Information Systems 56 (2016) 258–283

Contents lists available at ScienceDirect

Information Systems

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

Efficient discovery of Target-Branched Declare constraints

Claudio Di Ciccio [a,*], Fabrizio Maria Maggi [b], Jan Mendling [a]

[a] Vienna University of Economics and Business, Institute for Information Business (Building D2, Entrance C), Welthandelsplatz 1, A-1020 Vienna, Austria
[b] University of Tartu, Estonia

ARTICLE INFO

Available online 2 July 2015

Keywords:
Process mining
Knowledge discovery
Declarative process

ABSTRACT

Process discovery is the task of generating process models from event logs. Mining processes that operate in an environment of high variability is an ongoing research challenge because various algorithms tend to produce spaghetti-like process models. This is particularly the case when procedural models are generated. A promising direction to tackle this challenge is the usage of declarative process modelling languages like Declare, which summarise complex behaviour in a compact set of behavioural constraints on activities. A Declare constraint is branched when one of its parameters is the disjunction of two or more activities. For example, branched Declare can be used to express rules like "in a bank, a mortgage application is always eventually followed by a notification to the applicant by phone *or* by a notification by e-mail". However, branched Declare constraints are expensive to be discovered. In addition, it is often the case that hundreds of branched Declare constraints are valid for the same log, thus making, again, the discovery results unreadable. In this paper, we address these problems from a theoretical angle. More specifically, we define the class of Target-Branched Declare constraints and investigate the formal properties it exhibits. Furthermore, we present a technique for the efficient discovery of compact Target-Branched Declare models. We discuss the merits of our work through an evaluation based on a prototypical implementation using both artificial and real-life event logs.

1. Introduction

Process discovery is the important initial step of business process management that aims at arriving at an as-is model of an investigated process [1]. Due to this step being difficult and time-consuming, various techniques have been proposed to automatically discover a process model from event logs. These log data are often generated by information systems that support parts or the entirety of a process. The result is typically presented as a Petri net or a similar kind of flow chart and the automatic discovery is referred to as process mining.

While process mining has proven to be a powerful technique for structured and standardised processes, there is an ongoing debate on how processes with a high degree of variability can be effectively mined. One approach to this problem is to generate a declarative process model, which rather shows the constraints of behaviour instead of the available execution

* Corresponding author. Tel.: +43 1 31336 5222.
E-mail addresses: claudio.di.ciccio@wu.ac.at (C. Di Ciccio), f.m.maggi@ut.ee (F.M. Maggi), jan.mendling@wu.ac.at (J. Mendling).

http://dx.doi.org/10.1016/j.is.2015.06.009
0306-4379/

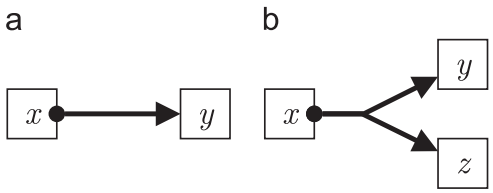

Fig. 1. Declare (a) and Target-Branched Declare (b) *Response* templates.

sequences. The resulting models are represented in languages like Declare. In many cases, they provide a way to represent complex, unstructured behaviour in a compact way, which would look overly complex in a spaghetti-like Petri net.

Declare is a process modelling language first introduced in [2]. The language defines a set of classes of constraints, the Declare templates, that are considered the most interesting ones for describing business processes. Templates are parameterised and constraints are instantiations of templates on real activities. For example, the *Response* constraint, stating that "activity *pay* is always eventually followed by activity *send invoice*" is an instantiation of the Declare template *Response* specifying that "an activity x is always eventually followed by an activity y". Templates have a graphical representation and formal semantics based on Linear Temporal Logic on Finite Traces (LTL_f). This allows Declare models to be verifiable and executable. Fig. 1a shows the graphical representation of the *Response* template. Its LTL_f semantics is $\Box(x \rightarrow \Diamond y)$. Constraints inherit the graphical representation and the LTL_f semantics from the corresponding templates.

The current techniques for the discovery of Declare models [3–7] are limited to the discovery of constraints based on the standard set of Declare templates. This means that the discovered constraints will involve one activity for each parameter specified in the corresponding templates. However, as described in [2], a constraint can define more than one activity for each parameter. For example, a *Response* constraint can be used to express rules like "in a bank, a mortgage application is always eventually followed by a notification to the applicant by phone *or* by a notification by e-mail". In this rule, the "mortgage application" plays the role of the *activation*. "Notification by phone" and "notification by e-mail" constitute the so-called *targets* of the constraint. In this case, we say that the target parameter branches and, in the graphical representation, this is displayed by multiple arcs connecting the activation to the branched targets. In LTL_f semantics, a branched parameter is replaced by a disjunction of parameters. Fig. 1b shows the graphical representation of the *Response* template branching on the target. Its LTL_f semantics is $\Box(x \rightarrow \Diamond(y \vee z))$.

Target-Branched Declare (TBDeclare) extends Declare by encompassing constraints that branch on target parameters, thus providing the process modellers with the possibility of defining a much wider set of constraints. In this paper, we address the problem of mining TBDeclare constraints efficiently. At the same time, the technique we propose aims at limiting the sheer amount of returned constraints to the set of the most meaningful ones. To this extent, we rely on formal properties of TBDeclare, i.e., (i) set-dominance and (ii) subsumption hierarchy. Set-dominance is based on the observation that, for example, stating that "a is always eventually followed by b or c" entails that "a is always eventually followed by b, c or d", i.e., since the set of targets for the first constraint is included in the set of targets for the second constraint, the first constraint is stronger than the second one. In this case, if both constraints hold in the provided event log, only the stronger one will be discovered. In addition, Declare constraints are not independent, but form a subsumption hierarchy. Therefore, a constraint (e.g., a is eventually followed by b or c) is redundant if a stronger constraint holds (e.g., a is directly followed by b or c). Also in this case, it is possible to keep the stronger constraint and discard the weaker one in the discovered model. The key idea of our proposed approach is to exploit set-dominance and subsumption hierarchy relationships, in combination with the use of interestingness metrics like constraint support and confidence [5], to drastically prune the set of discovered constraints. We present formal proofs to demonstrate the merits of this approach and a prototypical implementation for emphasising its feasibility and efficiency.

In this paper, we extend the work presented in [8] in four directions: (i) theoretical discussion, (ii) algorithm presentation, (iii) implementation improvement, and (iv) evaluation. From a foundational perspective, this paper formally elaborates on how the monotonicity of LTL_f temporal operators can be exploited to prove set-dominance for TBDeclare. The algorithm is presented in thorough detail here: it describes all the procedures undertaken to mine the constraints, along with trailing examples. The implementation of the algorithm is also improved now, as an entirely new technique for the computation of *AlternateResponse* and *AlternatePrecedence* constraints has been devised. In this way, a major limitation of the process discovery algorithm presented in [8] is resolved. Furthermore, this has enabled us to cover a broader range of experiments including the application to an additional benchmark based on the use of the log provided for the BPI challenge 2014 [9].

Against this background, this paper is structured as follows. Section 2 introduces the essential concepts of LTL_f and Declare as a background of our work. Section 3 provides the formal foundations for mining Target-Branched constraints. Section 4 defines the construction of a knowledge base from which the final constraint set is built. Section 5 describes the performance evaluation. Section 6 investigates our contribution in the light of related work. Section 7 concludes the paper with an outlook on future research.

2. Background

Process mining is the set of techniques that aims at understanding the behaviour of a process, given as input a set of data reporting the executions of such a process, i.e., an event log L. An event log consists of a collection of traces $\vec{\tau}_i$, with $i \in [1, |L|]$