Information Systems 38 (2013) 585–605

Contents lists available at SciVerse ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Simplifying discovered process models in a controlled manner

Dirk Fahland *, Wil M.P. van der Aalst

Eindhoven University of Technology, The Netherlands

ARTICLE INFO

Available online 25 July 2012

Keywords:
Process mining
Model simplification
Petri nets
Branching processes

ABSTRACT

Process models discovered from a process log using process mining tend to be complex and have problems balancing between overfitting and underfitting. An overfitting model allows for too little behavior as it just permits the traces in the log and no other trace. An underfitting model allows for too much behavior as it permits traces that are significantly different from the behavior seen in the log. This paper presents a post-processing approach to simplify discovered process models while controlling the balance between overfitting and underfitting. The discovered process model, expressed in terms of a Petri net, is unfolded into a branching process using the event log. Subsequently, the resulting branching process is folded into a simpler process model capturing the desired behavior.

1. Introduction

Information systems are becoming more and more intertwined with the operational processes they support. While supporting these processes, multitudes of events are recorded, cf. audit trails, database tables, transaction logs, data warehouses. The goal of *process mining* is to use such event data to extract process-related information. The most prominent problem of process mining is *process discovery*, that is, to automatically *discover* a process model by observing events recorded by some information system. The discovery of process models from event logs is a relevant, but also challenging, problem [1–3].

Input for process discovery is a collection of traces. Each trace describes the life-cycle of a process instance (often referred to as case). Output is a process model that is able to reproduce these traces. The automated discovery of process models based on event logs helps to jump-start process improvement efforts and provides an objective up-to-date process description. There are two other kinds of process mining. *Process extension* extends a given (handmade or discovered) process model with information from the log, for instance, by projecting a log on a discovered model to show bottlenecks and deviations. *Conformance checking* is the problem of measuring how well a handmade or discovered process model describes behavior in a given log [1].

The main problem of process discovery from event logs is to balance between *overfitting* and *underfitting*. A model is overfitting if it is too specific, i.e., the example behavior in the log is included, but new instances of the same process are likely to be excluded by the model. For instance, a process with 10 concurrent activities has $10! = 3{,}628{,}800$ potential interleavings. However, event logs typically contain fewer cases. Moreover, even if there are 3,628,800 cases in the log, it is extremely unlikely that all possible variations are present. Hence, an overfitting model (describing exactly these cases) will not capture the underlying process well. A model is underfitting when it over-generalizes the example behavior in the log, i.e., the model allows for behaviors very different from what was seen in the log. Process discovery is challenging because (1) the log typically only contains a *fraction* of all possible behaviors, (2) due to concurrency, loops, and choices the *search space has a complex structure*, and (3) there are *no*

* Corresponding author. Tel.: +31 40 2474804.
E-mail addresses: d.fahland@tue.nl (D. Fahland), w.m.p.v.d.aalst@tue.nl (W.M.P. van der Aalst).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.07.004

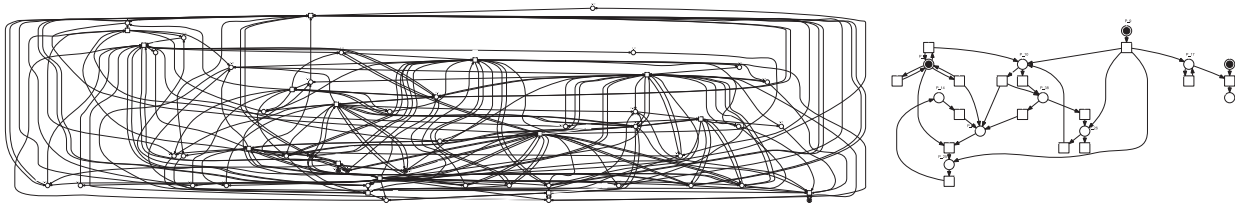

Fig. 1. Hospital patient treatment process after process discovery (left) and after subsequent simplification using the approach presented in this paper (right).

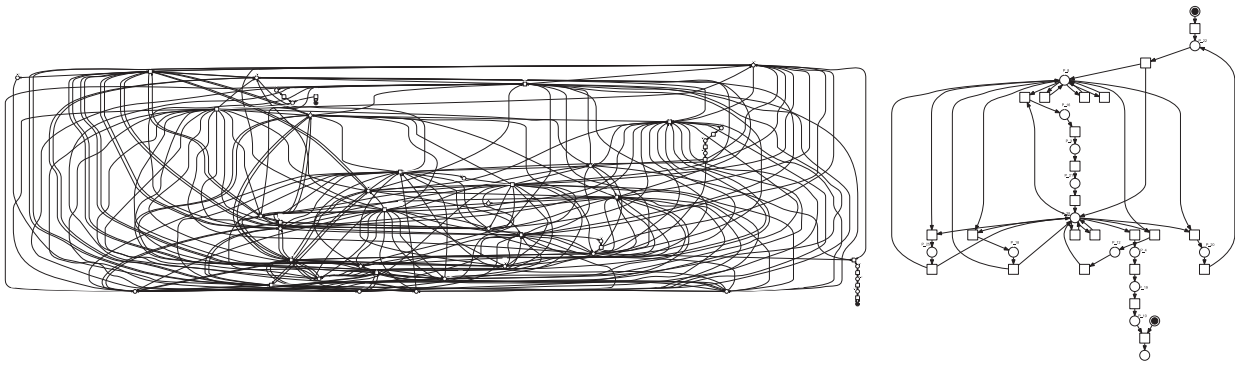

Fig. 2. Municipality complaint process after process discovery (left) and after subsequent simplification (right).

negative examples (i.e., a log shows what has happened but does not show what could not happen) [1,3].

A variety of approaches has been proposed to address these challenges [1,2]. Technically, all these approaches extract ordering constraints on activities which are then expressed as control-flow constructs in the resulting process model. Provided that enough event data are available and variability is low, today's approaches are able to discover the underlying process adequately. However, processes with more variability are more difficult to discover and numerous approaches have been proposed to deal with this.

Several approaches try to *abstract from infrequent behavior* and construct models that capture only the "highways" in processes. Examples are heuristic mining[1] [4], fuzzy mining [5], and genetic process mining [6]. The resulting models are relatively simple, but may not be able to reproduce all traces seen in the log. These techniques exploit the fact that for many processes the so-called "80/20-rule" holds, i.e., 80% of the observed cases can be explained by 20% of the paths in the process whereas the remaining 20% of cases is responsible for 80% of the variability. Although the techniques proposed in [4–6] can simplify models, parts of the event log are no longer explained by the model and the model is often not executable because split-join behavior is either unspecified (fuzzy mining) or implicit (heuristic mining and genetic process mining) [7].

[1] Historically, process discovery and process mining were used synonymously, as discovery was the first and most prominent process mining problem that was addressed. Thus, various discovery techniques are called "mining techniques" although they just focus on discovery. We will use their original name, but use the term "discovery" to refer to the problem.

Other approaches try to deal with variability by *constructing an over-general model*. Instead of leaving out infrequent behavior, everything is allowed unless there is strong evidence that it is not possible. This can easily be understood in terms of a Petri net. A Petri net with transitions T and without any places can reproduce any event log over a set of activities T. Adding a place to a Petri net corresponds to adding a constraint on the behavior. Techniques based on *language-based regions* [8,9] use this property. For example, as shown in [9] it is possible to solve a system of inequations to add places that do not inhibit behavior present in the event log. In [10], an approach based on convex polyhedra is proposed. Here the Parikh vector of each prefix in the log is seen as a polyhedron. By taking the convex hull of these convex polyhedra one obtains an over-approximation of the possible behavior. In [11], the authors resort to the use of OR-splits and OR-joins to create an over-general model that guarantees that all traces in the log can be reproduced by the model. Surprisingly, these over-general process models tend to be convoluted as illustrated by Figs. 1 and 2.

In [12] an approach to balance overfitting and underfitting is proposed. First, a transition system is constructed from the log; the user may balance generalization by influencing how states are generated from the log. Then, a Petri net is derived from this transition system. The approach requires expert knowledge to specify the right abstraction that balances overfitting and underfitting. If applied correctly, this technique yields simpler models (compare Fig. 3 (left) and Fig. 1 (left)), but even these models are still convoluted and can be simplified as shown by Fig. 3 (right).

The problem that we address in this paper is to *structurally simplify* a mined process model N while *preserving that*