A framework for multi-level semantic trace abstraction

Doctoral Consortium ICCBR 2017

Manuel Striani

PhD Candidate
Department of Computer Science, University of Torino
Corso Svizzera 185, 10149 Torino, Italy
striani@di.unito.it

1 Research Summary

Many commercial information systems and enterprise resource planning tools routinely adopted by organizations and companies worldwide, like those provided by, e.g., Oracle and SAP, record information about the executed business process instances in the form of an *event log* [5]. The event log stores the sequences (*traces* henceforth [3]) of actions that have been executed at the organization, typically together with key execution parameters, such as times, costs and resources.

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* and in *agile workflow* tools, which can take advantage of **trace comparison** and **retrieval**. Operational support [3] assists users while process instances are being executed, by making predictions about the instance completion, or recommending suitable actions, resources or routing decisions, on the basis of the comparison to already completed instances retrieved from the log. The agile workflow technology [10, 8] deals with adaptation and overriding needs in response to expected situations (e.g., new laws, reengineering efforts) as well as to unanticipated exceptions and problems in the operating environment (e.g., emergencies) [4], even if the default process schema is already in use by some running instances [9, 2]: in order to provide an effective and quick adaptation support, many agile workflow systems share the idea of recalling and reusing concrete examples of changes adopted in the past, recorded as traces in the event log. The CBR [1] methodology, and in particular the retrieval step, can therefore be adopted in this context.

In my PhD thesis, I am developing a framework to compare and retrieve process traces, represented at **different levels of abstraction**. The framework will then be interfaced to operational support or agile workflow tools, as well as to other analysis mechanisms. In this paper, I describe the methodological approach behind trace abstraction; the applications mentioned above will be considered in my future work.

1.1 Multi-level abstraction mechanism

We are developing a **semantic-based, multi-level abstraction mechanism**, able to operate on event log traces. In our approach, actions 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.

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

- two actions 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 action takes place), or by actions that descend from a different ancestor (*interleaved actions* henceforth). Our approach allows to deal with these situations, by creating a single *macro-action*, i.e., an abstract action that covers the whole time span of the two actions at hand, and is labeled as the common ancestor; the macro-action is however built only if the total delay length, or the total number/length of interleaved actions, do not overcome proper admissibility thresholds set by the user. The delays and interleaved actions are quantified and recorded, for possible use in further analyses. In particular, we have defined a **similarity metric** where this information is accounted for as a penalty, and affects the similarity value in abstract trace comparison;
- abstraction may generate different types of temporal constraints between pairs of macro-actions; specifically, given the possible presence of interleaved actions, we can obtain an abstracted trace with two (or more) overlapping or concurrent macro-actions. 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 **similarity metric** we adopt in trace comparison can deal with all types of temporal constraints.

Specifically, the procedure to abstract a trace operates as follows:

- for every action i in the trace:
  - i is abstracted as its ancestor at the taxonomy level selected by the user; the macro-action m_i, labeled as the identified ancestor, is created;
  - for every element j following i in the trace:
    - if j is a delay, its length is added to a variable $tot-delay$, that stores the total delay duration accumulated so far during the creation of m_i;
    - if j is an interleaved action, its length is added to a variable $tot-inter$, that stores the total interleaved actions durations accumulated so far during the creation of m_i;
    - if j is an action that, according to domain knowledge, abstracts as the same ancestor as i, m_i is extended to include j, provided that

$tot - delay$ and $tot - inter$ do not exceed domain-defined thresholds. j is then removed from the actions in the trace that could start a new macro-action, since it has already been incorporated into an existing one;

- the macro-action m_i is appended to the output abstracted trace which, in the end, will contain the list of all the macro-actions that have been created by the procedure.

The variables $tot - delay$ and $tot - inter$, accumulated during abstraction, are also provided as an output attribute of each macro-action and they will be used as a penalty in abstracted trace similarity calculation.

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

1. having defined a proper ***mechanism for abstracting event log traces***, able to manage non trivial situations (originating from the treatment of interleaving actions or delays between two actions sharing the same ancestor);
2. having provided ***a trace comparison facility***, which resorts to a ***similarity metric*** (extending the metric presented in [6]), able to take into account also the information recorded during the abstraction phase.

In the third year, I will concentrate on experimental work referring to trace comparison and I will deal with operational support, agile workflow management, or other activities, including process mining on abstracted traces. As regards process mining, in particular, we wish to test when abstraction allows to make clear and more readable process model.

1.2 Current development stage

With the help of an expert physician in stroke patient management, we have formalized medical domain knowledge in a taxonomy (which has been organized by goals) by using the Protègè ontology editor [7]. Actions in traces are mapped to the taxonomy leaves, so navigating the taxonomy it is possibile to abstract actions by goals. we have worked on a metric for trace comparison that is able to manage both temporal and non temporal information in traces, and to take into account information collected during the abstraction process.

The system architecture we have developed, is shown in Figure 1. Rectangles represent computational modules, while ovals and cylinders represent domain knowledge sources and the database. The first step to be executed is *event log preparation*, that takes in input the available database (DB), and exploits domain knowledge (the taxonomy); the event log will then undergo *abstraction*. The abstracted event log will be given as an input to *trace comparison* resorting to the metric we have developed, or to *process mining*, *operational support*, or other activities, that we plan to realize by resorting to ProM.

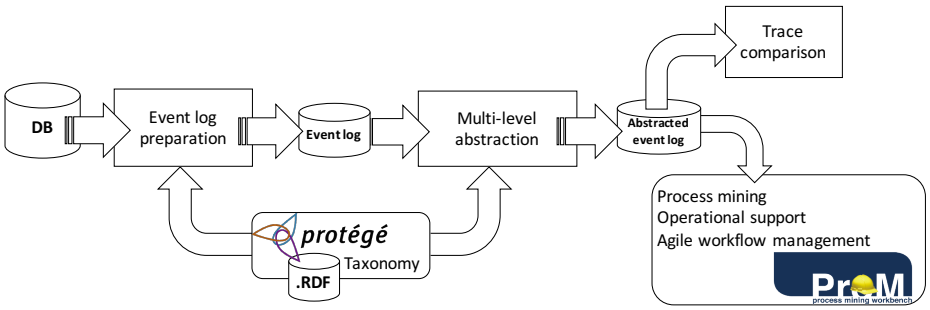

Fig. 1. Framework architecture and data flow

1.3 Future work

During my last PhD year, the framework will be tested in the field of stroke management, where we will adopt 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). The goal will be to show that, the application of the abstraction mechanism allows to obtain more homogeneous and compact clusters (i.e., able to aggregate closer examples), still making outliers clearly identifiable, and isolated in the cluster hierarchy. Some first encouraging results are already available.

As regards process mining, the ground processes (process learned on trace at the same level of taxonomy leaves) are typically "spaghetti-like": they presents an extremely large number of nodes and edges which make it hard to identify details. Our hypothesis is that models learned on abstracted traces will be much more compact and it will be possible for medical experts to analyze them. This topic will be studied during my last year as well.

Finally, we will provide abstracted traces as an input to operational support or agile workflow management facilities.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations and systems approaches. *AI Communications*, 7:39–59, 1994.
2. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolutions. *Data and Knowledge Engineering*, 24:211–238, 1998.
3. W. Van der Aalst. *Process Mining. Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
4. P. Heimann, G. Joeris, C. Krapp, and B. Westfechtel. Dynamite: dynamic task nets for software process management. In *Proceedings International Conference of Software Engineering*, pages 331–341, Berlin, 1996.
5. *http : //www.win.tue.nl/ieeetfpm*. IEEE Taskforce on Process Mining: Process Mining Manifesto (last accessed on 4/11/2013).
6. S. Montani and G. Leonardi. Retrieval and clustering for supporting business process adjustment and analysis. *Information Systems*, 40:128–141, 2014.
7. The Protégé Team. The Protégé website: urlhttp://protege.stanford.edu/, 2013.
8. M. Reichert and P. Dadam. Adeptflex-supporting dynamic changes of workflows without losing control. *J. Intell. Inf. Syst.*, 10:93–129, 1998.
9. S. Rinderle, M. Reichtert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems - a survey. *Data and Knowledge Engineering*, 50:9–34, 2004.
10. B. Weber and W. Wild. Towards the agile management of business processes. In K. D. Althoff, A. Dengel, R. Bergmann, M. Nick, and T. Roth-Berghofer, editors, *Professional knowledge management WM 2005, LNCS 3782*, pages 409–419, Washington DC, 2005. Springer, Berlin.