation when the booking hotel step failures. The 'book_hotel_NOK' event causes cancellation of parallel branches of booking a car and a flight and execution of a catching task (which is represented in the log as an event with the

label 'cancel'). Applying the state-based regions method for process discovery to this sample log gives us a labeled WF-net with a complicated control-flow structure (Fig. 3). This model is rather confounded. The clear structure of the core process (Fig. 1) can be hardly recognized. So, it is very important to find a method for discovering clear and readable workflow net with cancellations.

3 Logs, Petri nets and Transition systems. Definitions

Let S be a finite set. A *multiset* m over a set S is a mapping $m : S \to Nat$, where Nat is the set of natural numbers (including zero), i.e. a multiset may contain several copies of the same element.

For two multisets m, m' we write $m \subseteq m'$ iff $\forall s \in S : m(s) \leq m'(s)$ (the inclusion relation). The sum of two multisets m and m' is defined as usual: $\forall s \in S : (m + m')(s) = m(s) + m'(s)$, the difference is a partial function: $\forall s \in S$ such that $m(s) \geq m(s') : (m - m')(s) = m(s) - m'(s)$. By $\mathcal{M}(S)$ we denote the set of all finite multisets over S. Non-negative integer vectors are often used to encode multisets. Actually, the set of all multisets over finite S is a homomorphic image of $Nat^{|S|}$.

Definition 1 (Event log). *Let A be a set activities. A trace σ can be described as a sequence of activities, i.e., $\sigma \in A^*$. An event log L is a multiset of traces, i.e., $L \in \mathcal{M}(A^*)$.*

Definition 2 (Petri net). *Let P and T be disjoint sets of* places *and* transitions *and $F : (P \times T) \cup (T \times P) \to Nat$. Then $N = (P, T, F)$ is a* Petri net.

Let Σ be a finite alphabet. A labeled PN is a PN with a labeling function $\lambda : T \to \Sigma$ which maps every transition to a symbol (called a label) from Σ.

A *marking* in a Petri net is a function $m : P \to Nat$, mapping each place to some natural number (possibly zero). Thus a marking may be considered as a multiset over the set of places. Pictorially, P-elements are represented by circles, T-elements by boxes, and the flow relation F by directed arcs. Places may carry tokens represented by filled circles. A current marking m is designated by putting $m(p)$ tokens into each place $p \in P$.

For a transition $t \in T$ an arc (x, t) is called an *input arc*, and an arc (t, x) — an *output arc*; the *preset* ${}^\bullet t$ and the *postset* $t^\bullet$ are defined as the multisets over P such that ${}^\bullet t(p) = F(p, t)$ and $t^\bullet(p) = F(t, p)$ for each $p \in P$.

A transition $t \in T$ is *enabled* in a marking m iff $\forall p \in P\ m(p) \geq F(p, t)$. An enabled transition t may *fire* yielding a new marking $m' =_{\text{def}} m - {}^\bullet t + t^\bullet$, i. e. $m'(p) = m(p) - F(p, t) + F(t, p)$ for each $p \in P$ (denoted $m \xrightarrow{t} m'$, $m \xrightarrow{\lambda(t)} m'$, or just $m \to m'$). We say that m' is reachable from m iff there is a (possibly empty) sequence of firings $m = m_1 \to \cdots \to m_n = m'$ and denote it by $m \xrightarrow{*} m'$.

Workflow nets (WF-nets) is a special subclass of Petri nets designed for modeling workflow processes. A workflow net has one initial and one final place, and every place or transition in it is on a directed path from the initial to the final place.

Definition 3 ((Labeled) workflow net). *A (labeled) Petri net N is called a (labeled)* workflow net (WF-net) *iff*

1. *There is one source place $i \in P$ and one sink place $f \in P$ s. t. ${}^\bullet i = f^\bullet = \emptyset$;*
2. *Every node from $P \cup T$ is on a path from i to f.*
3. *The initial marking in N contains the only token in its source place.*

By abuse of notation we denote by i both the source place and the initial marking in a WF-net. Similarly, we use f to denote the final marking in a WF-net N, defined as a marking containing the only token in the sink place f. Fig. 1 gives an example of a WF-net.

Definition 4 (Reachability graph). *A reachability graph (RG) for a PN N is a graph with vertices corresponding to markings in N and with arcs defined as follows: (m_1, m_2) is an arc in the RG iff $m_1 \to m_2$ in N.*

For a labeled PN its RG has arcs labeled with the corresponding transitions labels.

Definition 5 (Transition system). *A* transition system *(TS) is a tuple $TS = (S, E, T, s_{in})$, where S is a finite non-empty set of states, E is a set of events, $T \subset S \times E \times S$ is a transition relation, and s_{in} is an initial state. Elements of T are called transitions and (by abuse of notation) will be denoted by $s \xrightarrow{e} s'$. A state s is reachable from a state s' iff there is a possibly empty sequence of transitions leading from s to s' (denoted by $s \xrightarrow{*} s'$). Each TS must satisfy the following basic axioms:*

1. *No self-loops: $\forall (s \xrightarrow{e} s') \in T : s \neq s'$;*
2. *No multiple arcs between a pair of states: $\forall (s \xrightarrow{e_1} s_1), (s \xrightarrow{e_2} s_2) \in T : [s_1 = s_2$ implies $e_1 = e_2]$;*
3. *Every event has an occurrence: $\forall e \in E : \exists (s \xrightarrow{e} s') \in T$;*
4. *Every state is reachable from the initial state: $\forall s \in S : s_{in} \xrightarrow{*} s$.*

We write $s \xrightarrow{e}$, or $\xrightarrow{e} s$ iff $\exists s' : s \xrightarrow{e} s'$, or $\exists s' : s' \xrightarrow{e} s$ correspondingly.

Now we define the notion of a region.

Definition 6 (Region). *Let $TS = (S, E, T, s_{in})$ be a transition system and $S' \subseteq S$ be a subset of states. S' is a region iff for each event $e \in E$ one of the following conditions hods:*

- *all the transitions $s_1 \xrightarrow{e} s_2$ enter S', i.e. $s_1 \notin S'$ and $s_2 \in S'$,*
- *all the transitions $s_1 \xrightarrow{e} s_2$ exit S', i.e. $s_1 \in S'$ and $s_2 \notin S'$,*
- *all the transitions $s_1 \xrightarrow{e} s_2$ do not cross S', i.e. $s_1, s_2 \in S'$ or $s_1, s_2 \notin S'$.*

Each TS has two *trivial regions*: the set of all states, and the empty set. For each state $s \in S$ we define the set of non-trivial regions, containing s (denoted by R_s). A region r' is said to be a subregion of a region r iff $r' \subseteq r$. A region r is called a minimal region iff it does not have any other subregions. A region r is a *pre-region* of an event e iff there is a transition labeled with e which exits r.

Now we define a notion of elementary transition system [8, 9].

Definition 7 (Elementary transition system(ETS)). *A* $TS = (S, E, T, s_{in})$ *is called elementary iff in addition to* 1-4 *it satisfies the following two axioms:*

5. *State separation property: two different states must belong to different sets of regions:*
   $\forall s, s' \in S : [(R_s = R_{s'})$ *implies* $(s = s')]$*;*
6. *Forward closure property: if state* s *is included in all pre-regions of event* e*, then* e *must be enabled by* s*:*
   $\forall s \in S \forall e \in E : [({}^{o}e \subseteq R_s)$ *implies* $(s \xrightarrow{e})]$.

A set S of states is called a *generalized excitation region* for an event a (denoted by $GER(a)$) iff S is a maximal (a maximal connected) set of states such that for every state $s \in S$ there is a transition $s \xrightarrow{a}$. An *excitation closure* condition is satisfied iff for each event $a : \bigcap_{r \in {}^{o}a} = GER(a)$.

4 Discovering a WF-net with cancellation regions

In this section we present a new method for process discovering, which allows constructing clear and readable process models with cancellations. To increase transparency and readability of process models with cancellations many languages for modeling business processes (such as BPMN, YAWL and others) use so called cancellation regions. A cancellation region is a subset of places in a model associated with a transition. Firing of this transition empties the region, i.e. removes all tokens happen to remain in its places.

In WF-nets cancellation regions can be naturally represented with the help of reset arcs. Now we define Petri nets with reset arcs and workflow nets with reset arcs (RWF-nets).

Definition 8 (Reset net). *A reset net is a tuple* (P, T, F, R)*, where*

- (P, T, F) *is a classical PN with places* P*, transitions* T*, and flow relation* F*,*
- $R : T \to 2^P$ *is a function mapping transitions to (possibly empty) subsets of places.*

For a transition $t \in T$, $R(t)$ is a subset of places, emptied by firing of t. When $p \in R(t)$, we say that (p, t) is a reset arc.

As in classical Petri nets a transition $t \in T$ is *enabled* in a marking m iff $\forall p \in P \; m(p) \geq F(p, t)$. An enabled transition t may *fire* yielding a new marking $m'(p) = \pi_{P \setminus R(t)}(m(p) - F(p, t)) + F(t, p)$ for each $p \in P$. Here $\pi_{P \setminus R(t)}$: $Nat^{|P|} \to Nat^{|P|}$ is a 'projection' function, which maps markings to markings by removing all tokens in reset places $R(t)$.

Definition 9 (Reset workflow net). *A reset net* N *is called a* reset workflow net (RWF-net) *iff*

1. *There is one source place* $i \in P$ *and one sink place* $f \in P$ *s. t.* ${}^{\bullet}i = f^{\bullet} = \emptyset$*;*
2. *Every node from* $P \cup T$ *is on a path from* i *to* f.

3. *The initial marking in N contains the only token in its source place.*
4. *There is no reset arc connected to the sink place, i.e., $\forall t \in T : o \notin R(t)$.*

An example of a RWF-net is shown in Fig. 7, reset arcs are denoted by double-headed arrows.

Given a log we construct a TS, states of which are formed on the basis of event sets. This can be done a standard way. After that we execute the procedure of merging the states with identical outflow. An example of a TS with states merged by outflow is presented in Fig. 4. Note that there might be situations when some dummy states and transitions should be added to a TS in order to derive a WF-net which has one source and one sink place [5].

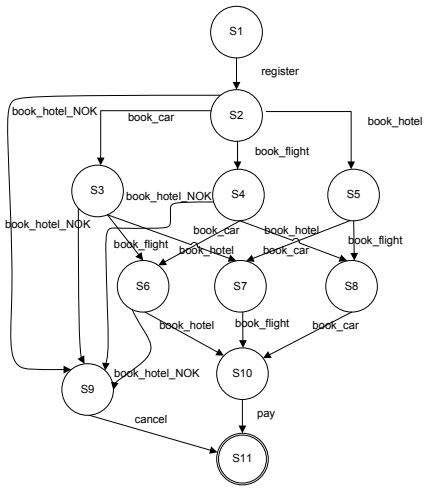

Fig. 4. Transition system for the log in Fig. 2

The state-based region algorithm [7–9] constructs a target PN in such a way that a TS is covered by its minimal regions and after that every minimal region is transformed to a place in a PN.

Our approach is based on the following assumption: *failure events* are always followed by some *catching event* in a log. In our example 'book_hotel_NOK' is such a *failure event*, and 'cancel' is a *catching event*.

Failure events inform us about errors during the process execution. A *cancellation state* is a state reached by a process after an occurrence of some *failure event*. *Catching events* inform about the work of a handler. And a *cancellation set* is a set of states which might be canceled in consequence of some error. To formalize these heuristics we now give the following definition.

Definition 10 (Cancellation state). *Let $TS = (S, E, T, s_{in})$ be a transition system. A state $s_c \in S$ is a cancellation state iff the following conditions hold:*

1. $\forall e \in E$ *s.t.* $(\overset{e}{\rightarrow} s_c)$ *we have* $(\forall s \in S : [\overset{e}{\rightarrow} s$ *implies* $(s = s_c)]$;
2. $\forall e \in E$ *s.t.* $(s_c \overset{e}{\rightarrow})$ *we have* $(\forall s \in S : [s \overset{e}{\rightarrow}$ *implies* $(s = s_c)]$;
3. $\forall e \in E$ *s.t.* $\overset{e}{\rightarrow} s_c$ *we have* $(\exists s_1, s_2 \in S : [((s_1, e, s), (s_2, e, s) \in T) \wedge (s_1 \neq s_2)])$;
4. $\exists! e : [s_c \overset{e}{\rightarrow}]$.

Definition 11 (Catching event). *Let* $TS = (S, E, T, s_{in})$ *be a transition system. An event* $e \in E$ *is a catching event iff* $(s_c \overset{e}{\rightarrow})$ *for some cancellation state* $s_c \in S$.

Definition 12 (Failure event). *Let* $TS = (S, E, T, s_{in})$ *be a transition system, a state* $s_c \in S$ *is a cancellation state,* e_f *is a failure event. A set of states* S *is a cancellation set for* e_f *(denoted by* $CS(e_f)$*) iff* $\forall s \in S$: $(s \overset{e}{\rightarrow} s_c)$.

All incoming and, correspondingly, outgoing transitions of a *cancellation state* are labeled with some separated events in TS. Events which label incoming transitions are called *failure events*. There is only one event which labels all outgoing transitions — a *catching event*. There are not less than two transitions for each *failure event*. The set of states which have outgoing transitions labeled with some *failure event* e_f is called a *cancellation set*, and is denoted by $CS(e_f)$.

In our example s_9 is a *cancellation state*, $\{s_2, s_3, s_4, s_6\}$ is a *cancellation set*, 'book_hotel_NOK' is a *failure event* and 'cancel' is a *catching event* (Fig. 4).

Note, that in our example each process terminates after the completion of the transition 'cancel', but in a general case 'cancel' may be followed by some other transitions.

Now we show that occurrence of cancellations in a log frequently leads to a complicated WF-net. This is also valid for our example (Fig. 3).

There could be such a situation when one of events occurs independently of the potential failures. See for example an event 'book_flight', booking flight procedure may start before the failure (and therefor can be interrupted) or after the successful completion of the booking hotel procedure without any possibility of being interrupted. The transitions labeled with 'book_flight' event connect states in the *cancellation set* and states which are not in the *cancellation set* at the same time. We now prove that if a TS contains a *cancellation state* as well as an event, which occurs independently of potential failures (independent parallel branches), then the generated labeled WF-net will contain transitions with identical labels.

Theorem 1. *Let* $TS = (S, E, T, s_{in})$ *be a transition system. A state* $s_c \in S$ *is a cancellation state,* $e_f \in E$ *is a failure event and* $CS(e_f)$ *is a corresponding cancellation set. Let* $e \in E$ *be an event for which the following conditions hold:*

- $e \neq e_f$;
- $\exists s^1 \in CS(e_f) : [(s^1 \overset{e}{\rightarrow})]$ – *there is a state from the cancellation set with an outgoing transition labeled by e;*
- $\exists s^2 \in CS(e_f) : [\neg(s^2 \overset{e}{\rightarrow})]$ – *the cancellation set contains a state that does not have outgoing transitions labeled by e;*

– $\exists s^3 \in S, s^3 \notin CS(e_f) : [(s^3 \xrightarrow{e})]$ – *there is a state not from the cancellation set which has outgoing transition labeled by e.*

Then the excitation closure condition for the event e is not satisfied.

The theorem conditions are illustrated by Fig. 5.

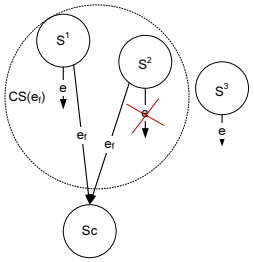

Fig. 5. Transition system for which excitation closure condition is not satisfied

Proof. We have to prove that $\bigcap_{r \in {}^o e} = GER(e)$ is not satisfied. Let us consider an arbitrary pre-region of e - r. According to the definition of a pre-region there is an exit transition labeled by e, that means that r contains all states with outgoing transitions labeled by e: $s^1, s^3 \in r$. Let e_f be a label for transitions which 'do not cross' the region r. Then $s_c \in r$, and hence $s^2 \in r$. If a transition e_f exits r, then we also have $s^2 \in r$. It means that every pre-region of e contains s^2 which does not have any outgoing transition labeled with e. In this case the *excitation closure* property $\bigcap_{r \in {}^o e} = GER(e)$ is not satisfied. □

By now we have proved that it is impossible to construct an equivalent labeled WF-net with transitions having unique labels in the presence of *cancellation state* and an event which occurs independently of the potential failures (existence of independent parallel branches), because in this case the *excitation closure* condition is not satisfied. If the *excitation closure* condition is not satisfied, then TS is not elementary and the target PN is splitted [8, 9]. It means that almost always an overcomplicated WF-net is obtained from logs with cancellations.

To overcome this problem we propose a new method of discovering a RWF-net from an event log. We start with discovering a regular structure of the process. For that we first construct a TS based on given event log. Then we delete all *cancellation states* together with their incident arcs from the TS (Fig. 6) and apply one of existing discovery algorithms to obtain a WF-net, representing the regular (without cancellations) behavior. The WF-net generated from this TS presented in Fig. 1. Then we add places, transitions and reset arcs needed for representing cancellations.

One may notice that according to the definition, the *cancellation state* forms a region itself, and hence it is transformed to a place of the target PN. So, this

place should be added to the target WF-net and connected by outgoing flows in a way it was connected in a WF-net generated from the TS with cancellation. The question remains, how to connect this place by incoming flows with other WF-net elements to achieve control flow simplicity and preserve semantics of the initial TS.

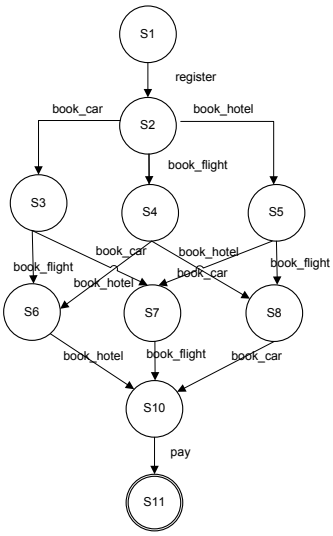

Fig. 6. Transition system without cancellations

We use an assumption that after deleting of the *cancellation state* each corresponding *cancellation set* is a minimal region. As we can see from the example above (Fig. 6) the case when *cancellation set* forms a minimal region might be rather common, especially when a process contains an exit transition labeled by a 'normal flow' event. Herein our example 'booking_hotel' is such a 'normal flow' event, which specifies the case when the booking hotel procedure has been terminated without failures.

Let us formalize the approach and prove its correctness under the assumption that after the deletion of the *cancellation state* each corresponding *cancellation set* is a minimal region.

Algorithm 1. *(Constructing a RWF-net from the TS with cancellations).*
Let $TS = (S, E, T, s_{in})$ be a transition system. Let $s_c \in S$ be a *cancellation state*, $e_{f_1}, ..., e_{f_n} \in E$ — *failure events*, $e_c \in E$ — a *catching event* and $CS(e_{f_1}), ..., CS(e_{f_n})$ — the corresponding *cancellation sets*.

1. Construct a TS $TS' = (S', E', T', s_{in})$ from TS by deleting the *cancellation state* and its incident arcs.

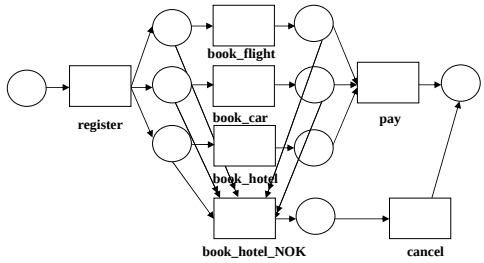

Fig. 7. Reset WF-net for booking process

2. Verify that $CS(e_{f_1}), ..., CS(e_{f_n})$ are minimal regions in TS', otherwise return a message that RWF-net cannot be constructed.
3. Construct WF' as a WF-net derived from TS' according to the state-based region algorithm. TS' is covered by its minimal regions and after that every minimal region is transformed to a place in WF' [7–9].
4. Perform the transformation of WF' by adding transitions corresponding to the *failure events* $e_{f_1}, ..., e_{f_n}$ and transition corresponding to the *catching event* $e_c \in E$ (Fig. 8).
5. Add outgoing control flow to the transition labeled by e_c, as if the state-based region algorithm was applied to the initial transition system TS.
6. For each *failure event* e_{f_i} a place corresponding to the minimal region $CS(e_{f_i})$ is connected by an outgoing arc with the transition denoting this *failure event*; all such transitions are connected by outgoing arcs with an additional place, which in turn is connected with the transition labeled by the *catching event* (Fig. 8). If there is only one *failure event* (see Fig. 7), it is connected directly with the transition labeled with the *catching event*.
7. All other places corresponding to the minimal regions, which contain states from $CS(e_{f_i})$, should be connected with the transition labeled by the *failure event* e_{f_i} by reset arcs.

The RWF-net which is manually constructed from the log (Fig. 2) according to Algorithm 1 is presented in Fig. 7. This RFW-net is structurally similar to the initial regular WF-net (Fig. 1) and the core process structure could be easily retrieved from the RWF-net. Let us prove the correctness of Algorithm 1.

Theorem 2. *Let $TS = (S, E, T, s_{in})$ be a transition system. Construct a RWF-net WF using an Algorithm 1. Then the labeled RG of WF is isomorphic (or split-isomorphic) to the minimized TS.*

Proof. The labeled RG of the intermediate WF-net WF' is safe and isomorphic (or split-isomorphic) to the minimized initial TS' according to the principles of the state-based region algorithm [6]. TS differs from TS' only in addition of the *cancellation state* $s_c \in S$ and its incident arcs (transitions). Let us consider an

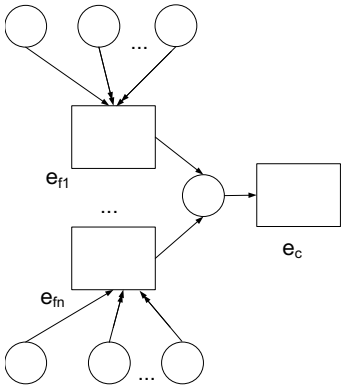

Fig. 8. Construction of the target RWF-net

arbitrary additional transition e_{f_i} which connects a state from the *cancellation set* $s \in S$ with the *cancellation state* $s_c \in S$. The presence of such a transition labeled by *failure event* e_{f_i} means that the target net can change its state having tokens in a place which corresponds to the *cancellation set* and other places which correspond to the minimal regions containing state s (these places have appropriate reset arcs in the target reset WF-net). Note that every minimal region in a TS corresponds to the place in the target WF-net [7–9]. And vice versa addition of new transitions, arcs and reset-arcs to the WF-net WF' will add necessary transitions to the TS. Also the outgoing flow of the transition labeled by *catching event* will be added according to the state-based region algorithm, taking into account that the *cancellation state* $s_c \in S$ forms a minimal region itself. All these arguments lead us to the conclusion that labeled RG of the target reset WF-net WF is isomorphic (or split-isomorphic) to the minimized initial TS.□

5 Conclusion

Construction of readable process models is an important requirement for process discovery techniques. Since cancellations occur in the majority of real-life event logs, it is necessary to construct an appropriate algorithm to deal with cancellations and synthesize simple and clear process models. In this work we have proved that occurrence of cancellations in a log frequently leads to process models with the overcomplicated control flow. We also described an algorithm, which discovers readable RWF-net models with clear regular structure from event logs. We have proved the correctness of this algorithm. In the future, we plan to implement this approach and apply it to real-life event logs.

References

1. W.M.P. van der Aalst *Discovery, Conformance and Enhancement of Business Processes.* Springer-Verlag, Berlin, 2011.
2. Object Management Group *Business Process Modeling Notation (BPMN) Version 2.0, OMG Final Adopted Specification.* Object Management Group, 2011.
3. W.M.P. van der Aalst and A.H.M. ter Hofstede YAWL: Yet another workflow language. *Information Systems.* Vol. 30, Nr. 4, pages 245-275. 2005.
4. W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, MVoorhoeve, M.T. Wynn Soundness of Workflow Nets with Reset Arcs. *Transactions on Petri Nets and Other Models of Concurrency.* Vol. 3, pages 50-70. 2005.
5. W.M.P. van der Aalst Discovery, Verification and Conformance of Workflows with Cancellation. In 4th International Conference, ICGT 2008, Vol. 5214 of *Lecture Notes in Computer Science*, pages 18-37. Springer, 2008.
6. W.M.P. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, and C.W. G?unther. Process Mining: A Two-Step Approach using Transition Systems and Regions. BPM Center Report BPM-06-30, BPMcenter.org, 2006.
7. M. Nielsen, G. Rozenberg, and P.S. Thiagarajan. Elementary transition systems. *Theoretical computer science.* Vol. 96, pages 3-33. 1992.
8. J. Cortadella,M.Kishinevsky,L. Lavagno, and A.Yakovlev. Synthesizing Petri nets from state-based models. Technical Report RR 95/09 UPC/DAC, Universitat Politecnica de Catalunya, April 1995.
9. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers.* Vol. 47, Nr. 8, pages 859-882. 1998.
10. B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, Application and Theory of Petri Nets, Vol. 3536 of *Lecture Notes in Computer Science*, pages 444-454. Springer-Verlag, Berlin, 2005.