24th International Conference on Business Information Systems (BIS 2021)

Knowledge Graphs

https://doi.org/10.52825/bis.v1i.39

Published: 02 July 2021

Domain-specific Event Abstraction

Finn Klessascheck[1], Tom Lichtenstein[1], Martin Meier[1], Simon Remy[1], Jan Philipp Sachs[2,4], Luise Pufahl[3], Riccardo Miotto[4,5], Erwin Böttinger[2,4], and Mathias Weske[1]

[1]Hasso Plattner Institute (HPI), University of Potsdam, Potsdam, Germany

[2]Digital Health Center, HPI, University of Potsdam, Potsdam, Germany

[3]Software & Business Engineering, Technische Universitaet Berlin, Berlin, Germany

[4]Hasso Plattner Institute for Digital Health at Mount Sinai, Icahn School of Medicine at Mount Sinai, New York, USA

[5]Department of Genetics and Genomic Sciences, Icahn School of Medicine at Mount Sinai, New York, USA

Abstract. Process mining aims at deriving process knowledge from event logs, which contain data recorded during process executions. Typically, event logs need to be generated from process execution data, stored in different kinds of information systems. In complex domains like healthcare, data is available only at different levels of granularity. Event abstraction techniques allow the transformation of events to a common level of granularity, which enables effective process mining. Existing event abstraction techniques do not sufficiently take into account domain knowledge and, as a result, fail to deliver suitable event logs in complex application domains. This paper presents an event abstraction method based on domain ontologies. We show that the method introduced generates semantically meaningful high-level events, suitable for process mining; it is evaluated on real-world patient treatment data of a large U.S. health system.

Keywords: Process mining, Event abstraction, Domain knowledge, Healthcare

1 Introduction

Many organizations have an inherent interest to monitor and understand their processes. For example, analyzing and adopting processes can improve their overall efficiency, e nsure that legal requirements are met, and maintain a desired quality level. To this end, process mining provides techniques to analyze processes based on event data recorded during their execution. However, such event data is not always available in the necessary format and often differs in its granularity in complex settings. This also applies to treatment processes in hospitals, which are typically highly heterogeneous, complex, multidisciplinary, ad-hoc, and susceptible to change [1]. In the past, process mining has been proven as a technique well-suited to derive an understanding of medical processes, like patient-flows, and to improve them accordingly [2].

When extracting event data for process mining of electronic health records (EHRs), multiple data sources have to be tapped into, including hospital information systems. This variety of data sources and differences in data granularity leads to a mismatch in the level of abstraction between different events. Moreover, the fact that many events are documented manually by physicians or other medical personnel typically leads to varying degrees of detail in the recorded events. Using the resulting event logs would result in complex process models [3]–[5]. In order to generate event logs with events of comparable granularity and to elicit useful process models, event abstraction is needed. To this end, a rich set of research works on event abstraction

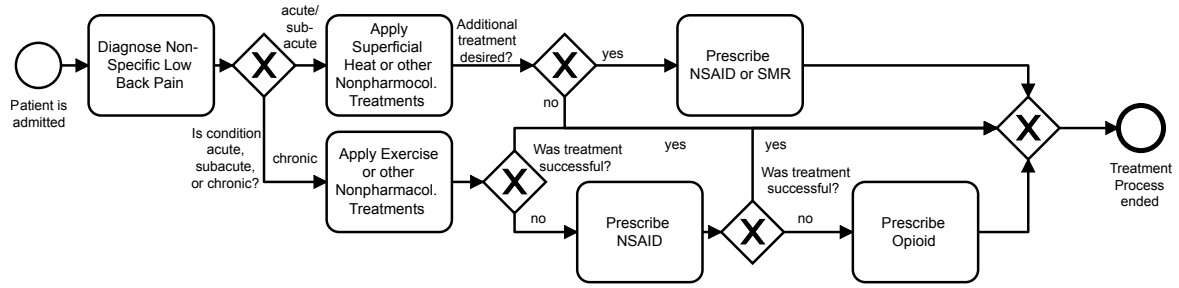

Figure 1. BPMN process model describing the treatment of LBP as per [7]. The events appearing in the event log of List. 1 belong to the same process but do not align with the activities of the process model, and make it difficult to put them into relation.

methods and techniques exists, as shown in [3], [6]. But fewer research works utilize existing domain knowledge. Those approaches mostly follow a bottom-up principle by starting from the observed event data. While the resulting process models are less complex and already improve the process analysis, bottom-up approaches do not guarantee a conceptually sound abstraction. There is a high prevalence of standards and ontologies in the medical field, containing domain-specific information such as medications, procedures, and diagnoses. This knowledge can be used to enrich abstraction mechanisms by actively selecting a use case-specific level of abstraction.

This paper presents an event abstraction method based on domain knowledge for process analysts to generate event logs for process mining. The method is domain-agnostic but was developed and tested for the healthcare domain. In the remainder of this paper, a motivating example is given in Sect. 2 followed by an overview of related work in Sect. 3. Afterwards, the domain-specific abstraction method, is presented in Sect. 4. Our approach is applied to the back pain treatment process data of a large health system in the U.S. in Sect. 5. The work is concluded and future research discussed in Sect. 6.

2 Motivating Example

In this section, we discuss the treatment process of non-specific low back pain (LBP), which is one of the most frequent complaints, as a motivating example. The evidence-based framework for diagnosis and treatment of LBP in the U.S. context is defined by a guideline from the American College of Physicians, which outlines under which circumstances and in what order certain interventions are to be conducted or medications to be prescribed [7], as shown in Fig. 1. As such, the guideline encompasses recommendations with regards to the prescription of a multitude of different drug classes, e.g., opioids (e.g., Oxycodone), non-steroidal anti-inflammatory drugs (NSAID; e.g., Ibuprofen), or skeletal muscle relaxants (SMR; e.g., Diazepam). Based on this, it could be of particular interest for a health system to which extent current treatment processes comply with the given clinical guideline.

```
<event>
    <date key=''time:timestamp'' value=''2018-08-04T13:04''/>
    <string key=''concept:name'' value=''NAPROXEN 500 MG TABLET''/>
    <string key=''event:context'' value=''EPIC MEDICATION''/>
    <string key=''event:code'' value=''19918''/>
</event>
<event>
    <date key=''time:timestamp'' value=''2018-09-01T13:59''/>
    <string key=''concept:name'' value=''OXYCODONE 5 MG TABLET''/>
    <string key=''event:context'' value=''EPIC MEDICATION''/>
    <string key=''event:code'' value=''20627''/>
</event>
```

Listing 1. Excerpt of a real-world event log containing low-level events.

The event data recorded in the hospital information system (HIS) is stored in a detailed manner due to medical necessity and billing purposes, including precise information about, e.g., the dosage of medications. Listing 1, extracted out of an EHR database, shows the event names that result from this data. While the event log contains fine-grained events, the treatment guideline modeled in Fig. 1 is concerned with more abstract, high-level activities (e.g., *Prescribe NSAID*)[1]. This makes it difficult to compare the guideline with the recorded treatment event data. Due to the fine-grained event data, discovering a process model without processing it beforehand would result in a highly complex model that would be hard to interpret. It would include, for example, numerous activities, splits and branches for each drug and its potential dosage. Thus, event abstraction is needed to enable the alignment of low-level events with the high-level activities of a clinical guideline and to discover more human-readable process models.

An easy approach would be to abstract all events that refer to the administration of drugs into the high-level event *Administer Drug*. On one side, this would reduce the complexity. On the other side, valuable information, like the administered drug, would be discarded, such that this abstraction would not be useful and diverge from the level of detail given by the clinical guideline. It would, for example, not be possible to differentiate between non-pharmacological treatments and pharmacological treatments or non-opioid and opioid treatments, as per the guideline. However, this categorization cannot be simply made with the available event information from the data warehouse (cf. List. 1), so that additional information, e.g., about the drug class or the effect mechanisms, is necessary. Standards and ontologies that contain information about medical concepts such as medications, procedures, and diagnoses, already exist in healthcare. Thus, in this work, we explore the application of such medical knowledge resources for exploiting the medical context of events and determining the right level of abstraction.

3 Related Work

In the following, we discuss related work on event abstraction in general and with a particular focus on healthcare. Diba et al. [6] identifies event abstraction as one out of three major tasks for event log generation. Similarly, van Zelst et al. [3] provides a taxonomy for event abstraction techniques and categorizations. In general, abstractions techniques can be distinguished based on their information richness, which again depends on the required input [6]. The first category comprises clustering and unsupervised learning approaches like [8], [9], which require no additional input. In addition, Richetti et al. [9] applies natural language processing to discover activities and objects from the event labels. Based on their semantic relations, similar activities get clustered and substituted. In contrast, Tax et al. [10] applies supervised learning to map low-level events to activity executions that requires labelled training data to learn the probabilistic mappings. Leemans et al. [11] utilizes event attributes to derive hierarchical models encapsulating low-level events in sub-processes. The approach lacks of a mechanism to guarantee semantically sound abstraction groups.

Another category comprises approaches using behavioural patterns to transfer low-level events into higher-level activities. As an example, Mannhardt et al. [12] provides a supervised event abstraction technique. With the help of behavioural activity patterns, domain knowledge is captured, based on which events are matched. Additionally, the approach by Baier et al. [13] utilizes enriched process models. They describe the issue of *n:m* mappings from events to activities and use an annotated process model to identify mappings between events and activities in which they additionally encoded domain knowledge. With the help of a gamified crowdsourcing approach, Sadeghianasl et al. [4] requires a group of domain-experts to participate in the game to equalize activity labels. Lastly, the authors of [14] present a knowledge-based

[1] Note that, while the example of this section only consists of medication-related events, the overall problem exists for diagnostic-/treatment-related events as well.

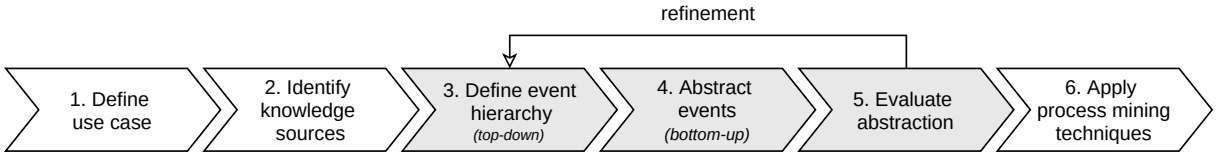

Figure 2. Method for domain-specific event abstraction. After defining the use case and identification of relevant knowledge sources, an adequate abstraction level is derived in an iterative manner.

abstraction mechanism based on domain-specific ontologies. Depending on a set of rules, events are mapped to the fundamental concepts of the ontologies; multiple events matching the same ground term within a given time window get merged into macro-activities. However, this approach heavily relies on predefined mapping rules.

For the healthcare domain, Mans et al. [15] illustrates the problem of different levels of data granularity by providing an overview of the spectrum of available data in an HIS. The authors classify different source systems based on the abstraction level and the data accuracy. Those findings are confirmed in a case study [16] on event log generation in healthcare in which event abstraction challenges and the need for suitable techniques, like abstraction tables, are discussed. In order to investigate treatment processes, the authors of [5] use, similarly to [11], sub-processes to encapsulate low-level events of treatment processes in sub-processes. The gold standard of the approach relies on the encoding of domain-knowledge within the activity labels, which might be added manually.

Most of the presented approaches are data-driven only and rely on structural features like control-flow pattern, while only a few also consider additional contextual information, e.g., [12], [14]. We will refer to those approaches as bottom-up. Still, they are missing mechanisms to generate a consistent level of abstraction, which is also medically sound.

4 Domain-specific Abstraction Method

In order to overcome the limitations of existing approaches, we introduce a method for domain-specific event abstraction. While the domain for the abstraction is not limited in any sense, one concrete abstraction instance can only be applied within one specific domain, and is not reusable across multiple domains. In the context of this work, the term abstraction means the unification of event identifiers for related events, so that several events are not combined to one event, but assigned to the same event class. The presented method combines two basic concepts of event abstraction, namely *bottom-up* and *top-down*. In the following, both concepts are presented, as well as the concrete abstraction procedure.

4.1 Basic Concepts of Event Abstraction

The first concept, which will subsequently be referred to as *bottom-up*, uses the low-level event log as its only input. Most of the abstraction approaches described in Sect. 3 can be considered to follow this concept. The advantage of a bottom-up abstraction is that little or no additional domain-specific information is required. However, such approaches produce results which might include high-level events at different levels of abstraction. Depending on the use case, it may be desirable to maintain a certain level of abstraction across the output event log as well as to produce high-level sound events for a domain. An abstraction that relies exclusively on the data level is therefore not sufficient in this case.

One way to overcome the lack of domain-specific information is to use an abstraction procedure following the concept of *top-down* abstraction. Top-down approaches first define the goal of the abstraction for a use case based on domain-specific information and map the low-level events to the targeted high-level events. In comparison to the bottom-up concept, the patterns

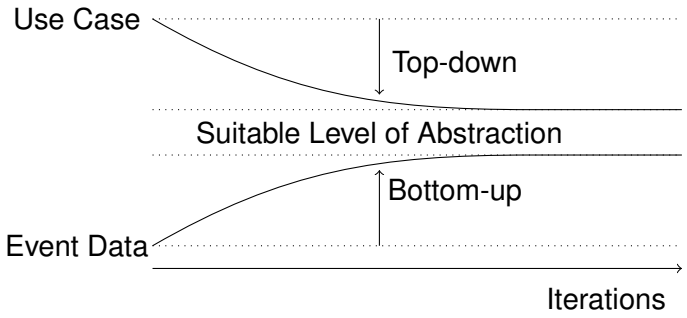

Figure 3. Conceptual depiction of the abstraction method. The use case and event data are alternately investigated from the top-down and bottom-up perspective to further restrict the abstraction scope.

and rules are not detected in the log, but defined by analysts in advance. The domain-specific patterns and rules are used to group related low-level events together using an *assignment algorithm* [12], [14].

The *top-down* approach allows to explicitly define the desired level of abstraction and can thus significantly improve the analysis of the process. However, an accurate abstraction of low-level events requires a deep understanding of the underlying data. Additionally, the abstraction must be conducted in such a way that it fits to the present data, which can be very complex and detailed, and is hard for the analyst to know beforehand. This can make exploratory analyses particularly difficult.

4.2 Alternating Event Abstraction

In order to find a suitable level of abstraction considering the available low-level event data as well as the abstraction goal of a use case, abstraction concepts of bottom-up and top-down are combined. Figure 2 depicts the method, which consists of six steps. In the first step, a use case needs to be defined due to the nature of the involved top-down concept. The use case may be guided by a domain-specific question, e.g., relating a treatment process to clinical guidelines or regulatory aspects. Next, knowledge sources relevant for the use case are collected and used as input for the subsequent steps. Before actual process mining techniques can be applied in step 6, an adequate abstraction level needs to be derived from steps 3 to 5 in an iterative manner. In these steps, the data and use case are alternately investigated from the top-down and the bottom-up perspective to further restrict the abstraction scope from each perspective with every iteration, as illustrated in Fig. 3. In order to achieve this, this paper proposes a hierarchy-based abstraction procedure.

Event Hierarchy. Since the desired granularity of an abstraction strongly depends on the use case, an *event hierarchy* is created as a basis for the abstraction of low-level event identifiers, which describes groups of related terms that are essential for the analysis. The event hierarchy is designed from a top-down perspective. It consists of several *abstraction sets*. Each abstraction set consists of an *abstraction term* and a *list of subordinate terms*. The abstraction term serves as the high-level event identifier for each low-level event that can be assigned to the abstraction set with the help of the corresponding subordinate terms. The general structure of an event hierarchy, as well as an exemplary abstraction set in the medical context, is illustrated in Fig. 4.

A relevant abstraction term in the medical context could be, for example, a class of drugs, such as NSAID. The low-level events are then mapped to the subordinate terms contained in the lists of the abstraction sets and abstracted to the corresponding abstraction term. The mapping is done by checking for each event whether its name contains one or multiple terms from the lists of the subordinate terms. Depending on the quality of the provided abstraction

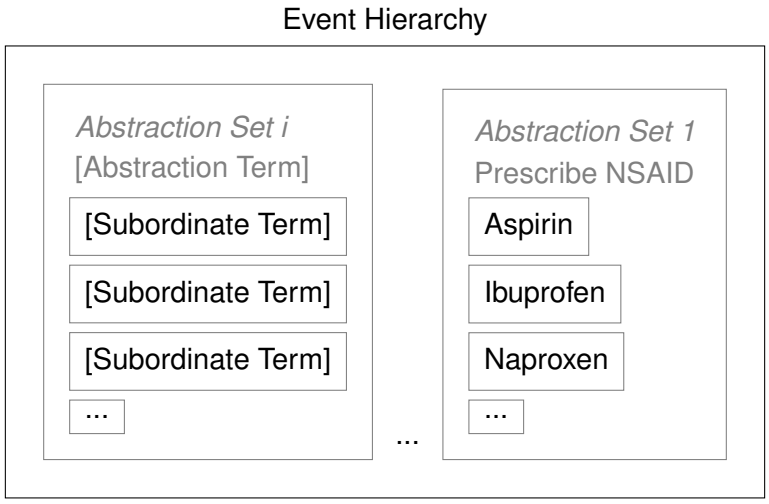

Figure 4. General structure of an event hierarchy with an example abstraction set.

sets, a single event name can contain different terms from different abstraction sets. In this case, either a domain expert can decide on the assignment, or the Levenshtein distance [17] or other kinds of distance measures could be used to determine the term most closely related to the event identifier. In order to avoid misleading abstractions, the lists of subordinate terms of different abstraction sets should not overlap. Having distinct abstraction sets allows a precise view of the data, which is directly tailored to the use case. After finding a matching term, the event identifier is then changed to the abstract term of the abstraction set. To preserve as much information as possible, the original name is added as an attribute to the event. If an event cannot be assigned to an abstraction set, the event keeps its original name. An abstraction set can additionally be marked as a filter set. If an event is assigned to an abstraction set that is marked as a filter, it is not included in the resulting event log.

Iterative Approximation of the Abstraction Level. Each iteration starts from the top-down perspective with the creation or refinement of the event hierarchy. After the abstraction, the resulting high-level events are analyzed from the bottom-up perspective. More concrete, it can be determined whether more abstraction sets should be added, or whether additional subordinate terms derived from low-level events that were not previously covered by any abstraction set should be added to one. Special attention should be paid to events that have not been abstracted to high-level events and to abstraction sets that abstract large parts of the event log. This evaluation can be done by either directly examining the resulting high-level events or by evaluating a process model, mined from the resulting event log. With that, the resulting event hierarchy is progressively refined towards the selected use case. Because of this, it cannot be used arbitrarily for other use cases.

5 Application to a Real-World Use Case

This section presents an exemplary application of the abstraction method to the real-world use case of Sect. 2. While the method itself is domain-agnostic in its application, a concrete example from the healthcare domain is suited to illustrate the concrete steps and their usefulness. As the first two steps (i.e., the use case definition and the collection of additional knowledge sources) are described in Sect. 2, this section evaluates and illustrates steps three to five.

5.1 Experimental Setup and Data Preparation

In this work we used EHR data from the Mount Sinai Health System, a large health system located in New York, NY, which generates a high volume of structured, semi-structured and unstructured data from inpatient, outpatient, and emergency room visits. Patients in the system

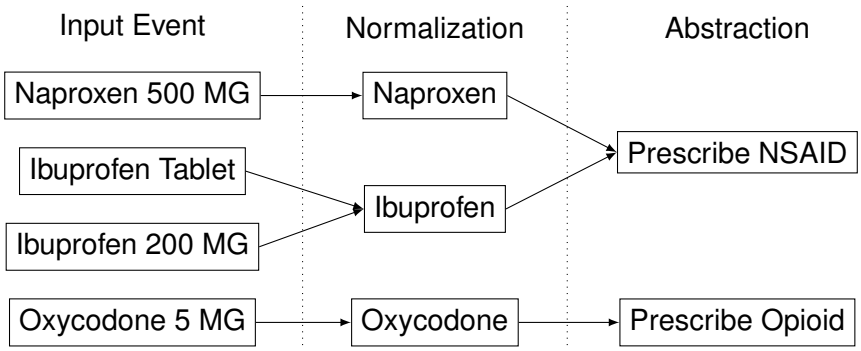

Figure 5. Example event abstraction process to illustrate the normalization and abstraction from the event log excerpt in List. 1 to the event log in List. 2.

can have up to 15 years of follow-up data. We accessed a de-identified dataset containing approximately 8.1 million patients from eight hospitals within the system, spanning the years from 2003 to 2018. This research was approved by the Institutional Review Board (IRB)[2] of the institution fully compliant with the HIPAA[3] regulations. In particular, in this study we included diagnosis codes (ICD-9/10), medications and procedures.

In order to be able to apply the method to the available data, preprocessing steps are necessary. Those steps include the normalization of event descriptors. This is necessary to not encode dosage information and other factors, which would otherwise lead to highly complex event hierarchies, without providing any additional value for the process analysis. The normalization is part of our prototypical implementation and is based on manually crafted rules derived by a medical expert. The normalization of the event log excerpt given in List. 1 is illustrated in Fig. 5, while the normalized event descriptors can be found in List. 2 attached as attributes. After the normalization, the iterative abstraction is applied to the low-level event log as described in Sect. 4.

```
<event>
    <date key=''time:timestamp'' value=''2018-08-04T13:04''/>
    <string key=''concept:name'' value=''Prescribe NSAID''/>
    <string key=''event:description'' value=''NAPROXEN 500 MG TABLET''/>
    <string key=''event:normalized'' value=''Naproxen''/>
    <string key=''event:context'' value=''EPIC MEDICATION''/>
    <string key=''event:code'' value=''19918''/>
</event>
<event>
    <date key=''time:timestamp'' value=''2018-09-01T13:59''/>
    <string key=''concept:name'' value=''Prescribe Opioid''/>
    <string key=''event:description'' value=''OXYCODONE 5 MG TABLET''/>
    <string key=''event:normalized'' value=''Oxycodone''/>
    <string key=''event:context'' value=''EPIC MEDICATION''/>
    <string key=''event:code'' value=''20627''/>
</event>
```

Listing 2. Example event log extract after the application of the abstraction method.

5.2 Evaluation

To demonstrate the usefulness of the proposed method, we created and analyzed two event logs using a prototypical implementation[4]. A detailed description of the event log generation pipeline can be found in [16]. Each log comprises 2,000 patients diagnosed with low back pain from the Mount Sinai Health System with a focus on the treatment processes. When creating the first event log, no abstraction was applied, and all events were included with the granularity

[2] IRB-19-02369

[3] Health Insurance Portability and Accountability Act

[4] https://github.com/bptlab/fiber2xes

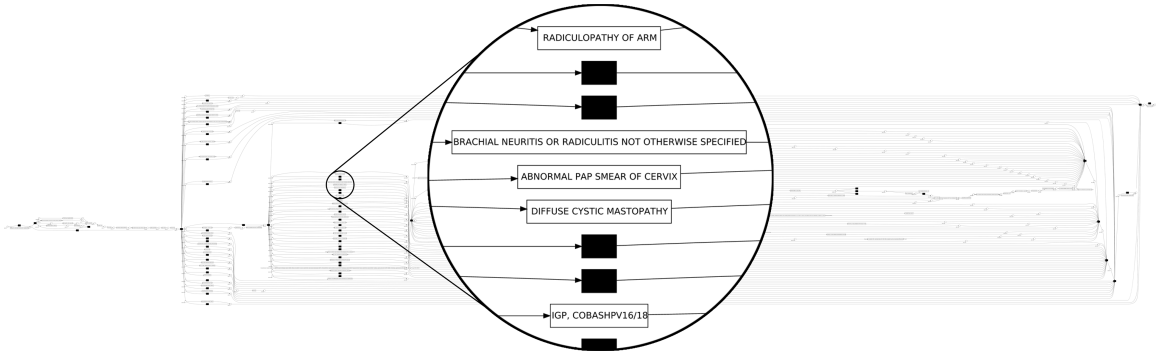

Figure 6. Discovered process model based on only three randomly selected traces from the event log without any abstraction and 60% noise filter.

as they were recorded in the data warehouse. To mine process models form the different event logs, we used the inductive miner plug-in [18] of ProM[5].

Fig. 6 shows the resulting process model, involving only three randomly selected traces of the event log without any abstraction. A visual examination of the diagram is very complex because of the large number of nodes and edges. A comparison to the model representing the clinical guideline in Fig. 1 is hardly possible. This complexity also complicates a variant analysis. The event log without any abstraction contains 1,998 trace variants and 28,540 different event classes. In contrast, Fig. 7 depicts the process model discovered from an event log, where our method was applied during its generation. Apparently, this model is much more compact. The abstracted event log contains 1,300 variants and the seven event classes that were previously defined in the event hierarchy. With this event log, the activities can be directly related to those present in the clinical guideline in Fig. 1. However, the abstracted event log still contains a high number of trace variants, which reflects the flexible nature of treatment processes. In summary, the event abstraction has resulted in a simpler process model that facilitates both interpretation and alignment with the clinical guideline.

5.3 Discussion

The combination of top-down and bottom-up concepts takes advantage of existing structured knowledge and outlines a clear procedure to produce sound abstractions with regard to the underlying domain that lead to semantically meaningful events. In other words, it enables the domain-specific event abstraction of event data. While this method requires manual work and input of a domain expert, the resulting abstractions have been shown capable of increasing the understandability and interpretability of the event logs. Abstractions produced by applying the presented method are comprehensible - there is no “black box” leading to abstracted events, and domain experts are always able to verify the validity of the abstraction. Furthermore, the method makes it possible to easily vary the level of abstraction when investigating a given use case. Also, resulting abstractions are highly flexible and configurable. In contrast to Leonardi et al. [14], no custom ontology needs to be defined, since the abstraction sets are directly derived from existing knowledge sources, like existing standards. This also increases their reusability in other event hierarchies as they have a certain general validity. The presented method could also be combined with existing abstraction techniques discussed in Sect. 3, such as the work of Mannhardt et al. [12], in order to add further perspectives to the resulting event log. They could be used, for example, to add a temporal perspective to the abstraction, where subsequent steps of a larger treatment process, are abstracted into a single event. Furthermore, this method could be improved by adding support for automating the creation of the event hierarchy, or evaluation of the abstraction results based on different metrics.

[5] http://www.promtools.org/

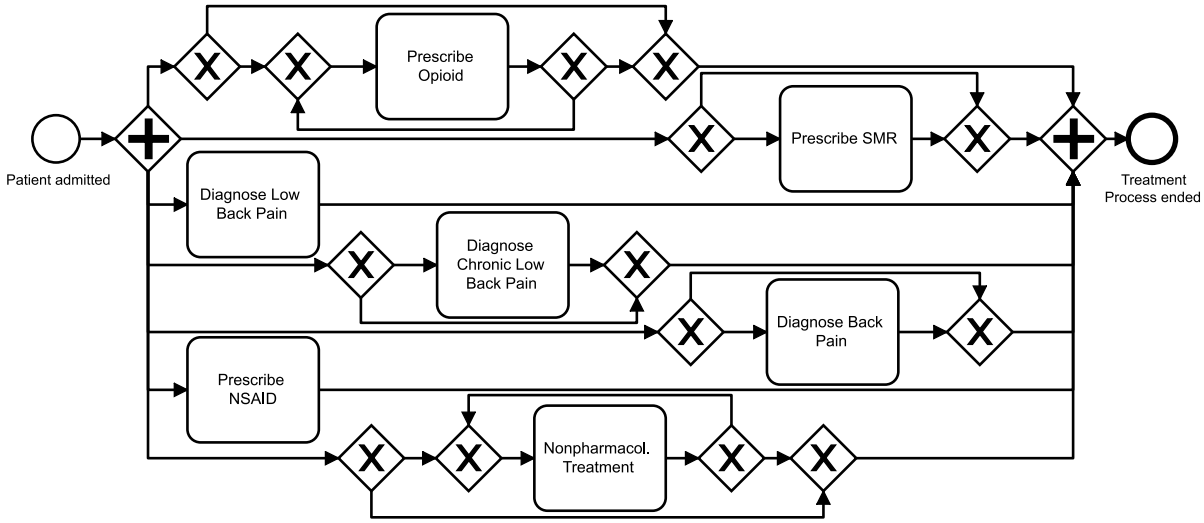

Figure 7. Discovered process model, including all 2,000 patients based on the event log after event abstraction, with 40% noise filter.

6 Conclusion

Different levels of granularity in event data hamper the effective use of process mining techniques. Therefore we proposed a structured method to achieve a semantically meaningful event abstraction. By combining top-down and bottom-up approaches, conceptually sound abstractions from low-level events are reached. Applying the method to a real-world event log from a U.S. health system shows the usefulness of the approach. Currently, we have developed and tested the domain-specific method exclusively for healthcare. Still, the steps are general and might also be useful for other domains, which need to be further studied. For future work, different metrics to evaluate the quality of generated event hierarchies and abstraction sets are worth investigating, as they could help guiding the application of the method through the different iterations. Moreover, interfaces for integrating existing ontologies such as RxNorm or SNOMED CT, as well as a more semi-automated approach for creation of event hierarchies, could reduce the manual effort.

Acknowledgments. Research reported in this paper was supported by the Office of Research Infrastructure of the National Institutes of Health under award numbers S10OD026880. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. This work was supported in part through the computational and data resources and staff expertise provided by Scientific Computing at the Icahn School of Medicine at Mount Sinai.

References

[1] P. Homayounfar, “Process mining challenges in hospital information systems,” in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2012, pp. 1135–1140.

[2] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro, “Process mining in healthcare: A literature review,” *Journal of Biomedical Informatics*, vol. 61, pp. 224–236, 2016.

[3] S. J. van Zelst, F. Mannhardt, M. de Leoni, and A. Koschmider, “Event abstraction in process mining: Literature review and taxonomy,” *Granular Computing*, pp. 1–18, 2020.

[4] S. Sadeghianasl, A. H. M. ter Hofstede, S. Suriadi, and S. Turkay, “Collaborative and interactive detection and repair of activity labels in process event logs,” in *2nd International Conference on Process Mining, ICPM*, 2020, pp. 41–48.

[5] X. Lu, A. Gal, and H. A. Reijers, “Discovering hierarchical processes using flexible activity trees for event abstraction,” in *2nd International Conference on Process Mining, ICPM*, 2020, pp. 145–152.

[6] K. Diba, K. Batoulis, M. Weidlich, and M. Weske, “Extraction, correlation, and abstraction of event data for process mining,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 3, 2020.

[7] A. Qaseem, T. J. Wilt, R. M. McLean, and M. A. Forciea, “Noninvasive treatments for acute, subacute, and chronic low back pain: A clinical practice guideline from the american college of physicians,” *Annals of Internal Medicine*, vol. 166, no. 7, pp. 514–530, 2017.

[8] F. Folino, M. Guarascio, and L. Pontieri, “Mining predictive process models out of low-level multidimensional logs,” in *Advanced Information Systems Engineering (CAiSE)*, ser. LNCS, vol. 8484, Springer, 2014, pp. 533–547.

[9] P. H. P. Richetti, F. A. Baião, and F. M. Santoro, “Declarative process mining: Reducing discovered models complexity by pre-processing event logs,” in *Business Process Management BPM*, ser. LNCS, vol. 8659, Springer, 2014, pp. 400–407.

[10] N. Tax, N. Sidorova, R. Haakma, and W. M. P. van der Aalst, “Event abstraction for process mining using supervised learning techniques,” in *Proceedings of SAI Intelligent Systems Conference (IntelliSys)*, ser. LNNS, vol. 15, Springer, 2016.

[11] S. J. J. Leemans, K. Goel, and S. J. van Zelst, “Using multi-level information in hierarchical process mining: Balancing behavioural quality and model complexity,” in *2nd International Conference on Process Mining, ICPM*, 2020, pp. 137–144.

[12] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, and P. J. Toussaint, “Guided process discovery - A pattern-based approach,” *Information Systems*, vol. 76, pp. 1–18, 2018.

[13] T. Baier, J. Mendling, and M. Weske, “Bridging abstraction layers in process mining,” *Information Systems*, vol. 46, pp. 123–139, 2014.

[14] G. Leonardi, M. Striani, S. Quaglini, A. Cavallini, and S. Montani, “Towards semantic process mining through knowledge-based trace abstraction,” in *International Symposium on Data-Driven Process Discovery and Analysis*, ser. LNBIP, vol. 340, Springer, 2017, pp. 45–64.

[15] R. Mans, W. M. P. van der Aalst, R. J. B. Vanwersch, and A. J. Moleman, “Process mining in healthcare: Data challenges when answering frequently posed questions,” in *Process Support and Knowledge Representation in Health Care - BPM, Revised Selected Papers*, ser. LNCS, vol. 7738, Springer, 2012, pp. 140–153.

[16] S. Remy, L. Pufahl, J.-P. Sachs, E. P. Böttinger, and M. Weske, “Event log generation in a health system: A case study,” in *International Conference on Business Process Management*, ser. LNCS, Springer, 2020.

[17] V. I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[18] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, “Discovering block-structured process models from event logs containing infrequent behaviour,” in *Business Process Management Workshops*, ser. LNBIP, vol. 171, Springer, 2013, pp. 66–78.