Towards semantic process mining through knowledge-based trace abstraction

S. Montani (1), M. Striani (2), S. Quaglini (3), A. Cavallini (4), G. Leonardi (1)

(1) DISIT, Computer Science Institute, Università del Piemonte Orientale, Alessandria, Italy
(2) Department of Computer Science, Università di Torino, Italy
(3) Department of Electrical, Computer and Biomedical Engineering, Università di Pavia, Italy
(4) I.R.C.C.S. Fondazione "C. Mondino", Pavia, Italy - on behalf of the Stroke Unit Network (SUN) collaborating centers

Abstract. Many information systems nowadays record data about the process instances executed at the organization in the form of *traces* in an event log. In this paper we present a framework able to convert actions found in the traces into higher level concepts, on the basis of domain knowledge. Abstracted traces are then provided as an input to semantic process mining.
The approach has been tested in the medical domain of stroke care, where we show how the abstraction mechanism allows the user to mine process models that are easier to interpret, since unnecessary details are hidden, but key behaviors are clearly visible.

1 Introduction

Most commercial information systems, including those adopted by many health care organizations, record information about the executed process instances in the form of an *event log* [15]. The event log stores the sequences (*traces* [9] henceforth) of actions that have been executed at the organization, typically together with key execution parameters, such as times, cost and resources. Event logs can be provided in input to **process mining** [15, 10] algorithms, a family of 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 log traces and build a process model, focusing on its control flow constructs. Classical process mining algorithms, however, provide a purely syntactical analysis, where actions in the traces are processed only referring to their names. Action names are strings without any semantics, so that identical actions, labeled by synonyms, will be considered as different, or actions that are special cases of other actions will be processed as unrelated.

On the other hand, the capability of relating *semantic structures* such as ontologies to actions in the log can enable trace comparison and process mining 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 process analysis much more flexible and reliable.

In fact, it has been observed that human readers are limited in their cognitive capabilities to make sense of large and complex process models [1, 25], while it would be often sufficient to gain a quick overview of the process, in order to familiarize with it in a short amount of time.

Interestingly, **semantic process mining**, defined as the integration of semantic processing capabilities into classical process mining techniques, has been recently proposed in the literature (see Section 5). However, while more work has been done in the field of semantic *conformance checking* (another branch of process mining) [8, 11], to the best of our knowledge semantic *process model discovery* needs to be further investigated.

In this paper, we present a **knowledge-based abstraction mechanism** (see Section 2), able to operate on event log traces. In our approach:

- actions in the log are mapped to the ground terms of an ***ontology***;
- a ***rule base*** is exploited, in order to identify which of the multiple ancestors of an action should be considered for abstracting the action itself. ***Medical knowledge and contextual information*** are resorted to in this step;
- when a set of consecutive actions on the trace abstract as the same ancestor, they are merged into the same abstracted *macro-action*, labeled as the common ancestor at hand. This step requires a proper treatment of delays and/or actions in-between that descend from a different ancestor.

Our abstraction mechanism is then provided as an input to **semantic process mining** (see Section 3). In particular, we rely on classical *process model discovery* algorithms embedded in the open source framework ProM [24], made semantic by the exploitation of domain knowledge in the abstraction phase.

We also describe our experimental work (see Section 4) in the field of stroke care, where the application of the abstraction mechanism on log traces has allowed us to mine simpler and more understandable process models.

2 Knowledge-based trace abstraction

In our framework, trace abstraction has been realized as a multi-step mechanism. The following subsections describe the various steps.

2.1 Ontology mapping

As a first step, every action in the trace to be abstracted is mapped to a ground term of an **ontology**, formalized resorting to domain knowledge.

In our current implementation, we have defined an ontology related to the field of stroke management, where ground terms are patient management actions, while abstracted terms represent medical goals. Figure 1 shows an excerpt of the stroke domain ontology, formalized resorting to the Protègè editor.

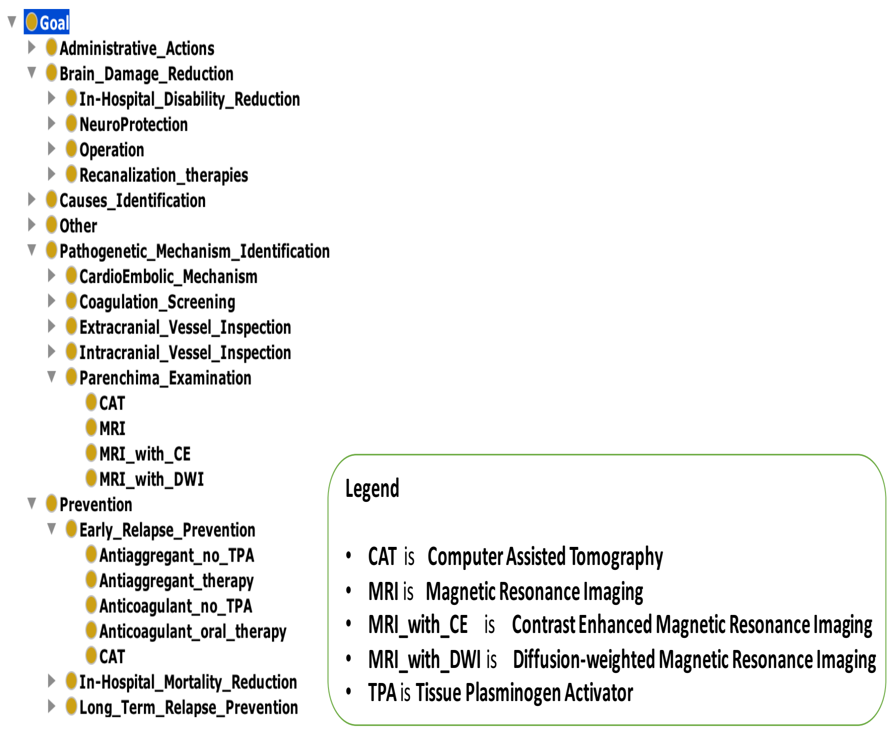

Fig. 1. An excerpt from the stroke domain ontology

In particular, a set of classes, representing the main goals in stroke management, have been identified, namely: "Administrative Actions", "Brain Damage Reduction","Causes Identification", "Pathogenetic Mechanism Identification", "Prevention", and "Other". These main goals can be further specialized into subclasses, according to more specific goals (e.g., "Parenchima Examination" is a subgoal of "Pathogenetic Mechanism Identification", while "Early Relapse Prevention" is a subgoal of "Prevention"), down to the ground actions, that will implement the goal itself.

Some actions in the ontology can be performed to implement different goals. For instance, a Computer Assisted Tomography (CAT) can be used to check therapy efficacy in "Early Relapse Prevention", or to perform "Parenchima Examination" (see figure 1).

The proper goal to be used in the abstraction phase will be selected on the basis of the context of execution, as formalized in the rule base, described in the following subsection.

2.2 Rule-based reasoning for ancestor selection

As a second step in the trace abstraction mechanism, a **rule base** is exploited to identify which of the multiple ancestors of an action in the ontology should be considered for abstracting the action itself. The rule base encodes medical knowledge. Contextual information (i.e., the actions that have been already executed on the patient at hand, and/or her/his specific clinical conditions) is used to activate the correct rules. The rule base has been formalized in Drools [17].

As an example, referring to the CAT action mentioned in the previous subsection, the following rule states that, if intra-venous (ev_tPA) or intra-arterial (ia_tPA) anti-thrombotic therapies have been administered, then CAT implements the "Early Relapse Prevention" goal.

```
rule "CAT"
    when
     (groundActionIsBefore("ev_tPA") ||
      groundActionIsBefore("ia_tPA"))
    then
        macroAction.setAncestorName("Early_Relapse_Prevention");
end
```

where "groundActionIsBefore" is a function that, given the name of a ground action, returns true if this action precedes CAT in the trace, false otherwise.

On the contrary, if the context is different (i.e., anti-thrombotic therapy was not administered), CAT has to be intended as a means for "Parenchima Examination" (see figure 1).

More complex situations, where it is necessary to activate a chain of multiple rules - not described here due to space constraints - can also be managed by our system.

2.3 Trace abstraction

Once the correct ancestor of every action has been identified, trace abstraction can be completed.

In this last step, when a set of consecutive actions on the trace abstract as the same ancestor, they have to be merged into the same abstracted ***macro-action***, labeled as the common ancestor at hand. This procedure requires a proper treatment of *delays*, and of actions in-between that descend from a different ancestor (*interleaved actions* henceforth).

Trace abstraction has been realized by means of the procedure described in Algorithm 1 below.

The function *abstraction* takes in input an event log *trace*, the domain ontology *onto*, and the *level* in the ontology chosen for the abstraction (e.g., $level = 1$ corresponds to the choice of abstracting the actions up to the sons of the ontology root). It also takes in input three thresholds (*delay_th*, *n_inter_th* and *inter_th*). These threshold values have to be set by the domain expert in order to limit the total admissible delay time within a macro-action, the total number of interleaved actions, and the total duration of interleaved actions, respectively. In fact, it would be hard to justify that two ground actions share the same goal (and can thus be abstracted to the same macro-action), if they are separated by very long delays, or if they are interleaved by many/long different ground actions, meant to fulfill different goals.

The function outputs an abstracted trace.

For every action i in *trace*, an iteration is executed (lines 3-27). First, a macro-action m_i, initially containing just i, and sharing its starting and ending times, is created. m_i is labeled referring to the ancestor of i (the one identified by the rule based reasoning procedure) at the abstraction *level* provided as an input. Accumulators for this macro-action (total-delay, num-inter and total-inter, commented below) are initialized to 0 (lines 4-10). Then, a nested cycle is executed (lines 11-25): it considers every element j following i in the trace, where a trace element can be an action, or a delay between a pair of consecutive actions. Different scenarios can occur:

- if j is a delay, $total - delay$ is updated by summing the length of j (lines 12-14).
- if j is an action, and j shares the same ancestor of i at the input abstraction *level*, then j is incorporated into the macro-action m_i. This operation is always performed, provided that $total - delay$, $number - inter$ and $total - inter$ do not exceed the threshold passed as an input (lines 15-19). j is then removed from the actions in *trace* that could start a new macro-action, since it has already been incorporated into an existing one (line 18). This kind of situation is described in Figure 2 (a).
- if j is an action, but does not share the same ancestor of i, then it is treated as an interleaved action. In this case, $num - inter$ is increased by 1, and $total - inter$ is updated by summing the length of j (lines 20-23). This situation, in the end, may generate different types of temporal constraints

Algorithm 1: Multi-level abstraction algorithm

```
abs_trace = abs_algorithm(trace, onto, level, delay_th, n_inter_th, inter_th);
abs_trace = ∅;
for every i ∈ activities in trace do
    if (i.startFlag = yes) then
        create : m_i as ancestor(i, level);
        m_i.start = i.start;
        m_i.end = i.end;
        total_delay = 0;
        num_inter = 0;
        total_inter = 0;
        for (every j ∈ elements in trace) do
            if (j is a delay) then
                total_delay = total_delay + j.length;
            else
                if (ancestor(j, level)=ancestor(i, level)) then
                    if (total_delay < delay_th ∧ num_inter <
                    n_inter_th ∧ total_inter < inter_th) then
                        m_i.end = max(m_i.end, j.end);
                        j.startFlag = no;
                    end
                else
                    num_inter = num_inter + 1;
                    total_inter = total_inter + j.length;
                end
            end
        end
        append m_i to abs_trace;
end
return abs_trace;
```

between macro-actions, as the ones described in Figure 2 (b) (Allen's *during* [2]) and Figure 2 (c) (Allen's *overlaps* [2]).

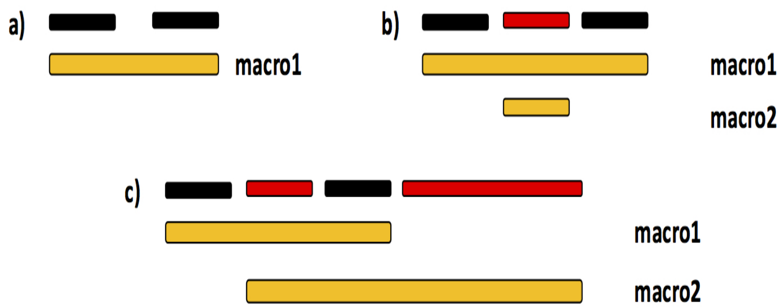

Fig. 2. Different trace abstraction situations: (a) two actions are abstracted to a single macro-action $macro1$, with a delay in between; (b) two actions are abstracted to a macro-action $macro1$, with an interleaved action in between, resulting in a different macro-action $macro2$ during $macro1$; (c) two actions are abstracted to a macro-action $macro1$, with an interleaved action in between, which is later aggregated to a fourth action, resulting in a macro-action $macro2$ overlapping $macro1$.

Finally, the macro-action m_i is appended to abs_trace, that, in the end, will contain the list of all the macro-actions that have been created by the procedure (line 26).

Complexity. The cost of abstracting a trace is $O(actions * elements)$, where *actions* is the number of actions in the input trace, and *elements* is the number of elements (i.e., actions + delay intervals) in the input trace.

3 Semantic process mining

In our approach, process mining, made semantic by the exploitation of the abstraction mechanism illustrated above, is implemented resorting to the well-known process mining tool ProM, extensively described in [24]. ProM (and specifically its newest version ProM 6) is a platform-independent open source framework that supports a wide variety of process mining and data mining techniques, and can be extended by adding new functionalities in the form of plug-ins.

For the work described in this paper, we have exploited ProM's Heuristic Miner [26]. Heuristic Miner is a plug-in for process model discovery, able to mine process models from event logs. It receives in input the log, and considers the order of the actions within every single trace. It can mine the presence of short-distance and long-distance dependencies (i.e., direct or indirect sequence of actions), and information about parallelism, with a certain degree of reliability. The output of the mining process is provided as a graph, known as the "dependency graph", where nodes represent actions, and edges represent control flow information. The output can be converted into other formalisms as well.

Currently, we have chosen to rely on Heuristics Miner, because it is known to be tolerant to noise, a problem that may affect medical event logs (e.g., sometimes the logging may be incomplete). Anyway, testing of other mining algorithms available in ProM 6 is foreseen in our future work.

4 Experimental results

In this section, we describe the experimental work we have conducted, in the application domain of stroke care.

The available event log is composed of more than 15000 traces, collected at the 40 Stroke Unit Network (SUN) collaborating centers of the Lombardia region, Italy. Traces are composed of 13 actions on average. The 40 Stroke Units (SUs) are not all equipped with the same human and instrumental resources: in particular, according to resource availability, they can be divided into 3 classes. Class-3 SUs are top class centers, able to deal with particularly complex stroke cases; class-1 SUs, on the contrary, are the more generalist centers, where only standard cases can be managed.

We have tested whether our capability to abstract the event log traces on the basis of their semantic goals allowed to obtained process models where unnecessary details are hidden, but key behaviors are clear. Indeed, if this hypothesis holds, in our application domain it becomes easier to compare process models of different SUs, highlighting the presence/absence of common paths, regardless of minor action changes (e.g., different ground actions that share the same goal) or irrelevant different action ordering or interleaving (e.g., sets of ground actions, all sharing a common goal, that could be executed in any order).

Figure 3 compares the process models of two different SUs (SU-A and SU-B), mined by resorting to Heuristic Miner, operating on ground traces. Figure 4, on the other hand, compares the process models of the same SUs as figure 3, again mined by resorting to Heuristic Miner, but operating on traces abstracted according to the goals of the ontology in figure 1. In particular, abstraction was conducted up to level 2 in the ontology (where level 0 is the root, i.e.. "Goal").

Generally speaking, a visual inspection of the two graphs in figure 3 is very difficult. Indeed, these two ground processes are "spaghetti-like" [9], and the extremely large number of nodes and edges makes it hard to identify commonalities in the two models.

The abstract models in figure 4, on the other hand, are much more compact, and it is possible for a medical expert to analyze them.

In particular, the two graphs in figure 4 are not identical, but in both of them it is easy to a identify the macro-actions which corresponds to the treatment of a typical stroke patient.

However, the model for SU-A at the top of figure 4 exhibits a more complex control flow (with the presence of loops), and shows three additional macro-actions with respect to the model of SU-B, namely "Extracranial Vessel Inspec-

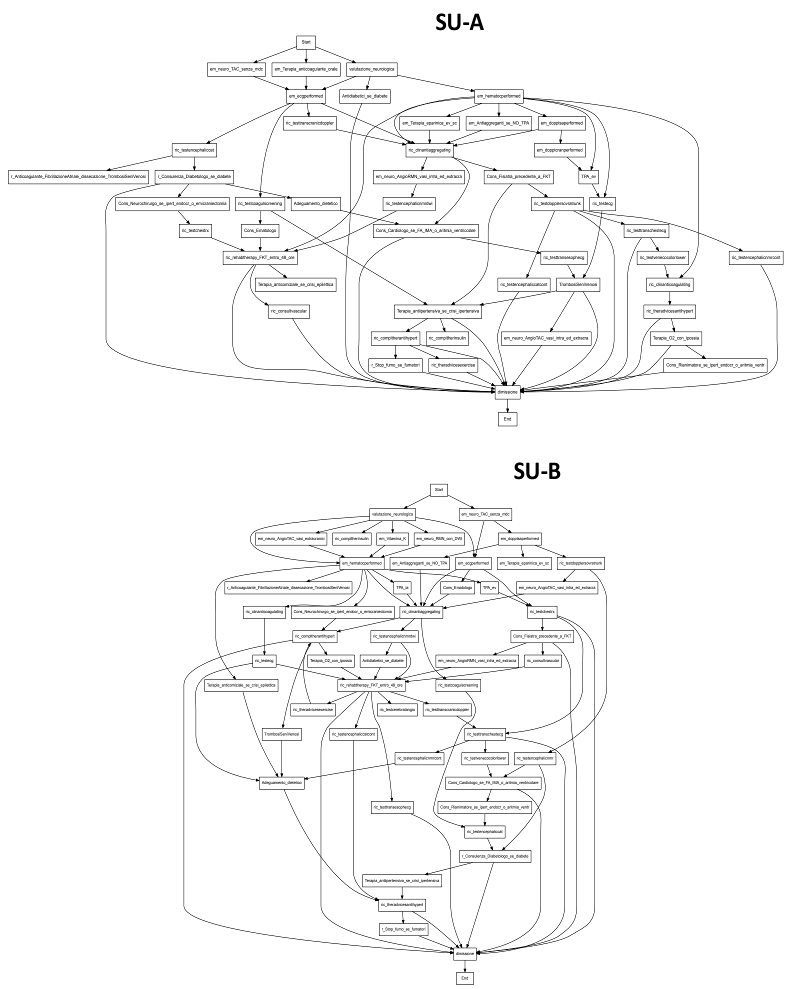

Fig. 3. Comparison between two process models, mined by resorting to Heuristic Miner, operating on ground traces. The figure is not intended to be readable, but only to give an idea of how complex the models can be

tion", "Intracranial Vessel Inspection" and "Recanalization". This finding can be explained, since SU-A is a class-2 SU, where different kinds of patients, including some atypical/more critical ones, can be managed, thanks to the availability of different skills and instrumental resources. These patients may require the additional macro-actions reported in the model, and/or the repetition of some procedures, in order to better characterize and manage the patient's situation.

On the other hand, SU-B is a class-1 SU, i.e., a more generalist one, where very specific human knowledge or technical resources are missing. As a consequence, the overall model control flow is simpler, and some activities are not executed at all.

Interestingly, our abstraction mechanism, while hiding irrelevant details, allows to still appreciate these differences.

5 Related works

The use of semantics in business process management, with the aim of operating at different levels of abstractions in process discovery and/or analysis, is a relatively young area of research, where much is still unexplored.

One of the first contributions in this field was proposed in [5], which introduces a process data warehouse, where taxonomies are exploited to add semantics to process execution data, in order to provide more intelligent reports. The work in [12] extends the one in [5], presenting a complete architecture that allows business analysts to perform multidimensional analysis and classify process instances, according to flat taxonomies (i.e., taxonomies without subsumption relations between concepts).

Hepp et al. [13] propose a framework able to merge semantic web, semantic web services, and business process management techniques to build semantic business process management, and use ontologies to provide machine-processable semantics in business processes [14]. The work in [21] develops in a similar context, and extends OLAP tools with semantics (exploiting ontologies rather than (flat) taxonomies).

The topic was studied in the SUPER project [20], within which several ontologies were created, such as the process mining ontology and the event ontology [19]; these ontologies define core terminologies of business process management, usable by machines for task automation. However, the authors did not present any concrete implementations of semantic process mining or analysis.

Ontologies, references from elements in logs to concepts in ontologies, and ontology reasoners (able to derive, e.g., concept equivalence), are described as the three essential building blocks for semantic process mining in [8]. This paper also shows how to use these building blocks to extend ProM's LTL Checker [23] to perform semantic auditing of logs.

The work in [6] focuses on the use of semantics in business process monitoring, an activity that allows to detect or predict process deviations and special situations, to diagnose their causes, and possibly to resolve problems by applying

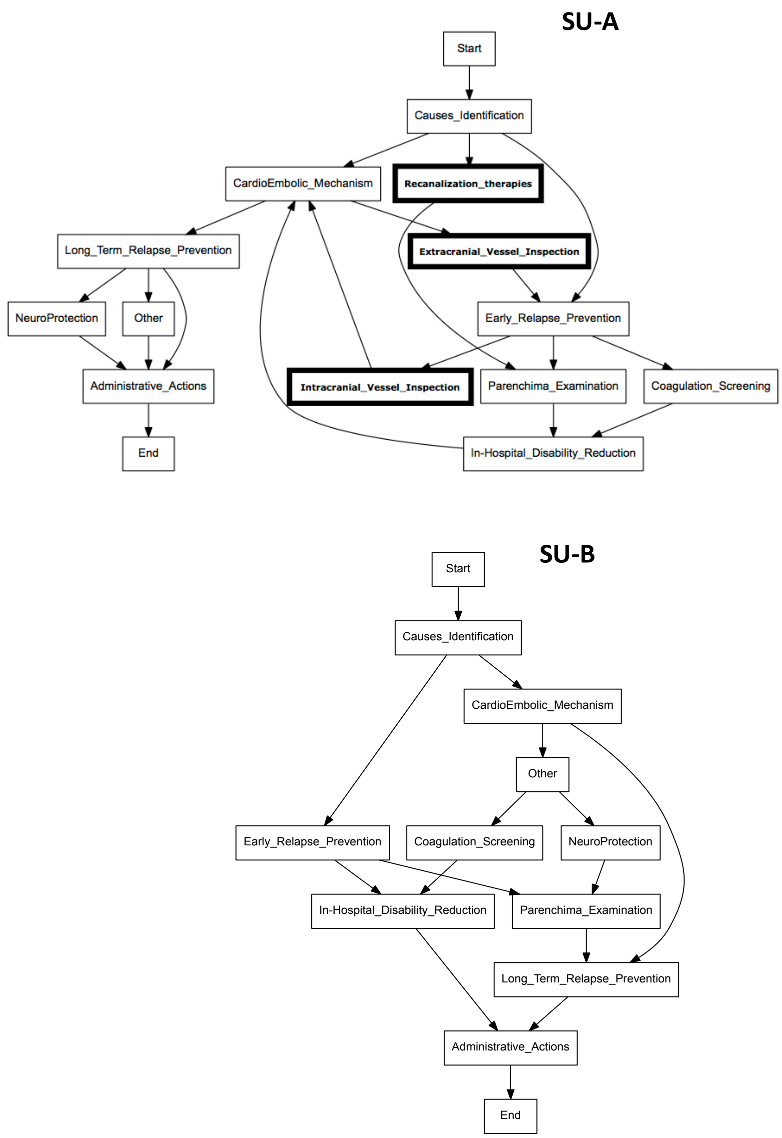

Fig. 4. Comparison between the two process models of the same SUs as figure 3, mined on abstracted traces. Additional macro-actions executed at SU-A are highlighted in bold

corrective actions. Detection, diagnosis and resolution present interesting challenges that, on the authors' opinion, can strongly benefit from knowledge-based techniques.

In [6, 7] the idea to explicitly relate (or annotate) elements in the event log with the concepts they represent, linking these elements to concepts in ontologies, is addressed.

In [7] an example of process discovery at different levels of abstractions is presented. It is however a very simple example, where a couple of ground actions are abstracted according to their common ancestor. However, the management of interleaved actions or delays is not addressed, and multiple inheritance is not considered. A more recent work [16] introduces a methodology that combines domain and company-specific ontologies and databases to obtain multiple levels of abstraction for process mining. In this paper data in databases become instances of concepts at the bottom level of a taxonomy tree structure. If consecutive tasks in the discovered model abstract as the same concepts, those tasks are aggregated. However, also in this work we could find neither a clear description of the abstraction algorithm, nor the management of interleaved actions or delays.

Other interesting contributions can be found in [4, 3, 22].

However, most of the papers cited above (including [8, 7]) present theoretical frameworks, and not yet a detailed technical architecture nor a concrete implementation of all their ideas.

Referring to medical applications, the work in [11] proposes an approach, based on semantic process mining, to verify the compliance of a Computer Interpretable Guideline with medical recommendations. In this case, semantic process mining refers to conformance checking rather than to process discovery (as it is also the case in [8]). These works are thus only loosely related to our contribution.

In conclusion, in the current research panorama, our work appears to be very innovative, for several reasons:

- many approaches, illustrating very interesting and sometimes ambitious ideas, just provide pure theoretical frameworks, which can be very important to inspire more engineering-style work. However, concrete implementations of algorithms and complete architectures of systems are often missing, leaving open research opportunities for contributions like the one we have presented;
- in semantic process mining, more work has been done in the field of conformance checking (also in medical applications), while process discovery still deserves attention (also because many approaches are still at the theoretical level, as commented above);
- as regards trace abstraction, it is often proposed as a very powerful means to obtain better process discovery and analysis results, but technical details of the abstraction mechanism are usually not provided, or are illustrated through very simple examples, where the issues related to the management of interleaved actions or delays do not emerge.

6 Concluding remarks and future work

In this paper, we have presented a framework for knowledge-based abstraction of event log traces. In our approach, abstracted traces are then provided as an input to semantic process mining. Semantic process mining relies on ProM algorithms; indeed, the overall integration of our approach within ProM is foreseen in our future work.

The first experimental results in the field of stroke management suggest that the capability of abstracting the event log traces on the basis of their semantic goal may allow to mine clearer process models, where unnecessary details are hidden, but key behaviors are clear.

In the future, we plan to conduct a validation study, by quantitatively comparing different process models (of different SUs) obtained from abstracted traces. Comparison will resort to knowledge-intensive process similarity metrics, such as the one we described in [18]. We will also extensively test the approach in different application domains.

Finally, an abstraction mechanism directly operating on process models (i.e., on the graph, instead of the event log), may be considered, and abstraction results will be compared to the ones currently enabled by our framework.

References

1. E. Rolón Aguilar, F. Ruiz, F. García, and M. Piattini. Evaluation measures for business process models. In H. Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 1567–1568. ACM, 2006.
2. J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.
3. A. Azzini, C. Braghin, E. Damiani, and F. Zavatarelli. Using semantic lifting for improving process mining: a data loss prevention system case study. In R. Accorsi, P. Ceravolo, and P. Cudré-Mauroux, editors, *Proceedings of the 3rd International Symposium on Data-driven Process Discovery and Analysis*, volume 1027 of *CEUR Workshop Proceedings*, pages 62–73. CEUR-WS.org, 2013.
4. A. Azzini and P. Ceravolo. Consistent process mining over big data triple stores. In *IEEE International Congress on Big Data, BigData Congress 2013*, pages 54–61. IEEE Computer Society, 2013.
5. F. Casati and M.C. Shan. Semantic analysis of business process executions. In C. S. Jensen, K. G. Jeffery, J. Pokorný, S. Saltenis, E. Bertino, K. Böhm, and M. Jarke, editors, *Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27, Proceedings*, volume 2287 of *Lecture Notes in Computer Science*, pages 287–296. Springer, 2002.
6. A. K. Alves de Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral. An outlook on semantic business process mining and monitoring. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM*

Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part II, volume 4806 of *Lecture Notes in Computer Science*, pages 1244–1255. Springer, 2007.

7. A. K. Alves de Medeiros and W. M. P. van der Aalst. Process mining towards semantics. In T. S. Dillon, E. Chang, R. Meersman, and K. P. Sycara, editors, *Advances in Web Semantics I - Ontologies, Web Services and Applied Semantic Web*, volume 4891 of *Lecture Notes in Computer Science*, pages 35–80. Springer, 2009.
8. A. K. Alves de Medeiros, W. M. P. van der Aalst, and C. Pedrinaci. Semantic process mining tools: Core building blocks. In W. Golden, T. Acton, K. Conboy, H. van der Heijden, and V. K. Tuunainen, editors, *16th European Conference on Information Systems, ECIS 2008, Galway, Ireland, 2008*, pages 1953–1964, 2008.
9. W. Van der Aalst. *Process Mining. Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
10. W. Van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters. Workflow mining: a survey of issues and approaches. *Data and Knowledge Engineering*, 47:237–267, 2003.
11. M. A. Grando, M. H. Schonenberg, and W. M. P. van der Aalst. Semantic process mining for the verification of medical recommendations. In V. Traver, A. L. N. Fred, J. Filipe, and H. Gamboa, editors, *HEALTHINF 2011 - Proceedings of the International Conference on Health Informatics, Rome, Italy, 26-29 January, 2011*, pages 5–16. SciTePress, 2011.
12. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
13. M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. Semantic business process management: A vision towards using semantic web services for business process management. In F. C. M. Lau, H. Lei, X. Meng, and M. Wang, editors, *2005 IEEE International Conference on e-Business Engineering (ICEBE 2005), 18-21 October 2005, Beijing, China*, pages 535–540. IEEE Computer Society, 2005.
14. M. Hepp and D. Roman. An ontology framework for semantic business process management. In A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, and B. Schnizler, editors, *eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik - Band 1, WI 2007, Karlsruhe, Germany, February 28 - March 2, 2007*, pages 423–440. Universitaetsverlag Karlsruhe, 2007.
15. $http : //www.win.tue.nl/ieeetfpm/doku.php?id = shared : process_mining_manifesto$. IEEE Taskforce on Process Mining: Process Mining Manifesto (last accessed on 22/11/2017).
16. W. Jareevongpiboon and P. Janecek. Ontological approach to enhance results of business process mining and analysis. *Business Proc. Manag. Journal*, 19(3):459–476, 2013.
17. M. N. De Maio M. Salatino, E. Aliverti. *Mastering JBoss Drools 6 for Developers*. Packt Publishing, 2016.
18. S. Montani, G. Leonardi, S. Quaglini, A. Cavallini, and G. Micieli. A knowledge-intensive approach to process similarity calculation. *Expert Syst. Appl.*, 42(9):4207–4215, 2015.
19. C. Pedrinaci and J. Domingue. Towards an ontology for process monitoring and mining. In M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, and N. Stojanovic, editors, *Proceedings of the Workshop on Semantic Business Process and Product*

Lifecycle Management SBPM 2007, held in conjunction with the 3rd European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, June 7, 2007, volume 251 of *CEUR Workshop Proceedings*, 2007.
20. C. Pedrinaci, J. Domingue, C. Brelage, T. van Lessen, D. Karastoyanova, and F. Leymann. Semantic business process management: Scaling up the management of business processes. In *Proceedings of the 2th IEEE International Conference on Semantic Computing (ICSC 2008), August 4-7, 2008, Santa Clara, California, USA*, pages 546–553. IEEE Computer Society, 2008.
21. D. Sell, L. Cabral, E. Motta, J. Domingue, and R. C. dos Santos Pacheco. Adding semantics to business intelligence. In *16th International Workshop on Database and Expert Systems Applications (DEXA 2005), 22-26 August 2005, Copenhagen, Denmark*, pages 543–547. IEEE Computer Society, 2005.
22. A. Syamsiyah, B. van Dongen, and W. van der Aalst. DB-XES: enabling process discovery in the large. In *Proceedings of the 6th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA*.
23. W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In R. Meersman, Z. Tari, M. Hacid, J. Mylopoulos, B. Pernici, Ö. Babaoglu, H. Jacobsen, J. P. Loyall, M. Kifer, and S. Spaccapietra, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2005.
24. B. van Dongen, A. Alves De Medeiros, H. Verbeek, A. Weijters, and W. Van der Aalst. The proM framework: a new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *Knowledge Mangement and its Integrative Elements*, pages 444–454. Springer, Berlin, 2005.
25. I. T. P. Vanderfeesten, H. A. Reijers, J. Mendling, W. M. P. van der Aalst, and J. S. Cardoso. On a quest for good process models: The cross-connectivity metric. In Z. Bellahsene and M. Léonard, editors, *Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008, Montpellier, France, June 16-20, 2008, Proceedings*, volume 5074 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 2008.
26. A. Weijters, W. Van der Aalst, and A. Alves de Medeiros. *Process Mining with the Heuristic Miner Algorithm, WP 166*. Eindhoven University of Technology, Eindhoven, 2006.