Expert Systems With Applications 81 (2017) 398–409

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Multi-level abstraction for trace comparison and process discovery

Stefania Montani[a,*], Giorgio Leonardi[a], Manuel Striani[b], Silvana Quaglini[c], Anna Cavallini[d]

[a] DISIT, Computer Science Institute, Università del Piemonte Orientale, Viale Michel 11, Alessandria, Italy
[b] Department of Computer Science, Università di Torino, Corso Svizzera 105, Torino, Italy
[c] Department of Electrical, Computer and Biomedical Engineering, Università di Pavia, Via Ferrata 1, Pavia, Italy
[d] I.R.C.C.S. Fondazione "C. Mondino", Via Mondino 2, Pavia, Italy - on behalf of the Stroke Unit Network (SUN) collaborating centers, Italy

ARTICLE INFO

Article history:
Received 24 November 2016
Revised 28 March 2017
Accepted 29 March 2017
Available online 31 March 2017

Keywords:
Abstraction
Trace comparison
Process discovery
Stroke management

ABSTRACT

Many information systems record executed process instances in the *event log*, a very rich source of information for several process management tasks, like process mining and trace comparison. In this paper, we present a framework, able to convert activities in the event log into higher level concepts, at different levels of abstraction, on the basis of domain knowledge. Our abstraction mechanism manages non trivial situations, such as interleaved activities or delays between two activities that abstract to the same concept.

Abstracted traces can then be provided as an input to an intelligent system, meant to implement a variety of process management tasks, significantly enhancing the quality and the usefulness of its output.

In particular, in the paper we demonstrate how trace abstraction can impact on the quality of process discovery, showing that it is possible to obtain more readable and understandable process models.

We also prove, through our experimental results, the impact of our approach on the capability of trace comparison and clustering (realized by means of a metric able to take into account abstraction phase penalties) to highlight (in)correct behaviors, abstracting from details.

1. Introduction

Many commercial information systems and enterprise resource planning tools, routinely adopted by organizations worldwide, record information about the executed business process instances in the form of an *event log*[1]. The event log stores the sequences (*traces* van der Aalst, 2011 henceforth) of activities that have been executed at the organization, typically together with some key parameters, such as execution times.

Event logs constitute a very rich source of information for several business process management tasks. Indeed, the experiential knowledge embedded in traces is directly resorted to, e.g., in *operational support* (van der Aalst, 2011) and in *agile workflow* tools (Weber & Wild, 2005), which can take advantage of *trace comparison* and retrieval, to make predictions about a running process instance completion, or to provide instance adaptation support, in response to expected situations as well as unanticipated exceptions in the operating environment. Moreover, event logs are the input to *process mining* (van der Aalst et al., 2003; van der Aalst, Weijters, & Maruster, 2004) algorithms, a family of mainly a-posteriori analysis techniques able to extract non-trivial knowledge from these historic data; within process mining, *process model discovery* algorithms, in particular, take as input the event log and build a process model, focusing on its control flow constructs.

All of these activities, however, provide a purely syntactical analysis, where activities in the event log are compared and processed only referring to their names. Activity names are strings without any semantics, so that identical activities, labeled by synonyms, will be considered as different, or activities that are special cases of other activities will be processed as unrelated.

Upgrading trace comparison and process mining to the conceptual layer can enhance existing algorithms towards more advanced and reliable approaches. Indeed, the capability of relating semantic structures such as taxonomies or ontologies to activities in the log can enable both trace comparison and process discovery techniques to work at different levels of abstraction (i.e., at the level of instances and/or concepts) and, therefore, to mask irrelevant details, to promote reuse, and, in general, to make trace and/or process model analysis much more flexible, and closer to the real user

* Corresponding author.
E-mail addresses: stefania.montani@uniupo.it (S. Montani), giorgio.leonardi@uniupo.it (G. Leonardi), striani@di.unito.it (M. Striani), silvana.quaglini@unipv.it (S. Quaglini), anna.cavallini@mondino.it (A. Cavallini).

[1] IEEE Taskforce on Process Mining: Process Mining Manifesto, http://www.win.tue.nl/ieeetfpm (last accessed on 4/11/2016).

http://dx.doi.org/10.1016/j.eswa.2017.03.063
0957-4174/

needs. As a matter of fact, *semantic process mining*, defined as the integration of semantic processing capabilities into classical process mining techniques, has been proposed in the literature since the first decade of this century (see, e.g., de Medeiros and van der Aalst, 2009; de Medeiros, van der Aalst, and Pedrinaci, 2008, and Section 5). However, while more work has been done in the field of semantic conformance checking (Grando, Schonenberg, & van der Aalst, 2011; de Medeiros et al., 2008), to the best of our knowledge semantic process discovery needs to be further investigated.

In this paper:

1. We present a semantic-based, multi-level abstraction mechanism, able to operate on event log traces. In our approach, activities in the log are mapped to instances of ground concepts (leaves) in a taxonomy, so that they can be converted into higher-level concepts by navigating the hierarchy, up to the desired level, on the basis of the user needs;
2. We provide the abstraction mechanism as an input to further analysis mechanisms, namely trace comparison and process discovery.

The abstraction mechanism has been designed to properly tackle non-trivial issues that could emerge. Specifically:

- Two activities having the same ancestor in the taxonomy (at the chosen abstraction level) may be separated in the trace by a *delay* (i.e., a time interval where no activity takes place), or by activities that descend from a different ancestor (*interleaved activities* henceforth). Our approach allows to deal with these situations, by creating a single *macro-activity*, i.e., an abstract activity that covers the whole time span of the two activities at hand, and is labeled as the common ancestor; the macro-activity is however built only if the total delay length, or the total number/length of interleaved activities, do not overcome proper admissibility thresholds set by the user. The delays and interleaved activities are quantified and recorded, for possible use in further analyses. In particular, we present a metric where this information is accounted for as a penalty, and affects the distance value in abstract trace comparison;
- Abstraction may generate different types of temporal constraints between pairs of macro-activities; specifically, given the possible presence of interleaved activities, we can obtain an abstracted trace with two (or more) overlapping or concurrent macro-activities. Our approach allows to represent (and exploit) this information, by properly maintaining both quantitative and qualitative temporal constraints in abstracted traces. Once again, this temporal information can be exploited in further analyses. In particular, the metric we adopt in trace comparison can deal with all types of temporal constraints.

The most significant and original *methodological contributions* of our work thus consist in:

- Having defined a proper *mechanism for abstracting event log traces*, able to manage non trivial situations (originating from the treatment of interleaved activities or delays between two activities sharing the same ancestor);
- Having provided *a trace comparison facility*, which resorts to a *metric* (extending the one we presented in Montani & Leonardi, 2014), able to take into account also the information recorded during the abstraction phase.

On the other hand, as for process discovery, we currently rely on classical algorithms embedded in the open source framework ProM (van Dongen, De Medeiros, Verbeek, Weijters, & der Aalst, 2005). It is worth noting that the abstraction mechanism could, in principle, be given as an input to different analysis techniques as well, besides the ones described in this paper.

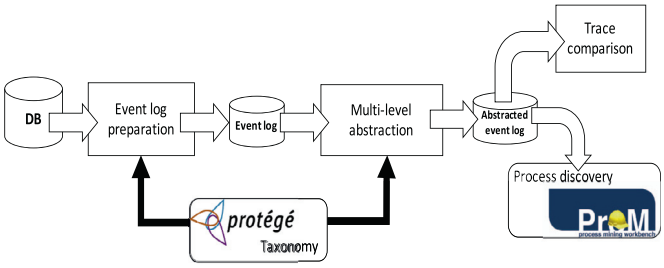

Fig. 1. Framework architecture and data flow.

In addition to these methodological contributions, in the paper we also describe our experimental work in the field of stroke care, in which we adopted multi-level abstraction and trace comparison to cluster event logs of different stroke units, in order to highlight correct and incorrect behaviors, abstracting from details, such as local resource constraints or local protocols, that are irrelevant to verify the medical appropriateness of a macro-activity. We also provide process discovery results, showing how the abstraction mechanism allows to obtain simpler and more understandable stroke process models.

The paper is organized as follows. Section 2 presents methodological and technical details of the framework. Section 3 describes an application of the abstraction facility as a pre-processing step for process model discovery. Section 4 provides experimental results. Section 5 addresses comparisons with related work. Section 6 is devoted to discussion. Finally, Section 7 presents our conclusions and future research directions.

2. Methods

This section describes methodological and technical details of our approach.

The architecture and the data flow of the framework we have developed are shown in Fig. 1.

The first step to be executed is *event log preparation*, that takes as input the available database (recording executed activities and additional data), and exploits domain knowledge (in the form of a taxonomy); the event log then undergoes *multi-level abstraction*, which resorts to domain knowledge as well. The abstracted event log can be given as an input to different process management tasks, such as *trace comparison* and *process discovery* (currently realized by resorting to ProM van Dongen et al., 2005).

The terminology we use, and the details about domain knowledge sources and computational modules, are described in the following subsections.

2.1. Terminology

Trace: a sequences of activities that belong to a same process execution.

Activity or **ground activity**: an activity recorded in an event log trace (corresponding to a leaf concept in the taxonomy described in Section 2.2).

Delay: time interval between two ground activities logged in a trace, where no other activity takes place.

Interleaved activity: a ground activity that descends from a different ancestor in the taxonomy of activities described in Section 2.2, with respect to the two ground activities that are currently being considered for abstraction. In the trace, the interleaved activity is placed between the two activities at hand.

Macro-activity: partial output of the abstraction process; a macro-activity is an abstracted activity that covers the whole time span of multiple ground activities, and is labeled as their common