Leveraging Semantic Labels for Multi-level Abstraction in Medical Process Mining and Trace Comparison

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

[a]*DISIT, Computer Science Institute, Università del Piemonte Orientale, Viale Michel 11, I-15121 Alessandria, Italy*
[b]*Department of Computer Science, Università di Torino, Italy*
[c]*Department of Electrical, Computer and Biomedical Engineering, Università di Pavia, Via Ferrata 1, I-27100 Pavia, Italy*
[d]*Istituto di Ricovero e Cura a Carattere Scientifico Fondazione "C. Mondino", Via Mondino 2, I-27100 Pavia, Italy - on behalf of the Stroke Unit Network (SUN) collaborating centers*

Abstract

Many medical information systems record data about the executed process instances in the form of an *event log*. In this paper, we present a framework, able to convert actions in the event log into higher level concepts, at different levels of abstraction, on the basis of domain knowledge. Abstracted traces are then provided as an input to trace comparison and semantic process discovery. Our abstraction mechanism is able to manage non trivial situations, such as interleaved actions or delays between two actions that abstract to the same concept. Trace comparison resorts to a similarity metric able to take into account abstraction phase penalties, and to deal with quantitative and

*Corresponding author. Tel. +39 0131 360158
Published version at: https://doi.org/10.1016/j.jbi.2018.05.012
Email address: `stefania.montani@uniupo.it` (Stefania Montani)

qualitative temporal constraints in abstracted traces. As for process discovery, we rely on classical algorithms embedded in the framework ProM, made *semantic* by the capability of abstracting the actions on the basis of their conceptual meaning. The approach has been tested in stroke care, where we adopted abstraction and trace comparison to cluster event logs of different stroke units, to highlight (in)correct behavior, abstracting from details. We also provide process discovery results, showing how the abstraction mechanism allows to obtain stroke process models more easily interpretable by neurologists.
Keywords: Abstraction, Trace comparison, Semantic process mining, Stroke management

1. Introduction

Today's medical information systems log enormous amounts of data, including details about the actions that have been executed at a given organization. Such data collections can be provided in the form of *event logs* [1], which maintain the sequences (*traces* [1] henceforth) of actions that have been completed, identified by their *names*, together with their *timestamps* and possible additional data elements.

Event logs are exploited by **process mining** [1], a term that encopasses a family of a-posteriori analysis techniques. Basically, event logs can be used to feed and run four types of process mining tasks[2, 3, 1]:

- *discovery*. A discovery technique takes as input the event log and produces a process model. Discovery is the most relevant and widely used process mining activity;

- *conformance.* A conformance technique takes as input the event log and an existing process model: the model is compared to the log of the same process, to measures the alignment (i.e., the conformance) between the model and reality;

- *enhancement.* An enhancement technique takes as input the event log and an existing process model as well. It aims at improving the existing model using information about the actual process recorded in the log; in fact, if needed, the a-priori model can be changed, to better mirror the log data.

- *operational support.* An operational support technique does not operate off-line, as in the previous cases. On the contrary, it is used to influence the running process instance by checking, predicting, or recommending actions to be executed. It typically resorts to the comparison and/or analysis of past traces similar to the running one.

The action *names* maintained in the traces in the event log 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 by the process mining techniques illustrated above.

On the other hand, the capability of relating *semantic structures* such as ontologies to actions in the log can enable all such tasks 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 mining 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 [4, 5], 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. One well-known way to address this issue is by applying *process model abstraction* [6], thus retaining the essential properties of the model on a particular level of analysis, while simultaneously hiding insignificant details for that level. Of course, deeper investigations can still be conducted, subsequently, on the detailed (ground) process model.

Interestingly, *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., [7, 8], and Section 6). However, while more work has been done in the field of semantic conformance checking [7, 9], to the best of our knowledge semantic process discovery and semantic-based trace analysis for operational support need to be further investigated.

In this paper, we present a **semantic-based, multi-level abstraction mechanism**, able to operate on event log traces. In our approach:

- actions in the event log are related to the medical goals they are aimed to fulfill, by means of an ***ontology***;
- a ***rule base*** is exploited, in order to identify which of the multiple goals of an action in the ontology should be considered as the correct abstraction of the action itself.

The abstraction mechanism is then provided as an input to further analysis mechanisms, namely **trace comparison** and **process discovery**.

The methodological approach has been tested in the field of stroke patient management, where we have adopted our framework for trace clustering and process model discovery. In the first case, we have verified that it is possible to obtain more homogeneous clusters, abstracting from details such as local resource constraints or local protocols, but still preserving the capability of isolating outlying situations; in the second experiment, we have mined more readable process models, where unnecessary details are hidden, but key behaviors are clear.

The paper is organized as follows. Section 2 introduces a motivating example in the field of stroke care. Section 3 presents the terminology that will be adopted in the paper. Section 4 presents methodological and technical details of the framework. Section 5 describes experimental results. Section 6 addresses comparisons with related works and discusses the novelty of the approach. Finally, Section 7 is devoted to conclusions and future research directions.

2. The need for multi-level abstraction in stroke care

In this section, we present our methodology in the context of a real-life scenario, where we aim at comparing two process traces executed at two different hospitals for the treatment of patients with ischemic stroke. The traces in Figure 1 show the sequence of actions performed by hospital T1 and T2, respectively, to treat two patients admitted in their emergency department and potentially affected by ischemic stroke in the acute phase. In hospital T1, the patient firstly undergoes a contrast-enhanced Computed Tomography scan (CAT-C), then a Diabetologist Counseling (DC), then a

Magnetic Resonance (MRI). After these examinations, the patient is treated with an Endovascular Procedure (EP) and with the administration of Anti-Aggregants (AAG). Hospital T2, instead, performs a different sequence on its patient: it starts with a MRI, then it continues by performing the CAT-C, then treating the patient with thrombolysis (tissue plasminogen activator - TPA) and finally by administering an Anticoagulant Oral Therapy (AOT).

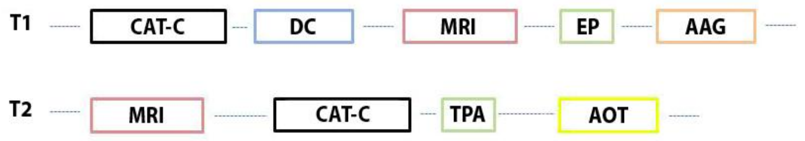

Figure 1: Traces showing the treatment of two stroke patients in two different hospitals

Apparently, the traces in Figure 1 describe two very different behaviors, considering both the actions and their order. However, the Italian guidelines for the treatment of stroke (ISO-SPREAD guidelines[1]), offer the domain knowledge for a more abstract interpretation.

According to the guidelines, in the acute phase some of the goals to be pursued are the identification of the pathogenetic mechanism of the ischemic stroke and its localization, the reduction of the brain damage through re-canalization therapies, if possible, and the secondary prevention to lower the risk of recurrence. The pathogenetic mechanism and localization of stroke can be investigated through neuro-imaging tests such as a CAT-C scan or a MRI, potentially executed both, in any order, for a deeper analysis. The main therapy to reduce brain damage is the TPA drug, which tries to dissolve the

[1]http://www.iso-spread.it/index.php, last accessed on 11/09/2017

thrombus and to restore the brain circulation in order to mitigate damages from ischemia, but mechanical revascularization through endovascular procedures could be performed, if necessary and if the patient is not eligible for TPA. Early relapse prevention can be obtained through the administration of anti-thrombotic drugs, mainly antiaggregants or anticoagulants.

Applying this knowledge to the traces in Figure 1, it is possible to obtain a higher level interpretation of the operations performed by the two hospitals, by means of abstracted traces that show the therapeutic and diagnostic goals instead of the ground actions. This "bird's eye view" allows to perform a comparison that ignores unnecessary details. In trace T1, CAT-C and MRI are merged into a single macro-action PE (Parenchima Examination); EP abstracts to a macro-action RT (Recanalization Therapy); AAG becomes an ERP (Early Relapse Prevention) macro-action. It is worth noting that in the trace T1, a DC is also executed. This action, however, is not part of any of the main goals to be achieved in the management of the acute phase, therefore it remains an isolated action, performed during PE. Regarding T2, MRI and CAT-C are abstracted as PE; TPA is abstracted as RT, while AOT is abstracted as ERP. The result of the abstraction process on the traces T1 and T2 is shown in Figure 2.

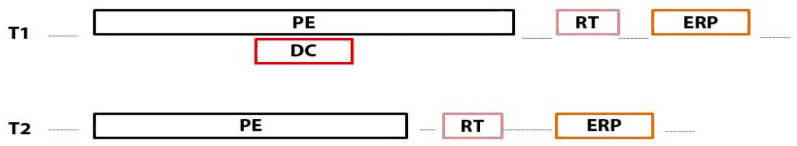

Figure 2: Abstraction of the traces in Figure 1

Comparing T1 and T2 at this higher level of abstraction, we can observe that they are indeed very similar: ground differences do exist, but they can be interpreted as minor variations of two process instances, which share the same goals, and are thus basically compliant with the guideline indications.

Interestingly, the abstraction technique is not only able to shed the light on the compatibility of apparently different behaviors, but is also able to preserve significant differences. For example, the abstracted traces in Figure 2 show even more clearly than the ones in Figure 1, that hospital T1 performs a single DC action not in line with the goals suggested by guideline, while hospital T2 is compliant with the guideline recommendations. This situation suggests an investigation of the causes for this non-compliance to be carried out by the T1 managers, who can conclude, for example, that the additional DC is driven by an excess of precaution: it could be considered as a waste of resources and time, if not even a damage for the patient, since other critical procedures could be delayed (DC could be useful in case of a diabetic patient indeed, but it should be performed after the acute phase, when it is very important that most urgent examinations and treatments are delivered as quick as possible).

3. Terminology

The terminology we will use henceforth is briefly summarized below:

Action or **ground action**: an action recorded in an event log trace.

Delay: time interval between two ground actions logged in a trace, within which no other action takes place.

Interleaved action: a ground action that, with respect to two ground actions being considered for abstraction, implements a different goal, and is placed between the two actions at hand.

Macro-action: partial output of the abstraction process; a macro-action is an abstracted action that covers the whole time span of multiple ground actions, and is labeled as their common goal in the ontology, at the specified abstraction level. As a special case, the macro-action can abstract a single ground action as its goal.
Abstracted trace: global output of the abstraction process; an abstracted trace is the transformation of an input trace into a new trace containing only macro-actions.

4. The multi-level abstraction framework

The architecture and the data flow of the framework we have developed are shown in Figure 3.

The first step to be executed when adopting the framework is *event log preparation*, which takes as input the available database (containing action execution information, such as timestamps, and additional data, like, e.g., patient's demographics and clinical data in our application domain), and exploits domain knowledge sources (an ontology and a rule base). This step generates an event log where traces are represented in an eXtensible Event Stream (XES) [10] file. The XES format is an extension of the MXML [11] format, where elements have an optional extra attribute called $modelReference$. This attribute allows to link an action to a concept in the ontology. Proper action attributes also allow to record contextual information to be used by rules, as it will be exemplified in Section 4.1.

More in detail, the *event log preparation* is articulated as follows:

1. we analyse the available database tables and their contents, for the

identification of the actions performed by the personnel to manage and treat the patients. In our current application, we rely upon the Stroke Registry, a common database exploited by all the Stroke Unit Network (SUN) collaborating centers of the Lombardia region, Italy. The Stroke Registry is strongly structured, all actions are coded, and action starting/ending times are explicitly reported. Moreover, the logging granularity is the one of ground actions in our ontology. The quality of data was carefully considered during the design and development of the Stroke Registry user's interface, which contains a lot of data input checks, constraints on the entered values (e.g., admissible value ranges), and strategies to minimize missing data, thus we are confident on a good data quality [12];

2. we exploit starting/ending times to sequentialize actions; we can thus reorganize the information of the Stroke Registry into a set of execution traces, where each trace contains the reconstruction of the clinical history of each patient, in the correct temporal order;
3. we translate the traces to the XES format, and complete all action attribute values, including $modelReference$.

The event log then undergoes *multi-level abstraction*, which resorts to the domain knowledge sources as well, and will be described in Section 4.2. The abstracted event log can be given as an input to *trace comparison*, presented in Section 4.3, or to *process discovery*, currently realized as illustrated in Section 4.4.

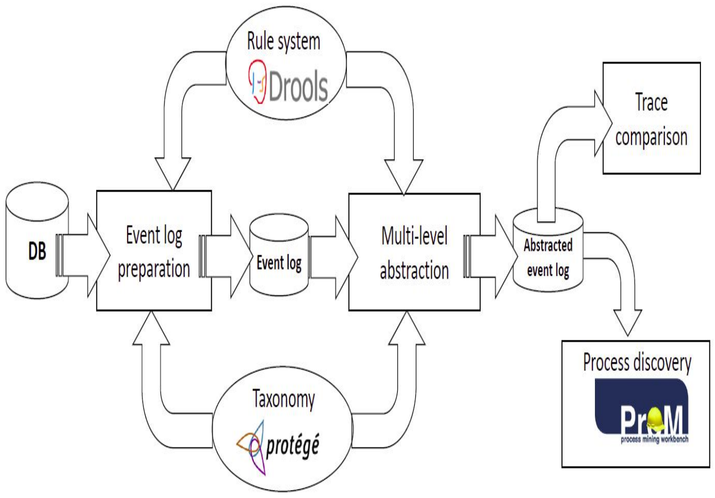

Figure 3: Framework architecture and data flow

4.1. Domain knowledge sources

In our framework, domain knowledge is provided by means of an ontology and of a rule base. In the paper, we will refer to the domain of stroke management.

An excerpt of our stroke management ontology is reported in Figure 4.

The **ontology**, which has been formalized by using the Protègè editor, extends the one we presented in [13, 14][2].

In detail, it comprises:

- a *goal taxonomy*, composed by a set of classes, representing the main goals in stroke management, namely: "(Secondary) Prevention", "Pathogenetic Mechanism Identification", "Brain Damage Reduction" and

[2]A partial view of the ontology, closely related to the examples and experiments of this paper, can be accessed in the form of Protègè files at https://drive.google.com/file/d/1yInh0WwJrI0NK42cA6-NW6-j91V2VafA/view?usp=sharing

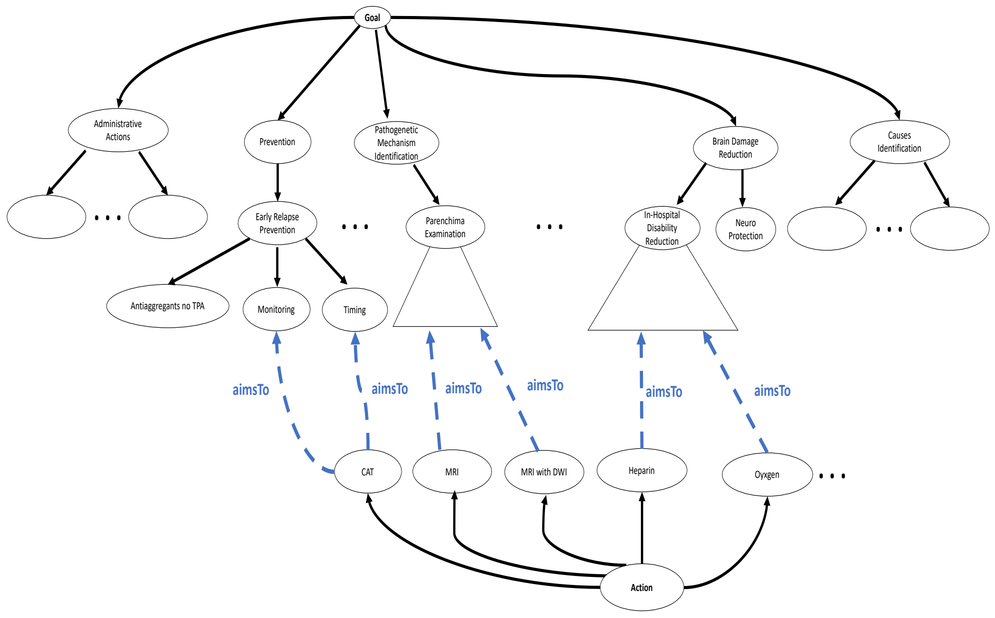

Figure 4: An excerpt from the stroke domain ontology

"Causes Identification". Moreover, the class "Administrative Actions" collects procedures related to the administrative aspects of health care, such as admission, discharge, transfer, and so forth. These main goals can be further specialized into subclasses, according to more specific goals (relation "is-a"; e.g., "Early Relapse Prevention" is a subgoal of "Prevention");

- an *action taxonomy*, composed by all the ground actions that can be logged in stroke management traces (in "is-a" relation with the general class "Action");

- a set of "aimsTo" relations, which formalize that a ground action can be executed to implement a (sub)goal. Multiple "aimsTo" relations could connect a given ground action to different goals (e.g., CAT (Computer

Aided Tomography) can implement “Monitoring” or “Timing” even within the same guideline).

All ground actions are being mapped to SNOMED concepts[3]. The ontology partially showed in Figure 4 is therefore being integrated with the most comprehensive and precise clinical health terminology product in the world, accepted as a common standard for health terms.

In the abstraction step (see Section 4.2), the user will specify at what level of the goal taxonomy s/he wants to operate, i.e., whether at the level of the most general goals (level 1, since level 0 is the root “Goal” in Figure 4), or at the level of progressively more specific subgoals.

In the case of multiple “aimsTo” relations, the proper goal to be used to abstract a given action will be selected by the rule base, which extends a previous version published in [13].

The **rule base**, in fact, encodes medical knowledge as well. Contextual information (i.e., the actions that have been already executed on the patient at hand and are logged earlier in the trace, and/or her/his specific clinical conditions) is used to activate the correct rules. The rules are thus meant to infer why the action was performed in practice, in the specific trace at hand[4].

The rule base has been formalized in Drools [15]. Patient findings have been mapped to SNOMED concepts as well.

As an example, referring to the CAT action, the rules reported below

[3]http://www.snomed.org/snomed-ct, last accessed on 11/09/2017

[4]Note that, if this disambiguation is not needed, because multiple goals are admissible for the same action in the given trace, the system can be easily adapted, by simply skipping the rule-based step, and, e.g., maintaining the indication of all the plausible goals (thus allowing for multiple mergers of the abstracted macro-action, see Section 4.2). This direction will be considered in our future work.

(showed as a simplified pseudocode with respect to the system internal representation, for the sake of simplicity) state that, if the patient has experienced a severe brain damage and suffers from atrial fibrillation, s/he must initiate a proper therapy. Such a therapy starts with ASA (a class of anti-inflammatory drugs), and continues with daily AC (anti-coagulant drug) administration. Before the first AC, a CAT is required, to assess AC starting time, which could be delayed in case CAT detects a hemorrhagic transformation. After a few days of AC administration, another CAT is needed, to monitor therapeutic results. Therefore, depending on the context, CAT can implement the "Timing" or the "Monitoring" goal (see Figure 4). Forward chaining on the rules below allows to determine the correct goal for the CAT action.

```
rule "SevereDamage"
    when
      (
         Damage(value > threshold) &&
         AtrialFibrillation(value=true)
       )
     then
        logicalInsertFact (DamFib);
end

rule "Fibrillation1"
    when
        existInLogical(DamFib) &&
        isBefore("CAT", "AC")
     then
        setGoalName("CAT", "Timing");
end

rule "Fibrillation2"
    when
        existInLogical(DamFib) &&
       isAfter("CAT", "AC")
     then
```

```
        setGoalName("CAT", "Monitoring");
end
```

4.2. Abstraction

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

- two actions having the same goal (relation “aimsTo”) in the ontology may be separated in the trace by a delay, or by interleaved actions. Our approach allows to deal with these situations, by creating a single macro-action; 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 medical expert. The delays and interleaved actions are quantified and recorded, for possible use in further analyses. In particular, in Section 4.3 we will present a **similarity metric** where this information is accounted for as a penalty, and affects the similarity value in abstracted 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. An example will be shown in Section 4.2.2. 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 manage all types of temporal constraints.

To deal with these issues, we have defined a dynamic programming algorithm, able to identify, for every goal a at the input abstraction level, the best (i.e., the longest in time) macro-actions that abstract as a, independently of where (i.e., in correspondence of what ground action) they start in the trace. This optimization criterion (defining macro-actions with the longest duration), indicated by our medical co-author, generates abstracted traces with a low number of macro-actions. Possible different criteria could be considered in the future, and, given the modularity of the implementation, this change would only affect a few portions of the code.

The algorithm is illustrated in the following Section.

4.2.1. Dynamic programming algorithm

Our dynamic programming multi-level abstraction procedure takes as input an event log trace, the domain ontology, the rule base, and the level in the ontology chosen for the abstraction (where level 1 corresponds to an abstraction up to the most general goals, i.e., the children of the root "Goal" in Figure 4). It also takes as 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.

First, an initialization phase is conducted, where every action i in the input trace is labeled referring to the goal of i in the ontology, at the abstraction *level* provided as an input. The correct goal to be chosen is identified exploiting the ontology and the rule base, as explained in Section 4.1. This activity will partition the actions in the input trace.

At the end of the initialization phase, for every element in the partition (i.e., for every set of actions in the trace sharing the same goal a), Algorithm 1 is applied. Function $abs\_dynProgr$ takes as input the actions in the *partition* element, the number $n(a)$ of actions that belong to the partition element itself, the input *trace*, and the three thresholds discussed above. It outputs a triangular matrix M. The matrix M, whose dimension is $n(a) * n(a)$ (line 2), maintains the solutions of subproblems.

Elementary subproblem solutions will be stored along the diagonal: $M[i, i]$ (line 10) will contain the length of the macro-action starting at the beginning of the i-th action in the partition element, and ending at the end of the i-th action itself (lines 4-6; indeed, in the elementary subproblem, the macro-action contains only the ground action i, that has been properly abstracted, i.e., labeled as its goal). In the i-th cycle, a set of accumulators for this macro-action ($totaldelay_i$, num_inter_i and $total\_inter_i$) are also initialized to 0 (lines 7-9).

After the elementary subproblem calculation, the rest of the matrix cells can be filled, row by row (lines 12-22). For every row i, an iteration is executed, considering all the actions in the partition element that follow the i-th action (lines 13-21): accumulators ($total\_delay_i$, num_inter_i and $total\_inter_i$) are updated, by taking into account, respectively, the total delay, the number of interleaved actions, and the total length of the interleaved actions that appear in the input trace between action i and action j (lines 14-16). If the accumulators do not exceed the thresholds passed as an input, the ending time of the macro-action starting at the beginning of the i-th action is updated to include action j (line 18), and the macro-action is stored in $M[i, j]$ (line 19). If any of the accumulators exceeds the threshold, the row will not be completely filled. In this way, the last filled cell for every row corresponds to the longest macro-action that can be abstracted starting at the i-th action in the input trace.

Once the matrix has been filled, our procedure identifies the optimal solution, i.e., the macro-action of maximal time length in M. This macro-action is provided as a solution item in the abstracted trace. Later, rows and columns involving the optimal macro-action are deleted, in order to eliminate other macro-actions that are included in (or are started by) the optimal one, and to split macro-actions that are ended by the optimal one, so that only the non-overlapping prefixes will be considered further. The process of optimum identification is then repeated on every sub-matrix obtained after this row and column deletion.

In the end, a set of macro-actions, abstracted as the goal of the given partition element, is provided by our approach; these macro-actions will be appended to the output abstracted trace. This part of the code is not shown, for the sake of brevity.

Matrix building and optimal solution identification are repeated for every partition, to build the overall output abstracted trace. The procedure, in the end, may generate different types of temporal constraints [16] between

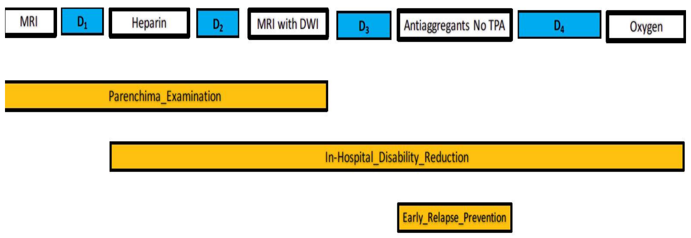

Figure 5: Abstraction example. Ground actions (white) and delays (blue) are reported in the top layer (rectangles width is proportional to elapsed time), while the subsequent three layers report the abstracted macro-actions on the same timeline

macro-actions, as it will be exemplified in Section 4.2.2.

Complexity.The cost of abstracting a trace is $O(\sum_{a=1}^{partition-elements} n(a) * n(a))$, where $n(a)$ is the number of actions in the a-th partition element.

Property. The dynamic programming algorithm always identifies, for every goal a at the input abstraction level, the best (i.e., the longest) macro-action that abstracts as a, independently of where (i.e., in correspondence of what ground action) it starts in the trace.

4.2.2. An abstraction example

Figure 5 shows a trace abstraction example. The two ground actions "MRI" and "MRI with DWI" are abstracted to the macro-action "Parenchima Examination", when abstracting up to level 2 in the ontology of Figure 4. While creating this macro-action m_i, the lengths of the two delays D_1 and D_2 are summed up and accumulated in $total\_delay_i$. Similarly, the length of the interleaved action "Heparin" is accumulated in tot_inter_i, and num_inter_i is set to 1. If, for instance, $D_1 + D_2$ had exceed the delay threshold $delay\_th$, "MRI with DWI" would not have added to the macro-action started by "MRI". On the other hand, "Heparin" and "Oxygen" are abstracted to the macro-action "In-Hospital Disability Reduction", with "MRI with DWI" and

```
$M = abs\_dynProgr(partition, n(a), trace, delay\_th,$
  $n\_inter\_th, inter\_th);$
create matrix $M[n(a)][n(a)]$ initially empty;
foreach $i \leftarrow 1$ to $n(a)$ do
    create macro-action : $m_i$
    $m_i.start = i.start$
    $m_i.end = i.end$
    $total\_delay_i = 0$
    $num\_inter_i = 0$
    $total\_inter_i = 0$
    $M[i, i] = \{m_i\}$
end
foreach $i \leftarrow 1$ to $n(a)$ do /* rows */
    foreach $j \leftarrow i + 1$ to $n(a)$ do /* columns */
        $total\_delay_i = total\_delay_i + length(delay\ i - j)$
        $num\_inter_i = num\_inter_i + |interl.act.i - j|$
        $total\_inter_i = total\_inter_i + length(interl.act.i - j)$
        if (($total\_delay_i < delay\_th$) $\wedge$ ($num\_inter_i <$
          $n\_inter\_th$) $\wedge$ ($total\_inter_i < inter\_th$)) then
            $m_i.end = max(m_i.end, j.end)$
            $M[i, j] = \{m_i\}$
        end
    end
end
return $M$
```
Algorithm 1: Dynamic programming algorithm – part of the multi-level abstraction procedure

"Anti-aggregants No TPA" operating as interleaved actions, and D_2, D_3 and D_4 operating as delays. "Anti-aggregants No TPA" alone abstracts to the macro-action "Early Relapse Prevention". In the end, the macro-actions "Parenchima Examination" and "In-Hospital Disability Reduction" *overlap* [16], while the macro-action "Early Relapse Prevention" is *during* [16] the macro-action "In-Hospital Disability Reduction". This reflects the inherent parallelism of several clinical workflows, which are governed not only by clinical protocols, but also by organizational constraints, such as the availability of diagnostic instruments and personnel shifts.

4.3. Trace comparison

In our framework, we have extended a metric we described in [17], which worked on ground traces, in order to permit the comparison of abstracted traces as well.

In the current, more general approach, every trace is a sequence of actions (whether ground actions or abstracted macro-actions), each one stored with its execution starting and ending times. Therefore, an action is basically a symbol (plus the temporal information). Starting and ending times allow to get information about action durations, as well as qualitative (e.g., Allen's *before*, *overlaps*, *equals* etc. [16]) and quantitative temporal constraints (e.g., delay length, overlap length [18]) between pairs of consecutive actions/macro-actions.

The main features of the metric published in [17] are summarized below. The extensions needed to deal with abstracted traces are also discussed later in this section.

In the metric in [17], we first take into account action types, by calculating a modified edit distance which we have called **Trace Edit Distance** [17]. As the classical edit distance [19], Trace Edit Distance tests all possible combinations of editing operations that could transform one trace into the other one. However, the cost of a *substitution* is not always set to 1. Indeed, as already observed, we have organized actions in an ontology: we can therefore adopt a more **semantic** approach, and apply Palmer's distance [20], to impose that the closer two actions are in the ontology, the less penalty we introduce for substitution.

Definition 1: Palmer's Distance.
Let α and β be two actions in the ontology t, and let γ be the closest common goal of α and β. *Palmer's Distance* $dt(\alpha, \beta)$ between α and β is defined as:

$$dt(\alpha, \beta) = \frac{N_1 + N_2}{N_1 + N_2 + 2 * N_3}$$

where N_1 is the number of arcs in the path from α and γ in t, N_2 is the number of arcs in the path from β and γ, and N_3 is the number of arcs in the path from the ontology root ("Goal") and γ.

It is worth noting that other distance definitions can be relied upon if domain knowledge is available as a semantic network with different characteristics. As an example, the metric in [21] can be relied upon when dealing with an incomplete ontology, or with an ontology containing many dense sub-ontologies. Our framework is modular and easily adaptable to this end.

Trace Edit Distance $trace_{NGLD}(P, Q)$ is then calculated as the Normalized Generalized Levenshtein Distance (NGLD) [22] between two traces P and Q (interpreted as two strings of symbols). Formally, we provide the following definitions:

Definition 2: Trace Generalized Levenshtein Distance.

Let P and Q be two traces of actions, and let α and β be two actions. The Trace Generalized Levenshtein Distance $trace_{GLD}(P, Q)$ between P and Q is defined as:

$$trace_{GLD}(P, Q) = min\{\sum_{i=1}^{k} c(e_i)\}$$

where $(e_1, \ldots, e_k)$ transforms P into Q, and:

- $c(e_i) = 1$, if e_i is an action insertion or deletion;
- $c(e_i) = dt(\alpha, \beta)$, if e_i is the substitution of α (appearing in P) with β (appearing in Q), with $dt(\alpha, \beta)$ defined as in Definition 1 above.

Definition 3: Trace Edit Distance (Trace Normalized Generalized Levenshtein Distance).

Let P and Q be two traces of actions, and let $trace_{GLD}(P, Q)$ be defined as in Definition 2 above. We define Trace Edit Distance $trace_{NGLD}(P, Q)$ between P and Q as:

$$trace_{NGLD}(P, Q) = \frac{2 * trace_{GLD}(P, Q)}{|P| + |Q| + trace_{GLD}(P, Q)}$$

where $|P|$ and $|Q|$ are the lengths (i.e., the number of actions) of P and Q respectively.

Trace Edit Distance then takes the combination of editing operations associated to the minimal cost. Such a choice corresponds to a specific alignment of the two traces (*optimal alignment* henceforth), in which each action in one trace has been matched to an action in the other trace–or to a gap.

Given the optimal alignment, we can then take into account temporal information. In particular, we compare the durations of aligned actions by means of a metric we called **Interval Distance** [17]. Interval distance calculates the normalized difference between the length of two intervals (representing action durations in this case).

Moreover, we take into account the temporal constraints between two pairs of subsequent aligned actions on the traces being compared (e.g., actions A and B in trace P; the aligned actions A' and B' in trace Q). We quantify the distance between their qualitative constraints (e.g., A and B overlap in trace P; A' meets B' in trace Q), by resorting to a metric known as **Neighbors-graph Distance** [17]. If Neighbors-graph Distance is 0, because the two pairs of actions share the same qualitative constraint (e.g., A and B overlap in trace P; A' and B' also overlap in trace Q), we compare quantitative constraints by properly applying Interval Distance again (e.g., by calculating Interval Distance between the two overlap lengths).

In the metric in [17], these three contributions (i.e., Trace Edit Distance, Interval Distance between durations, Neighbors-graph Distance or Interval Distance between pairs of actions) are finally put in a linear combination with non-negative weights.

When working on macro-actions, however, the metric in [17] needs to be extended. Indeed, two otherwise identical abstracted traces, for simplicity composed by just one macro-action each, may differ only because the macro-action in the first trace includes some delay or interleaved action, while the macro-action in the second trace does not (it is a pure direct sequence of ground actions sharing the same goal). Our extended metric allows to penalize this difference, by considering, given the optimal macro-actions alignment, two additional contributions:

- a penalty due to the different length of the delays incorporated into the two aligned macro-actions;
- a penalty due to the different number, length and type of interleaved actions in the two aligned macro-actions being compared.

The extended metric includes in the linear combination these two penalties as well.

Of course, if the user is not interested in highlighting these aspects, s/he can set to 0 the weight of these penalty contributions.

Formally, we provide four definitions. Definition 3 and Definition 4 were published in [14], but are reported here in order to make the section self-contained. Definition 5 and Definition 6, on the other hand, represent a new contribution of this paper.

Delay penalty is defined straightforwardly as follows:

Definition 3: Delay Penalty.

Let A and B be two macro-actions, that have been matched in the optimal alignment. Let $delay_A = \sum_{i=1}^{k} length(i)$ be the sum of the lengths of all the k delays that have been incorporated into A in the abstraction phase (and let $delay_B$ be analogously defined). Let $maxdelay$ be the maximum, over all the abstracted traces, of the sum of the lengths of the delays incorporated in an abstracted trace. The Delay Penalty $delay_p(A, B)$ between A and B is defined as:

$$delay_p(A, B) = \frac{|delay_A - delay_B|}{maxdelay}$$

As for interleaved actions penalty, we have foreseen different possible definitions (and additional ones may be introduced, on the basis of domain needs).

The first definition operates analogously to delay penalty, by summing up the lengths of all interleaved actions that have been incorporated within a single macro-action in the abstraction phase, without distinguishing among the different types of such interleaved actions.

Definition 4: Interleaving Length Penalty.
Let A and B be two macro-actions, that have been matched in the optimal alignment. Let $inter_A = \sum_{i=1}^{k} length(i)$ be the sum of the lengths of all the k interleaved actions that have been incorporated into A in the abstraction phase (and let $inter_B$ be analogously defined). Let $maxinter$ be the maximum, over all the abstracted traces, of the sum of the lengths of the interleaved actions incorporated in an abstracted trace. The Interleaving Length Penalty $interL_p(A, B)$ between A and B is defined as:

$$interL_p(A, B) = \frac{|inter_A - inter_B|}{maxinter}$$

The next definition compares the number of interleaved actions in the two macro-actions matched by the optimal alignment.

Definition 5: Interleaving Number Penalty.

Let A and B be two macro-actions, that have been matched in the optimal alignment. Let $number_A$ be the number of the interleaved actions that have been incorporated into A in the abstraction phase (and let $number_B$ be analogously defined). Let $maxnumber$ be the maximum, over all the abstracted traces, of the number of the interleaved actions incorporated in an abstracted trace. The Interleaving Number Penalty $interN_p(A, B)$ between A and B is defined as:

$$interN_p(A, B) = \frac{|number_A - number_B|}{maxnumber}$$

The last definition we present is the most complex. In this case, we want to sum up the lengths of all the interleaved actions as in Definition 4, but every length will be weighted by Palmer's distance [20] (see Definition 1) between the interleaved action itself and the goal that labels the macro-action incorporating it. In this way, if a very unrelated action (e.g., counseling about smoke habits) is interleaved in a macro-action (e.g., pathogenetic mechanism identification), it will increase the penalty value more significantly with respect to a more similar (i.e., closer) action in the ontology (e.g., a diagnostic action).

Definition 6: Interleaving Weighted Penalty.

Let A and B be two macro-actions, that have been matched in the optimal alignment. Let $weighted_A = \sum_{i=1}^{k} length(i) * dt(i, A)$ be the sum of the lengths of all the k interleaved actions that have been incorporated into A in the abstraction phase, each weighted by Palmer's distance between the interleaved action itself and A. Let $weighted_B$ be analogously defined. Let $maxinter$ be the maximum, over all the abstracted traces, of the sum of the lengths of the interleaved actions incorporated in an abstracted trace. The Interleaving Weighted Penalty $interW_p(A,B)$ between A and B is defined as:

$$interW_p(A,B) = \frac{|weighted_A - weighted_B|}{maxinter}$$

It is worth noting that our metric, given its capability to manage both quantitative and qualitative temporal constraints, enables to properly deal with temporal information at all abstraction levels.

By allowing the treatment of abstraction penalties (in a very flexible way, thanks to the different definitions we provide), and the management of temporal information, the metric is therefore able to address all the issues we cited in Section 4.2.

4.4. Process discovery

In our approach, we are resorting to the well-known process mining tool ProM, extensively described in [23]. 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 experimental work described in this paper, we have exploited ProM's Heuristic Miner [24]. Heuristic Miner [24] is a plug-in for process discovery, able to mine process models from event logs. Heuristic Miner receives as 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, 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 algorithms available in ProM 6 is foreseen in our future work. Moreover, the interface of our framework to ProM will allow us to test additional analysis plug-ins in the future.

5. Results

In this section, we describe two experimental works we have conducted, in the application domain of stroke care. In the first one (see Section 5.1), we have studied the impact of multi-level abstraction on trace comparison; in particular, we have designed a set of clustering experiments, to verify whether it is possible to highlight correct behaviors and anomalies with respect to the latest clinical practice guidelines for stroke management, abstracting from details (such as, e.g., local resource constraints or local medical practice), that are irrelevant to the verification of medical appropriateness of a macro-action.

In the second work (see Section 5.2), we wished to verify whether our support to process discovery, and specifically the capability of abstracting the traces on the basis of their semantic goal, allows to obtain clearer medical process models, where unnecessary details are hidden, but key behaviors are clear.

Parameter setting. In the experiments, thresholds were common to all traces in the log, and set as follows: $delay\_th = 300$ minutes, $n\_inter\_th = 3$, $inter\_th = 300$ minutes. This choice was set by our medical co-author, on the basis of medical knowledge and experience. Interestingly, we also made tests with different thresholds (making changes of up to 10%). The impact of these changes on process discovery (in terms of fitness) will be presented in Section 5.2.

The metric we adopted for trace comparison is the one we described in Section 4.3, where the linear combination weights were all equal and their sum was 1, and we resorted to Definition 6 for interleaving penalty.

Experimental dataset. The available event log was composed of more than 15000 traces (each trace representing a different patient), collected at the 40 Stroke Unit Network (SUN) collaborating centers of the Lombardia region, Italy. Our medical co-author belongs to one of the SUN stroke units. Thus, she has a very deep insight into the registry data. Traces were composed of 13 actions on average, with a median of 15.

Machine characteristics. Experiments were run on a machine equipped with an Intel(R) Xeon(R) CPU E5-2640v2, CPU @ 2GHz, 4GB RAM.

Results are provided in the following.

5.1. Trace comparison

As a first experimental work, we have analyzed the impact of our abstraction mechanism on trace comparison, and more precisely on the quality of trace clustering.

In our study, we considered the traces of every single Stroke Unit (SU), and compared clustering results on ground traces with respect to those on abstracted traces.

Specifically, we resorted to a hierarchical clustering technique, known as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [25]. UPGMA is typically applied in bioinformatics, where sequences of symbols (similar to our traces) have to be compared. The algorithm operates in a bottom-up fashion. At each step, the nearest two clusters are combined into a higher-level cluster. The distance between any two clusters A and B is taken to be the average of all distances between pairs of objects “x” in A and “y” in B, that is, the mean distance between elements of each cluster. After the creation of a new cluster, UPGMA properly updates a pairwise distance matrix it maintains. UPGMA also allows to build the phylogenetic tree (the hierarchy) of the obtained clusters.

In our experiments, the hypothesis we wished to test was the following: “the application of the abstraction mechanism as a pre-processing step for UPGMA clustering allows to obtain more *homogeneous* and compact clusters (i.e., able to aggregate closer examples); however, outliers are still clearly identifiable, and isolated in the cluster hierarchy”.

Homogeneity is a widely used measure of the quality of the output of a clustering method (see e.g., [26, 27, 28, 29]). The average of the homogeneity H of the individual clusters can be calculated on (some of) the clusters obtained through the method at hand, in order to assess clustering quality. Average cluster homogeneity allows one to compare the output of different clustering techniques on the same dataset, or the output obtained by differently setting up the same clustering technique, as we did by pre-processing traces by means of the abstraction mechanism.

We computed the average of cluster homogeneity values level by level in the hierarchies. In order to avoid biases, to calculate homogeneity we resorted to the classical normalized Levensthein's edit distance [19], with no use of semantic, temporal or abstraction information (indeed, if homogeneity is calculated resorting to the metric defined in this paper, it obviously increases when working on abstracted traces, since Palmer's distance [20] decreases when operating at higher levels of the hierarchy).

As a first example, we report on the results of applying UPGMA to the 200 traces of a specific SU.

Figure 6 shows a comparison of the average homogeneity values, computed level by level in the cluster hierarchies, operating (i) on ground traces, (ii) on traces abstracted at level 2 in the ontology, and (iii) on traces abstracted at level 1 in the ontology. As it can be observed, the more we abstract, the more homogeneity increases. Indeed, homogeneity on abstracted traces is always higher then the one calculated on ground traces; moreover, homogeneity when abstracting at level 1 (i.e., up to the most general medical goals) is never lower than the one calculated at level 2.

It is also interesting to study the management of outliers, i.e., in our application domain, traces that could correspond to the treatment of atypical patients, or to medical errors. These traces record rather uncommon actions, and/or present uncommon temporal constraints among their actions. For instance, trace 73 is very peculiar: it describes the management of a patient suffering from several inter-current complications (diabetes, hypertension),

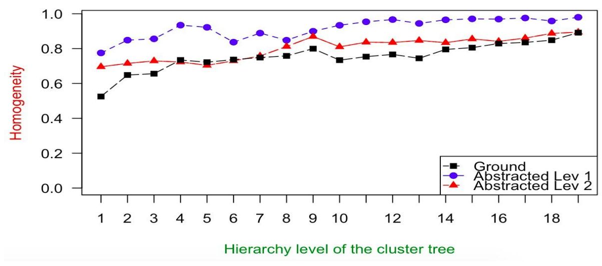

Figure 6: Comparison between average homogeneity values, computed level by level in the three cluster hierarchies obtained by UPGMA working at different levels of abstraction, in the first SU

who required many extra-tests and many specialist counseling sessions, interleaved to more standard actions.

Ideally, these anomalous traces should remain isolated as a singleton cluster for many UPGMA iterations, and be merged to other nodes in the hierarchy as late as possible, i.e., close to the root (level 0). When working on ground traces, 9 outliers of the example SU were merged very late to the hierarchy, as expected (between level 1 and level 7, see Figure 7). In particular, trace 73 was merged only at level 1. Very interestingly, this capability of isolating outliers was preserved when working on abstracted traces. Indeed, as it can be seen in the figure, despite some differences, all 9 outliers were early isolated, in both the abstraction experiments. Trace 73, in particular, was the latest trace to be merged to the cluster trees at both abstraction levels.

Very similar considerations can be drawn when observing the results on another SU, that we report in Figures 8 and 9. In this case, homogeneity on abstracted traces was always higher then the one calculated on ground traces as well (see Figure 8); moreover, all 11 outliers were always isolated between level 1 and level 8 (see Figure 9).

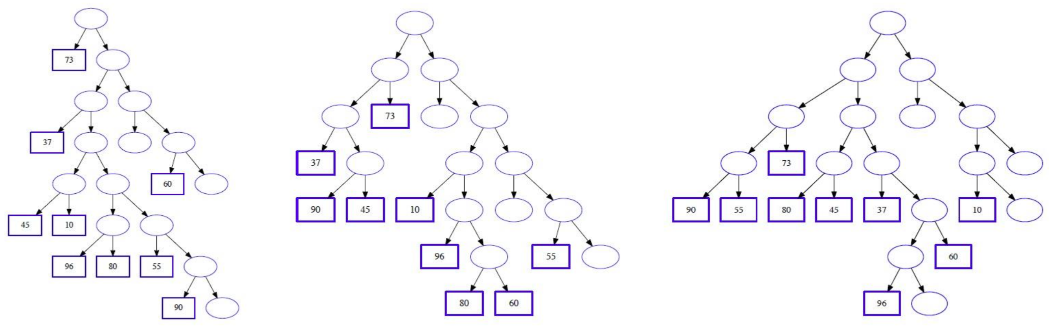

Figure 7: Identification of outliers (in rectangles) in cluster hierarchies in the first SU. From left to right: part of the cluster hierarchy built on ground traces, on traces abstracted at level 2, and on traces abstracted at level 1

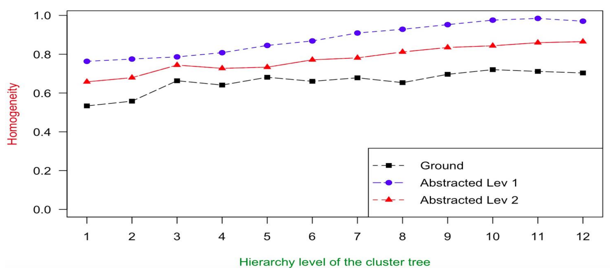

Figure 8: Comparison between average homogeneity values, computed level by level in the three cluster hierarchies obtained by UPGMA working at different levels of abstraction, in the second SU

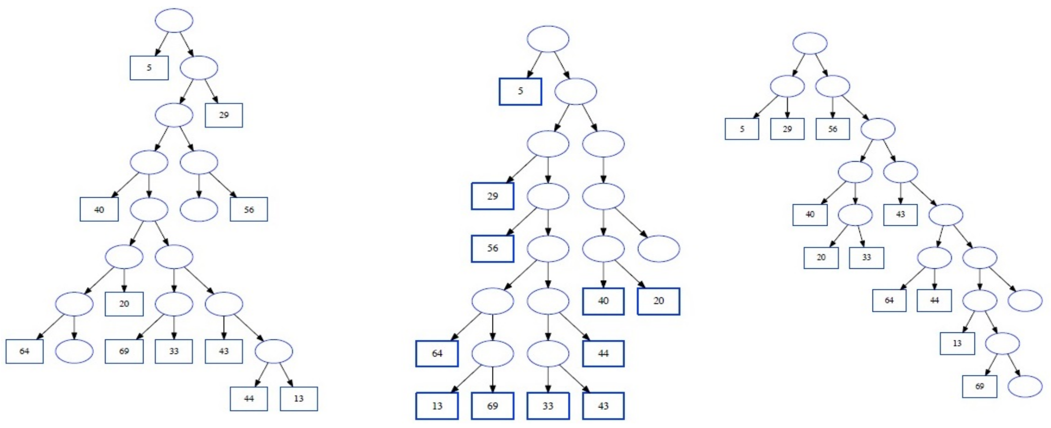

Figure 9: Identification of outliers (in rectangles) in cluster hierarchies in the second SU. From left to right: part of the cluster hierarchy built on ground traces, on traces abstracted at level 2, and on traces abstracted at level 1

In conclusion, our hypothesis was verifed by the experiments (even considering different subsets of the Stroke Registry data), since the application of the abstraction mechanism allowed to obtain more homogeneous clusters, still clearly isolating outlying traces.

5.2. Process discovery

As a second experimental work, we have tested whether our capability to abstract the event log traces on the basis of their semantic goal 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).

In particular, we mined the process models of all the available SUs, both operating on ground traces, and on traces abstracted at different levels, with respect to the ontology in Figure 4.

As an example, Figure 10 compares the process models of two different SUs (SU1 and SU2), mined by resorting to Heuristic Miner [24], operating on ground traces. Figure 11 compares the process models of the same SUs as Figure 10, again mined by resorting to Heuristic Miner, but operating on traces abstracted at level 1 of the ontology in Figure 4, i.e., referring to the most general medical goals. Figure 12, instead, compares the process models of the same SUs operating on traces abstracted at level 2 of the ontology.

Generally speaking, a visual inspection of the two graphs in Figure 10

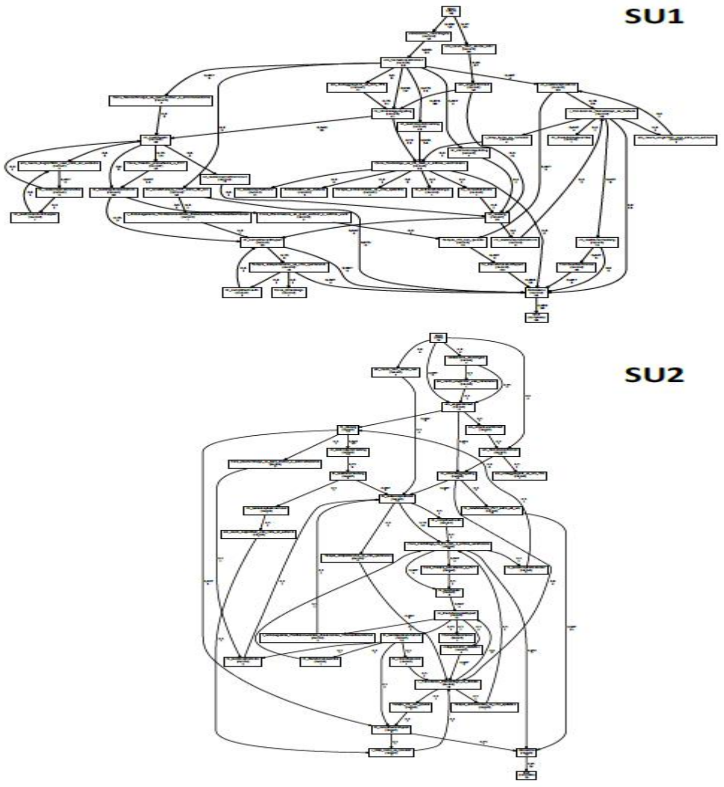

Figure 10: Comparison between two process models, mined by resorting to Heuristic Miner, operating on ground traces. The figure is not intended to be readable, but only to give an idea of how complex the models can be

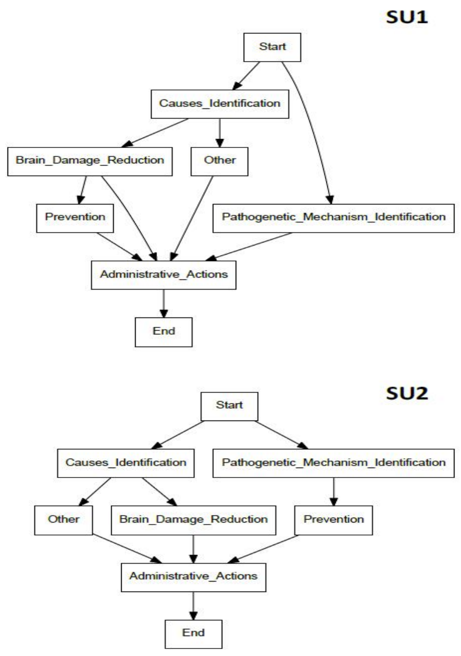

Figure 11: Comparison between the two process models of the same SUs as Figure 10, mined operating on traces abstracted at level 1

is very difficult. Indeed, these two ground processes are "spaghetti-like" [1], and the extremely large number of nodes and edges makes it hard to identify commonalities in the two models. The abstracted models in Figures 11 and 12, 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 11 show exactly the same macro-actions, which indeed represent the main goals/steps of the treatment of a typical stroke patient (see also Figure 4). Only minor variations in task parallelization (e.g., "Prevention" is in parallel to "Pathogenetic Mechanism Identification" in SU1, while the two tasks are executed in sequence in SU2) can be observed, due to (mostly irrelevant) differences in local best practices.

Some more significant differences can be observed in Figure 12[5] where, in addition to control flow changes, the "Recanalization Therapy" action can be observed in SU2 (highlighted in bold). This is a very reasonable outcome, since SU2 is a well equipped SU, where various kinds of patients, including atypical ones, can be managed, thanks to the availability of different skills and instrumental resources. On the other hand at SU1 some very specific human knowledge and technical resources are missing. As a consequence, the invasive and complex recanalization therapy is not performed here.

We also report on the process discovery results on a third SU, which is a more generalist and less equipped SU both with respect to SU1, and with respect to SU2. In this case, as it can be observed in Figure 13, the "Recanalization Therapy" is still missing; moreover, the "Intra-cranial Vessel Inspection", which requires some advanced skills and proper resources as well, is missing too.

Finally, Table 1 reports our results on the calculation of *fitness* [1] on the process models mined for our 40 SUs, at different levels of abstraction. Fitness evaluates whether a process model is able to reproduce all execution sequences that are in the event log. If the log can be replayed correctly, fitness evaluates to 1. In the worst case, it evaluates to 0[6].

[5]Note that some medical goals do not conceptually specialize to more specific subgoals; for instance "Administrative Actions" at level 1 specializes to a single subgoal which is still called "Administrative Actions" at level 2, since no further discrimination is required.

[6]More formally, fitness of a log L on a model N represented as a Petri Net is [1]:
$fitness(L,N) = \frac{1}{2}(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times c_{N,\sigma}}) + \frac{1}{2}(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_{N,\sigma}}{\sum_{\sigma \in L} L(\sigma) \times p_{N,\sigma}})$
where $p_{N,\sigma}$ denotes the number of produced tokens when replaying a trace σ on N; similarly, $c_{N,\sigma}$, $m_{N,\sigma}$, $r_{N,\sigma}$ are the number of consumed, missing and remaining tokens (respectively) when replaying σ on N. $\sum_{\sigma \in L} L(\sigma) \times m_{N,\sigma}$ is total number of missing tokens when replaying the entire event log, because $L(\sigma)$ is the frequency of trace σ and $m_{N,\sigma}$ is

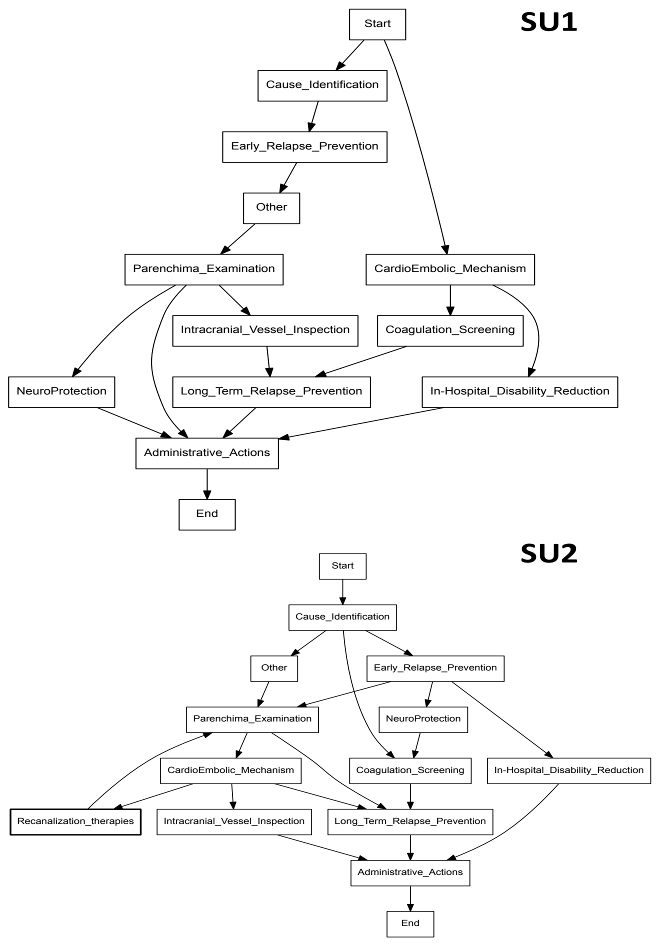

Figure 12: Comparison between the two process models of the same SUs as Figure 10, mined operating on traces abstracted at level 2

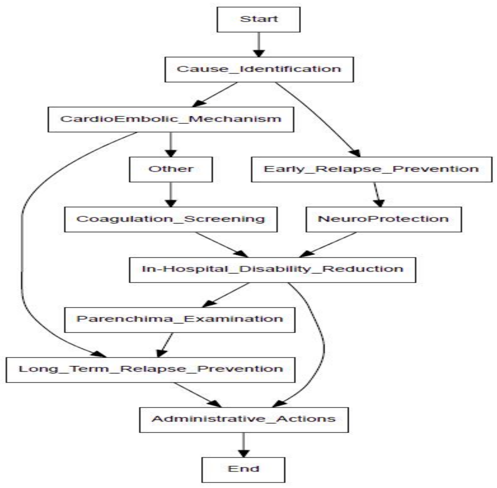

Figure 13: Process model of a third SU, less equipped with respect to SU1 and SU2, mined operating on traces abstracted at level 2

Table 1: Average fitness values calculated on the mined process models, when operating at different levels of abstraction

Ground	**Abs. level 2**	**Abs. level 1**
0.54	0.78	0.89

Fitness calculation is available in ProM.

As it can be observed from the table, the more the traces are abstracted, the more the average fitness value increases in the corresponding models obtained by the mining algorithm. In particular, as regards SU1 presented in detail above, fitness passed from 0.47396724 (process mined on ground traces), to 0.82176177 (process mined on traces abstracted at level 2), to 0.9062092 (process mined on traces abstracted at level 1). Similarly, for SU2 fitness passed from 0.4985412 (process mined on ground traces), to 0.6786843

the number of missing tokens for a single instance of σ.

(process mined on traces abstracted at level 2), to 0.8952279 (process mined on traces abstracted at level 1).

In conclusion, our abstraction mechanism, while hiding irrelevant details, allows to still appreciate relevant differences between models, such as, e.g., the presence/absence of relevant actions, as in the case of recanalization. Moreover, very interestingly, abstraction proves to be a means to significantly increase the quality of the mined models, measured in terms of fitness, a well known and largely adopted quantitative indicator.

While we do not have a direct measure of the quality of abstraction (there is no specific indicator of how good the abstracted event log is), we could test the impact of abstraction threshold change (see Algorithm 1) on the quality of process discovery, by measuring, once again, fitness values.

Table 2 reports on the effects of abstraction threshold variation. The first two columns show the fitness values of three different SUs, measured when the process models are mined working on abstracted traces (at level 1 and 2 respectively), having set the abstraction thresholds to the values indicated by the domain expert. Columns 3 and 4 show the corresponding fitness values, having altered the thresholds of +10%. As it can be observed, threshold changes had a minimal impact on fitness values: they often did not change at all; in other cases, the change involved the third or fourth decimal place. Very similar results were obtained after a change of -10%.

For the sake of completeness, we also provide some results about computational performance of the various steps of the framework. Abstraction on a log of 481 traces, as an example, took 1230.346 seconds. Process discovery on ground traces took 2.062 seconds, while it took only 0.489 seconds

Table 2: Impact of abstraction threshold change on fitness values, calculated on three mined process models, operating at different levels of abstraction

	Expert lev.1	**Expert lev.2**	**Change lev.1**	**Change lev.2**
SUa	0.9044	0.7821	0.9043	0.7820
SUb	0.9552	0.8104	0.9532	0.8104
SUc	0.8759	0.7268	0.8759	0.7268

when working on traces abstracted at level 1. Clustering on ground traces took 633.209 seconds, while it took 540.087 seconds when operating on traces abstracted at level 1.

So, abstraction is the most costly step, but it is an off-line procedure: as such, it is not time-critical. Moreover, the other steps become more efficient when run on abstracted traces.

6. Related works and discussion

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 [30], 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 [31] extends the one in [30], 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). The work in [32] develops in a similar context, and extends OLAP tools with semantics (exploiting ontologies rather than (flat) taxonomies). Hepp et al. [33] 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 [34].

Semantic business process management is further developed in the SUPER project [35], within which several ontologies are created, such as the process mining ontology and the event ontology [36]; these ontologies define core terminologies of business process management, usable by machines for task automation. However, the authors do 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 and analysis in [7]. This paper also shows how to use these building blocks to extend ProM's LTL Checker [37] to perform semantic auditing of logs. The work in [38] 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 corrective actions. Detection, diagnosis and resolution present interesting challenges that, on the authors' opinion, can strongly benefit from knowledge-based techniques. In [38, 8] 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 also addressed.

In [8] 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. Neither the management of interleaved actions or delays, nor the correct identification of temporal constraints generated when aggregating different macro-actions are addressed. Moreover, most of the papers cited above (including [33, 39, 38, 7, 8]) present theoretical frameworks, and not yet a detailed technical architecture nor a concrete implementation of all their ideas.

In [40] the authors characterize the manifestation of commonly used process model constructs in the event log and adopt pattern definitions that capture these manifestations, and propose a means to form abstractions over these patterns. In particular, the approach identifies loops in traces, and replaces the repeated occurrences of the manifestation of the loop by an abstracted entity that encodes the notion of a loop. It also identifies common functionalities in the traces and replaces them with abstract entities. The work in [41] operates similarly, and abstracts composite actions and loops from sets of multiple/repeated simpler actions. These works, however, do not make use of semantic information.

A more recent work [42] introduces a methodology that combines domain and company-specific ontologies and databases to obtain multiple levels of abstraction for process mining and analysis. In this paper data in databases become instances of concepts at the bottom level of the 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 not find any clear description of the abstraction algorithm, and neither the management of interleaved actions or delays, nor the correct identification of temporal constraints generated when aggregating different macro-actions were properly addressed.

Another interesting approach to abstraction in process models is the one in [6]. The authors propose abstraction to generate more readable high-level views on business process models. They are able to discover sets of related actions, where each set corresponds to a coarse-grained task in an abstract process model. Specifically, abstraction resorts to a clustering technique, where action properties (such as, e.g., roles and resources) are exploited to aggregate the different actions into the common task. The authors adopt the enhanced Topic Vector Space Model to reflect the semantic relations between action property values: in this way, the distance between two different, but related values, can be lower that 1. Differently from our approach, however, the abstraction solution described in [6] is not applied to traces - and therefore cannot be adopted for trace comparison. Moreover, it requires that all action properties are available and logged - which, unfortunately, is often not the case, for instance in medicine, where logging may be incomplete in practice. Moreover, clustering does not take into account temporal relations between actions, in the sense that it may also aggregate actions executed at temporally distant phases of the model control flow; on the other hand, our approach, by operating on traces, which log the temporal sequence of action executions and their temporal constraints, strongly relies on temporal information, maintains it, and allows to exploit it in further analyses, such as abstracted trace comparison. Thus, the work in [6] adopts a significantly different technique to process model abstraction with respect to our proposal; nonetheless, it is certainly a relevant related work, and it would be interesting to compare abstraction results obtained through that method to our medical logs, in order to evaluate pros and cons of the two methodologies.

Some approaches to process mining and workflow analysis exist that were specifically designed to medical applications. While most of them rely on classical business process management literature for process representation and discovery [43, 44, 45] others adopt different solutions, such as the use of probabilistic models [46, 47] or sequential pattern mining and statistical techniques [48]. Among the classical approaches, it is worth mentioning [45]. This work adopts process mining techniques to learn surgical process models both from manually recorded traces of executed steps, and from sensor data. Interestingly, sensor data need abstraction before being employed. Unfortunately, the abstraction technique is not described in the paper. Considering probabilistic approaches, on the other hand, in [47] the authors introduce pMinerR, a library focused on dealing with process discovery and conformance checking in the medical domain, embedded into R, one of the most common software environment for data analysis. pMineR exploits some aspects taken from the computer interpretable clinical guidelines field, in particular in terms of human-readability. To this end, it exploits Markov Models for process model representation, which are usually well-known and easy to understand for medical user.

The work in [48], instead, combines sequential pattern mining with statistical temporal analysis to characterize the transitions between events in patient workflows. Pattern mining produces a set of sequences of events/actions that depict the services delivered to patients; such sequences are then enriched using both temporal and clinical information. Using this information it is possible to better characterize the groups of patients who were cared according to different workflows, and subsequently better assess the clinical relevance of the extracted patterns. The contribution in [48] is relevant for comparison to our work, as it addresses the topic of abstraction, at least to some extent. In particular, where there exists a trace with two subsequent identical actions, the user may consider to aggregate them into a single longer action that has the same starting time as the first occurrence and the same ending time as the second occurrence of the action itself. As shown by this example, however, the abstraction mechanism is syntactic and rather straightforward, with respect to the complexity of our semantic approach, which therefore appears to be an innovative contribution for the medical field.

Always referring specifically to medical applications, the work in [9] proposes an approach, based on semantic process mining, to verify the compliance of a Computer Interpretable Guideline with medical recommendations. In this case, however, semantic process mining refers to conformance checking rather than to process discovery (as it is also the case in [7]). This work is thus only loosely related to our contribution.

As regards trace comparison, as already observed, in this paper we have extended a metric we published in [17], able to exploit domain knowledge in action comparison, and to manage all types of temporal constraints. Other metrics for trace comparison have been proposed in the literature. In particular, [49] combines a contribution related to action similarity, and a contribution related to delays between actions. As regards the temporal component, it relies on an interval distance definition which is quite similar to ours. Differently from what we do, however, no search for the optimal action alignment is performed. The distance function in [49] does not exploit action duration, and does not rely on semantic information about actions, as we do. Finally, it does not deal with different types of qualitative temporal constraints. Another interesting contribution is [50], which addresses the problem of defining a similarity measure able to treat temporal information, and is specifically designed for clinical workflow traces. Interestingly, the authors consider qualitative temporal constraints between matched pairs of actions, resorting to the Neighbors-graph Distance, as we do. However, in [50] the alignment problem is strongly simplified, as they only match actions with the same name. In this sense, our approach is also much more semantically oriented. Several metrics for comparing process models, instead of traces, also exist. Most of them are based on proper extensions of the edit distance as well and, in some cases, allow for a semantic comparison among model actions (see, e.g., [51, 52]). However, given the very different structure of a process model (which is a graph) with respect to a trace, these works are only loosely related to our contribution.

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

- many approaches, presenting very interesting and sometimes ambitious ideas, just provide theoretical frameworks, while concrete implementations of algorithms and complete architectures of systems are often missing;

- 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 we presented in Section 4.2 (related to the management of interleaved actions or delays, and to the correct identification of temporal constraints generated when aggregating different macro-actions) do not emerge;
- as regards trace comparison, to the best of our knowledge, our previously published metric [17], enhanced to deal with abstracted traces, still represents one of the most complete contributions to properly account for both non temporal and temporal information, and to perform a semantic comparison between actions.

7. Conclusions

In this paper, we have presented a framework for semantic multi-level abstraction of event log traces. In our architecture, abstracted traces are then provided as an input to different analysis techniques – namely, trace comparison and process discovery in the current implementation[7]. The most significant and original methodological contributions of our work consist in:

1. having defined a proper ***mechanism for abstracting event log traces***, able to manage non trivial situations (originating from the

[7]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.

treatment of interleaved actions or delays between two actions sharing the same goal);

2. having provided ***a trace comparison facility***, which resorts to a ***similarity metric*** (extending the metric we presented in [17]), 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 [23]; indeed, the overall integration of our approach within ProM is foreseen in our future work.

Experimental results on trace comparison (and more specifically on trace clustering) in the field of stroke management have shown that it is easier to identify common behaviors in abstracted traces, with respect to ground traces: in fact, cluster homogeneity, when operating on abstracted traces, reaches higher values. At the same time, outliers (i.e., anomalies and incorrect behaviors) are still clearly visible in abstracted traces as well (and clearly detected by the clustering method we used). Process discovery experiments have proved that the capability of abstracting the event log traces on the basis of their semantic goal allows to mine clearer process models, where unnecessary details are hidden, but key behaviors are clear. In the future, we plan to conduct further experiments, e.g., by 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 [53]. Moreover, we plan to compare abstraction results obtained through our approach to the ones provided by the approach in [6], in order to evaluate pros and cons of the two techniques on real medical logs.

From a methodological viewpoint, we plan to extend our approach, in particular as regards the expressiveness of the rule base. Specifically, we would like to introduce temporal constraints in rule antecedents, to better define the context of execution of an action, and thus to find its correct goal in the case of multiple "aimsTo" relations. For instance, the execution of a specific action before the action to be abstracted could trigger a rule only if it took place within 24 hours. Moreover, rules could be made "fuzzy": referring to the previous example, a time delay of 25 hours may be accepted as well. The use of such fuzzy thresholds in the rules would make knowledge acquisition simpler, and would limit the number of rules to be acquired, thus mitigating the risk of defining a too big rule base, that could impact the computational performance of the system.

Moreover, as observed, the actions exploited to implement medical goals are being mapped to SNOMED concepts. Mapping is not complete yet. Currently, we basically associate SNOMED codes to the actions, and, through the codes, we get a pointer to one of the largest medical knowledge vocabularies. As a next step, we will check the semantic coherence between our ontology and SNOMED axioms.

On the other hand, since not all the medical goals needed in our application are reported in SNOMED (and the "has-intent" SNOMED relation covers only partially our needs), we could not map all terms at higher ontology levels to SNOMED codes, and we often had to define the goal/subgoal and "aimsTo" relations from scratch. Possible analyses for a tighter integration will be conducted in the future.

Moreover, we will investigate if the various attributes reported in SNOMED (like, e.g., the type of substances administered to a patient, or the morphological part of the body involved) may be used to abstract actions along different dimensions (other than the goal), which could be of interest in some new medical project.

Finally, we wish to extensively test the overall approach in different application domains.

As regards the two last points, it is however worth noting that our framework, and in particular the abstraction mechanism, are general, but strongly knowledge-based. Therefore:

- our work can be applied to different domains (i.e., other than stroke), but this change requires the preliminary acquisition of the corresponding knowledge sources (i.e., ontology and rules) in the new domain;
- abstraction can be made according to a different dimension (e.g., consumed resources, or morphological part of the body involved), but this change requires the preliminary acquisition of the corresponding knowledge sources according to the new perspective (e.g., an ontology assessing what resources are consumed by what actions - if not available through SNOMED).

Therefore, these adaptations might be quite time consuming. Nevertheless, we believe that such improvements will make our framework more complete and flexible, and that these further tests will testify its usefulness in practice.

References

References

[1] W. V. der Aalst, Process Mining. Data Science in Action, Springer, 2016.

[2] W. V. der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, A. Weijters, Workflow mining: a survey of issues and approaches, Data and Knowledge Engineering 47 (2003) 237–267.

[3] W. van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, IEEE Trans. Knowl. Data Eng. 16 (9) (2004) 1128–1142.

[4] E. R. Aguilar, F. Ruiz, F. García, M. Piattini, Evaluation measures for business process models, in: H. Haddad (Ed.), Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006, ACM, 2006, pp. 1567–1568.

[5] I. T. P. Vanderfeesten, H. A. Reijers, J. Mendling, W. M. P. van der Aalst, J. S. Cardoso, On a quest for good process models: The cross-connectivity metric, in: Z. Bellahsene, M. Léonard (Eds.), Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008, Montpellier, France, June 16-20, 2008, Proceedings, Vol. 5074 of Lecture Notes in Computer Science, Springer, 2008, pp. 480–494.

[6] S. Smirnov, H. A. Reijers, M. Weske, From fine-grained to abstract process models: A semantic approach, Inf. Syst. 37 (8) (2012) 784–797.

[7] A. K. A. de Medeiros, W. M. P. van der Aalst, C. Pedrinaci, Semantic process mining tools: Core building blocks, in: W. Golden, T. Acton, K. Conboy, H. van der Heijden, V. K. Tuunainen (Eds.), 16th European Conference on Information Systems, ECIS 2008, Galway, Ireland, 2008, 2008, pp. 1953–1964.

[8] A. K. A. de Medeiros, W. M. P. van der Aalst, Process mining towards semantics, in: T. S. Dillon, E. Chang, R. Meersman, K. P. Sycara (Eds.), Advances in Web Semantics I - Ontologies, Web Services and Applied Semantic Web, Vol. 4891 of Lecture Notes in Computer Science, Springer, 2009, pp. 35–80.

[9] M. A. Grando, M. H. Schonenberg, W. M. P. van der Aalst, Semantic process mining for the verification of medical recommendations, in: V. Traver, A. L. N. Fred, J. Filipe, H. Gamboa (Eds.), HEALTHINF 2011 - Proceedings of the International Conference on Health Informatics, Rome, Italy, 26-29 January, 2011, SciTePress, 2011, pp. 5–16.

[10] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, W. M. P. van der Aalst, XES, XESame, and ProM 6, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 60–75.

[11] B. van Dongen, W. van der Aalst, A meta model for process mining data, in: M. Missikoff, A. D. Nicola (Eds.), EMOI - INTEROP'05, Enterprise Modelling and Ontologies for Interoperability, Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Co-located with CAiSE'05 Conference, Porto (Portu-

gal), 13th-14th June 2005, Vol. 160 of CEUR Workshop Proceedings, CEUR-WS.org, 2005.

[12] G. Lanzola, E. Parimbelli, A. Cavallini, S. Quaglini, Data quality and completeness in a web stroke registry as the basis for data and process mining, J Healthc Eng 5 (2) 163–184.

[13] S. Montani, M. Striani, S. Quaglini, A. Cavallini, G. Leonardi, Knowledge-based trace abstraction for semantic process mining, in: A. ten Teije, C. Popow, J. H. Holmes, L. Sacchi (Eds.), Artificial Intelligence in Medicine - 16th Conference on Artificial Intelligence in Medicine, AIME 2017, Vienna, Austria, June 21-24, 2017, Proceedings, Vol. 10259 of Lecture Notes in Computer Science, Springer, 2017, pp. 267–271.

[14] S. Montani, M. Striani, S. Quaglini, A. Cavallini, G. Leonardi, Semantic trace comparison at multiple levels of abstraction, in: D. W. Aha, J. Lieber (Eds.), Case-Based Reasoning Research and Development - 25th International Conference, ICCBR 2017, Trondheim, Norway, June 26-28, 2017, Proceedings, Vol. 10339 of Lecture Notes in Computer Science, Springer, 2017, pp. 212–226.

[15] M. N. D. M. M. Salatino, E. Aliverti, Mastering JBoss Drools 6 for Developers, Packt Publishing, 2016.

[16] J. Allen, Towards a general theory of action and time, Artificial Intelligence 23 (1984) 123–154.

[17] S. Montani, G. Leonardi, Retrieval and clustering for supporting business process adjustment and analysis, Information Systems 40 (2014) 128–141.

[18] A. Lanz, B. Weber, M. Reichert, Workflow time patterns for process-aware information systems, in: Proc. BMMDS/EMMSAD, 2010, pp. 94–107.

[19] A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, Soviet Physics Doklady 10 (1966) 707–710.

[20] M. Palmer, Z. Wu, Verb Semantics for English-Chinese Translation, Machine Translation 10 (1995) 59–92.

[21] E. Chiabrando, S. Likavec, I. Lombardi, C. Picardi, D. Theseider-Dupré, Semantic similarity in heterogeneous ontologies, in: P. D. Bra, K. Grønbæk (Eds.), HT'11, Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, Eindhoven, The Netherlands, June 6-9, 2011, ACM, 2011, pp. 153–160.

[22] L. Yujian, L. Bo, A normalized levenshtein distance metric, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 1091–1085.

[23] B. van Dongen, A. A. De Medeiros, H. Verbeek, A. Weijters, W. V. der Aalst, The proM framework: a new era in process mining tool support, in: G. Ciardo, P. Darondeau (Eds.), Knowledge Mangement and its Integrative Elements, Springer, Berlin, 2005, pp. 444–454.

[24] A. Weijters, W. V. der Aalst, A. A. de Medeiros, Process Mining with the Heuristic Miner Algorithm, WP 166, Eindhoven University of Technology, Eindhoven, 2006.

[25] R. Sokal, C. Michener, A statistical method for evaluating systematic relationships, University of Kansas Science Bulletin 38 (1958) 1409–1438.

[26] A. Yip, T. Chan, T. Mathew, A Scale Dependent Model for Clustering by Optimization of Homogeneity and Separation, CAM Technical Report 03-37, Department of Mathematics, University of California, Los Angeles, 2003.

[27] R. Sharan, R. Shamir, CLICK: A clustering algorithm for gene expression analysis, in: Proc. International Conference on Intelligent Systems for Molecular Biology, 2000, p. 260â€“268.

[28] R. Duda, P. Hart, D. Stork, Pattern classification, Wiley-Interscience, New York, 2001.

[29] P. Francis, D. Leon, M. Minch, A. Podgurski, Tree-based methods for classifying software failures, in: Int. Symp. on Software Reliability Engineering, IEEE Computer Society, 2004, pp. 451–462.

[30] F. Casati, M. Shan, Semantic analysis of business process executions, in: C. S. Jensen, K. G. Jeffery, J. Pokorný, S. Saltenis, E. Bertino, K. Böhm, M. Jarke (Eds.), Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague,

Czech Republic, March 25-27, Proceedings, Vol. 2287 of Lecture Notes in Computer Science, Springer, 2002, pp. 287–296.

[31] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, M. Shan, Business process intelligence, Computers in Industry 53 (3) (2004) 321–343.

[32] D. Sell, L. Cabral, E. Motta, J. Domingue, 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, IEEE Computer Society, 2005, pp. 543–547.

[33] M. Hepp, F. Leymann, J. Domingue, A. Wahler, 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, M. Wang (Eds.), 2005 IEEE International Conference on e-Business Engineering (ICEBE 2005), 18-21 October 2005, Beijing, China, IEEE Computer Society, 2005, pp. 535–540.

[34] M. Hepp, D. Roman, An ontology framework for semantic business process management, in: A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, B. Schnizler (Eds.), eOrganisation: Service-, Prozess-, Market-Engineering: 8. Internationale Tagung Wirtschaftsinformatik - Band 1, WI 2007, Karlsruhe, Germany, February 28 - March 2, 2007, Universitaetsverlag Karlsruhe, 2007, pp. 423–440.

[35] C. Pedrinaci, J. Domingue, C. Brelage, T. van Lessen, D. Karastoyanova, 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, IEEE Computer Society, 2008, pp. 546–553.

[36] C. Pedrinaci, J. Domingue, Towards an ontology for process monitoring and mining, in: M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, N. Stojanovic (Eds.), 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, Vol. 251 of CEUR Workshop Proceedings, 2007.

[37] W. M. P. van der Aalst, H. T. de Beer, 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, S. Spaccapietra (Eds.), 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, Vol. 3760 of Lecture Notes in Computer Science, Springer, 2005, pp. 130–147.

[38] A. K. A. de Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, L. Cabral, An outlook

on semantic business process mining and monitoring, in: R. Meersman, Z. Tari, P. Herrero (Eds.), 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, Vol. 4806 of Lecture Notes in Computer Science, Springer, 2007, pp. 1244–1255.

[39] M. E. Kharbili, S. Stein, E. Pulvermüller, Policy-based semantic compliance checking for business process management., MobIS Workshops 420 (2008) 178–192.

[40] R. P. J. C. Bose, W. van der Aalst, Abstractions in process mining: A taxonomy of patterns, in: U. Dayal, J. Eder, J. Koehler, H. A. Reijers (Eds.), Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings, Vol. 5701 of Lecture Notes in Computer Science, 2009, pp. 159–175.

[41] K. Z. Haigh, F. Yaman, RECYCLE: learning looping workflows from annotated traces, ACM TIST 2 (4) (2011) 42:1–42:32.

[42] W. Jareevongpiboon, P. Janecek, Ontological approach to enhance results of business process mining and analysis, Business Proc. Manag. Journal 19 (3) (2013) 459–476.

[43] Z. Huang, X. Lu, H. Duan, On mining clinical pathway patterns from

medical behaviors, Artificial Intelligence in Medicine 56 (1) (2012) 35–50.

[44] S. Tsumoto, H. Iwata, S. Hirano, Y. Tsumoto, Similarity-based behavior and process mining of medical practices, Future Generation Comp. Syst. 33 (2014) 21–31.

[45] T. Neumuth, Surgical process modeling, Innovative Surgical Sciences 2 (2017) 123–138.

[46] Z. Huang, W. Dong, L. Ji, C. Gan, X. Lu, H. Duan, Discovery of clinical pathway patterns from event logs using probabilistic topic models, Journal of Biomedical Informatics 47 (2014) 39–57.

[47] R. Gatta, J. Lenkowicz, M. Vallati, E. Rojas, A. Damiani, L. Sacchi, B. D. Bari, A. Dagliati, C. Fernández-Llatas, M. Montesi, A. Marchetti, M. Castellano, V. Valentini, pminer: An innovative R library for performing process mining in medicine, in: A. ten Teije, C. Popow, J. H. Holmes, L. Sacchi (Eds.), Artificial Intelligence in Medicine - 16th Conference on Artificial Intelligence in Medicine, AIME 2017, Vienna, Austria, June 21-24, 2017, Proceedings, Vol. 10259 of Lecture Notes in Computer Science, Springer, 2017, pp. 351–355.

[48] A. Dagliati, L. Sacchi, A. Zambelli, V. Tibollo, L. Pavesi, J. H. Holmes, R. Bellazzi, Temporal electronic phenotyping by mining careflows of breast cancer patients, Journal of Biomedical Informatics 66 (2017) 136–147.

[49] S. Kapetanakis, M. Petridis, B. Knight, J. Ma, L. Bacon, A case based reasoning approach for the monitoring of business workflows, in: I. Bichindaritz, S. Montani (Eds.), Proc. International Conference on Case Based Reasoning (ICCBR), Vol. 6176 of Lecture Notes in Computer Science, Springer, Berlin, 2010, pp. 390–405.

[50] C. Combi, M. Gozzi, B. Oliboni, J. Juarez, R. Marin, Temporal similarity measures for querying clinical workflows, Artificial Intelligence in Medicine 46 (2009) 37–54.

[51] R. Bergmann, Y. Gil, Similarity assessment and efficient retrieval of semantic workflows, Information Systems 40 (2014) 115–127.

[52] S. Montani, G. Leonardi, S. Quaglini, A. Cavallini, G. Micieli, A knowledge-intensive approach to process similarity calculation, Expert Syst. Appl. 42 (9) (2015) 4207–4215.

[53] S. Montani, G. Leonardi, S. Quaglini, A. Cavallini, G. Micieli, A knowledge-intensive approach to process similarity calculation, Expert Syst. Appl. 42 (9) (2015) 4207–4215.