

# Mining Process Model Descriptions of Daily Life through Event Abstraction

N. Tax, N. Sidorova, R. Haakma, W.M.P. van der Aalst

**Abstract** Process mining techniques focus on extracting insight in processes from event logs. Process mining has the potential to provide valuable insights in (un)healthy habits and to contribute to ambient assisted living solutions when applied on data from smart home environments. However, events recorded in smart home environments are on the level of sensor triggers, at which process discovery algorithms produce overgeneralizing process models that allow for too much behavior and that are difficult to interpret for human experts. We show that abstracting the events to a higher-level interpretation can enable discovery of more precise and more comprehensible models. We present a framework for the extraction of features that can be used for abstraction with supervised learning methods that is based on the XES IEEE standard for event logs. This framework can automatically abstract sensor-level events to their interpretation at the human activity level, after training it on training data for which both the sensor and human activity events are known. We demonstrate our abstraction framework on three real-life smart home event logs and show that the process models that can be discovered after abstraction are more precise indeed.

---

Niek Tax  
Eindhoven University of Technology, e-mail: [n.tax@tue.nl](mailto:n.tax@tue.nl)

Natalia Sidorova  
Eindhoven University of Technology e-mail: [n.sidorova@tue.nl](mailto:n.sidorova@tue.nl)

Reinder Haakma  
Philips Research e-mail: [reinder.haakma@philips.com](mailto:reinder.haakma@philips.com)

Wil M.P. van der Aalst  
Eindhoven University of Technology e-mail: [w.m.p.v.d.aalst@tue.nl](mailto:w.m.p.v.d.aalst@tue.nl)

## 1 Introduction

Process mining is a fast growing discipline that combines methods from computational intelligence, data mining, process modeling and process analysis [1]. *Process discovery*, the task of extracting process models from logs, plays an important role in process mining. There are many different process discovery algorithms ([2, 7, 37, 36, 18]), which can discover many different types of process models, including BPMN models, Petri nets, process trees, UML activity diagrams, and state-charts.

While originally the scope of process mining has been on business processes, it has broadened in recent years towards other application areas, including the analysis of human behavior [29, 31, 32]. Process model descriptions of human behavior can be used amongst others to aid lifestyle coaching for healthy living, or to assess the ability of independent living of elderly or people with illness.

Events in the event log are generated by e.g. motion sensors placed in the home, power sensors placed on appliances, open/close sensors placed on closets and cabinets, etc. This clearly distinguishes process mining for smart homes from the traditional application domain of business processes, where events in the log are logged by IT systems when an business tasks are performed.

In event logs from business processes the event labels generally have a clear semantic meaning, like *register mortgage request*. In the smart home domain the events are on the sensor level, while the human expert is interested in analyzing the behavior in terms of activities of daily life. Additionally, simply using the sensor that generated the event as the event label has been shown to result in non-informative process models that overgeneralize the event log and allow for too much behavior [34]. In the field of process mining such overgeneralizing process models are generally referred to as being *imprecise*.

In our earlier work [31] we showed how to discover more precise process models by taking the name of the sensor as a starting point for the event label and then refine the labels using the time of the day at which the event occurred. However, labels in such process models still represent sensors, and they have no direct interpretation on the human activity level. In this paper we leverage diary style annotations of the activities performed on a human activity level and use them learn a mapping from sensor-level events to human activity events. This enables discovery of process models that describe the human activities directly, leading to more comprehensible and more precise descriptions of human behavior. Often it is infeasible or simply too expensive to obtain such diaries for periods of time longer than a couple of weeks. To mine a process model of human behavior more than a couple of weeks of data is needed. Therefore, there is a need to infer human level interpretations of behavior from sensors.

With supervised learning techniques the mapping from sensor-level events to human activity level events can be learned through examples, without requiring a hand-made ontology of how human activities relate to sensors. Similar approaches have been explored in the activity recognition field, where continuous-valued time series from sensors are mapped to time series of human activity. Change points in

these time series are triggered by sensor-level events like *opening/closing the fridge door*, and the annotations of the higher level events (e.g. *cooking*) are often obtained through manual activity diaries. However, in contrast to techniques from the activity recognition field, we operate on discrete events on the sensor-level instead of continuous time series.

In this paper we extend the work started in [33]. We describe a framework for supervised abstraction of events that enables the discovery of more precise process models from smart home event logs. Additionally, the process models obtained represent human activity directly, thereby enabling direct analysis of human behavior itself, instead of indirect analysis through sensor-level models. In Section 2 we give an overview of the related work from the activity recognition field. Basic concepts, notations, and definitions that we use throughout the rest of the paper are introduced in Section 3. In Section 4 we explain conceptually why abstraction from sensor-level to human activity level events can help to the process discovery step to find more precise process models. In Section 5 we describe a framework for retrieving useful features for abstraction from event logs using specific concepts of the IEEE XES standard for event logs [12]. Section 6 demonstrates the added value of supervised event abstraction for process mining in the smart home domain and show that it enables discovery of more precise models on three real life smart home event logs. Section 7 concludes the paper and identifies some areas of future work.

## 2 Related Work

Event abstraction based on supervised learning is an unexplored problem in process mining. Most related work for abstracting from sensor-level to human activity level events can be found in the field of activity recognition, which focuses on the task of detecting different types of human activity from either passive sensors [14, 30], wearable sensors [3, 16], or cameras [22].

Activity recognition methods generally operate on discrete time windows over the time series of continuous-valued sensor values and aim to map each time window onto the correct type of human activity, e.g. *eating* or *sleeping*. Activity recognition methods can be classified into probabilistic approaches [14, 30, 3, 16] and ontological reasoning approaches [5, 25]. The advantage of probabilistic approaches over ontological reasoning approaches is their ability to handle noisy, uncertain and incomplete sensor data [5].

Tapia [30] was the first to explore supervised learning for inference of human activities from passive sensors, using a naive Bayes classifier. Many more recent activity recognition approaches use probabilistic graphical models [14, 13]: Van Kasteren et al. [14] explored Conditional Random Fields [17] and Hidden Markov Models [23], and Van Kasteren and Kröse [13] applied Bayesian Networks [10] on the activity recognition task. Kim et al. [15] found Hidden Markov Models to be unable to capture long-range or transitive dependencies between observations,

which results in difficulties recognizing multiple interacting activities (concurrent or interwoven). Conditional Random Fields do not possess these limitations.

Our work differentiates itself from existing activity recognition work in the form of the input data on which they operate and in the goal that it aims to achieve. On the input side, activity recognition techniques consider the data to be a multidimensional time series of the sensor values over time, based on which time windows are mapped onto human activities. An appropriate time window size is determined using domain knowledge of the data set. Instead, we aim for a generic method that does not require this domain knowledge, and that works in general for any event log. An approach based on time windows contrasts with our aim for generality, as no single time window size exists that is suitable for all event logs. The durations of the events within a single event log might differ drastically (e.g. one event might take seconds, while another event takes months), in which case time window based approaches will either miss short events in case of larger time windows or resort to very large numbers of time windows resulting in very long computational time. Therefore, we map each individual sensor-level event to a human activity level event and do not use time windows. In a smart home environment context with passive sensors, each change in a binary sensor value can be considered to be a low-level event. A second difference with existing activity recognition techniques is that our framework aims to find an abstraction of the data that enables discovery of more precise process models, where classical activity recognition methods do not have a link with the application of process mining.

Other related work can be found in the area of process mining, where several techniques address the challenge of abstracting low-level (e.g. sensor-level) events to high level (e.g. human activity level) events ([4, 11, 9]). Most existing event abstraction methods rely on clustering methods, where each cluster of low-level events is interpreted as one single high-level event. However, using unsupervised learning introduces two new problems. First, it is unclear how to label high-level events that are obtained by clustering low-level events. Current techniques require the user / process analyst to provide high-level event labels themselves based on domain knowledge. Secondly, unsupervised learning gives no guidance with respect to the desired level of abstraction. Many existing event abstraction methods contain one or more parameters to control the size the clusters, and finding the right level of abstraction providing meaningful results is often a matter of trial-and-error.

One abstraction technique from the process mining field that does not rely on unsupervised learning is proposed by Mannhardt et al. [20]. This approach relies on domain knowledge to abstract low-level events into high-level events, requiring the user to specify a low-level process model for each high-level activity. However, in the context of human behavior it is unreasonable to expect the user to provide the process model in sensor terms for each human activity from domain knowledge.

### 3 Preliminaries

In this section we introduce basic concepts and notation used throughout the paper.

We use the standard sequence definition, and denote a sequence by listing its elements, e.g. we write  $\langle a_1, a_2, \dots, a_n \rangle$  for a (finite) sequence  $s : \{1, \dots, n\} \rightarrow S$  of elements from some alphabet  $S$ , where  $s(i) = a_i$  for any  $i \in \{1, \dots, n\}$ .

#### 3.1 Petri nets

A process modeling notation that is commonly used in process mining is the Petri net. Petri nets are directed bipartite graphs consisting of transitions and places, connected by arcs. Transitions represent activities, while places represent the state of the system before and after execution of a transition. Labels are assigned to transitions to indicate the type of activity that they represent. A special label  $\tau$  is used to represent invisible transitions, which are only used for routing purposes and do not represent any real activity.

**Definition 1 (Labeled Petri net).** A labeled Petri net is a tuple  $N = (P, T, F, R, \ell)$  where  $P$  is a finite set of places,  $T$  is a *finite set* of transitions such that  $P \cap T = \emptyset$ , and  $F \subseteq (P \times T) \cup (T \times P)$  is a set of directed arcs, called the flow relation,  $R$  is a finite set of labels representing event types, with  $\tau \notin R$  a label representing an invisible action, and  $\ell : T \rightarrow R \cup \{\tau\}$  is a labeling function that assigns a label to each transition.

The state of a Petri net is defined w.r.t. the state that a process instance can be in during its execution. A state of a Petri net is captured by the marking of its places with tokens. In a given state, each place is either empty, or it contains a certain number of tokens. A transition is enabled in a given marking if all places with an outgoing arc to this transition contain at least one token. Once a transition fires (i.e. is executed), a token is removed from all places with outgoing arcs to the firing transition and a token is put to all places with incoming arcs from the firing transition, leading to a new state.

**Definition 2 (Marking, Enabled transitions, and Firing).** A marked Petri net is a pair  $(N, M)$ , where  $N = (P, T, F, L, \ell)$  is a labeled Petri net and  $M \in \mathbb{N}^P$  denotes the marking of  $N$ . For  $n \in (P \cup T)$  we use  $\bullet n$  and  $n \bullet$  to denote the set of inputs and outputs of  $n$  respectively. Let  $C(s, e)$  indicate the number of occurrences (count) of element  $e$  in multiset  $s$ . A transition  $t \in T$  is enabled in a marking  $M$  of net  $N$  if  $\forall p \in \bullet t : C(M, p) > 0$ . An enabled transition  $t$  may fire, removing one token from each of the input places  $\bullet t$  and producing one token for each of the output places  $t \bullet$ .

Figure 1 shows four Petri nets, with the circles representing places, the squares representing transitions. The gray rectangles represent (invisible)  $\tau$ -transitions. Places depicted as  $\odot$  belong to the final marking, indicating that the process execution can terminate in this marking.

The Petri net shown in Figure 1c initially has one token in the place  $p1$ , indicated by the dot. Firing the enabled silent transition takes the token from  $p1$  and puts a token in both  $p2$  and  $p3$ , enabling both  $MC$  and  $DCC$ . When  $MC$  fires, it takes the token from  $p2$  and puts a token in  $p4$ . When  $DCC$  fires, it takes the token from  $p3$  and puts a token in  $p5$ . After  $MC$  and  $DCC$  have both fired, resulting in a token in both  $p4$  and  $p5$ ,  $W$  is enabled. Executing  $W$  takes the token from both  $p4$  and  $p5$ , and puts a token in  $p6$ , which is a place that belongs to the final marking, indicates that the process execution can stop here. Alternatively, it can fire the silent transition, taking the token from  $p6$  and placing a token in  $p2$  and  $p5$ , which allows for execution of  $MC$  and  $W$  to reach the marking consisting of  $p6$  again. We refer the interested reader to [24] for an extensive review of Petri nets.

### 3.2 Conditional Random Field

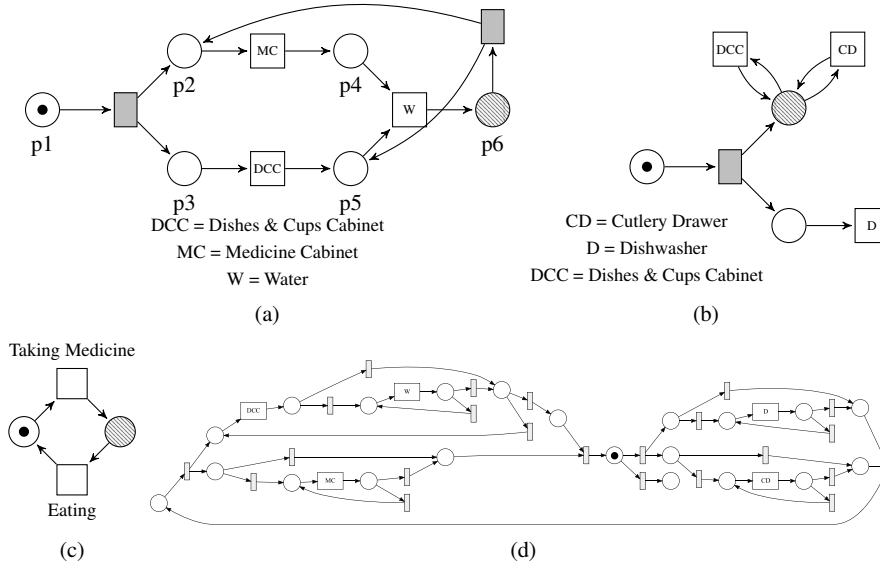
We consider the recognition of human activity level events as a sequence labeling task in which each sensor-level event is classified into one of the human activity level events. Linear-chain Conditional Random Fields (CRFs) [17] are a type of probabilistic graphical model which has shown to perform well on many sequence labeling tasks in the fields of language processing and computer vision. Conceptually CRFs can be regarded as a sequential version of multiclass logistic regression, i.e., the predictions in the prediction sequence are dependent on each other. A CRF models the conditional probability distribution of the label sequence given an observation sequence using a log-linear model. Linear-chain CRFs take the following form:

$$p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{t=1} \sum_k \lambda_k f_k(t, y_{t-1}, y_t, x)\right) \quad (1)$$

where  $Z(x)$  is the normalization factor which makes sure that the values of the probability distribution range from zero to one.  $X = \langle x_1, \dots, x_n \rangle$  is an observation sequence (the sensor-level events),  $Y = \langle y_1, \dots, y_n \rangle$  is the associated label sequence (the human activity level events),  $f_k$  and  $\lambda_k$  respectively are feature functions and their weights. Feature functions, which can be binary or real valued, are defined on the observations and are used to compute label probabilities. In contrast to Hidden Markov Models [23], CRFs do not assume the feature functions to be mutually independent.

## 4 Motivating Example

Figure 1 shows a simplistic example demonstrating how a process can seem unstructured at the sensor level of events, while being structured at a human behavior level. Petri net 1c shows the process at the human activity level. The *Taking medicine* hu-



**Fig. 1** A human activity level process model (c) where the two transitions themselves are defined as process models (shown in a and b), and the Inductive Miner result on the sensor-level traces generated from this model (d).

man activity level activity is itself defined as a process, which is shown in Figures 1a to 1c. *Eating* is also defined as a process, which is shown in Figure 1b. When we apply the Inductive Miner process discovery algorithm [18] to sensor-level traces generated by the hierarchical process of Figure 1c, we obtain the process model shown in Figure 1d. This process model allows for almost all possible sequences over the alphabet  $\{CD, D, DCC, MC, W\}$ , with the only constraint introduced by the model being that if a *W* occurs, then it has to be preceded by a *DCC* event. Firing of all other transitions in the model can be skipped. Behaviorally this model is very close to the so called "flower" model [1], the model that allows for all behavior over its alphabet. The alternating structure between *Taking medicine* and *Eating* that was present in the human activity level process in Figure 1c cannot be observed in the process model in Figure 1d. This is caused by high variance in start and end events of the sensor-level subprocesses of *Taking medicine* and *Eating* as well as by the overlap in types of activities between these two subprocesses. Both subprocesses contain *DCC*, and the miner cannot see that there are actually two different contexts for the *DCC* activity to split the label in the model. Abstracting the sensor-level events to their respective human activity level events before applying process discovery to the resulting human activity log unveils the alternating structure between *Eating* and *Taking medicine* as shown in Figure 1c.

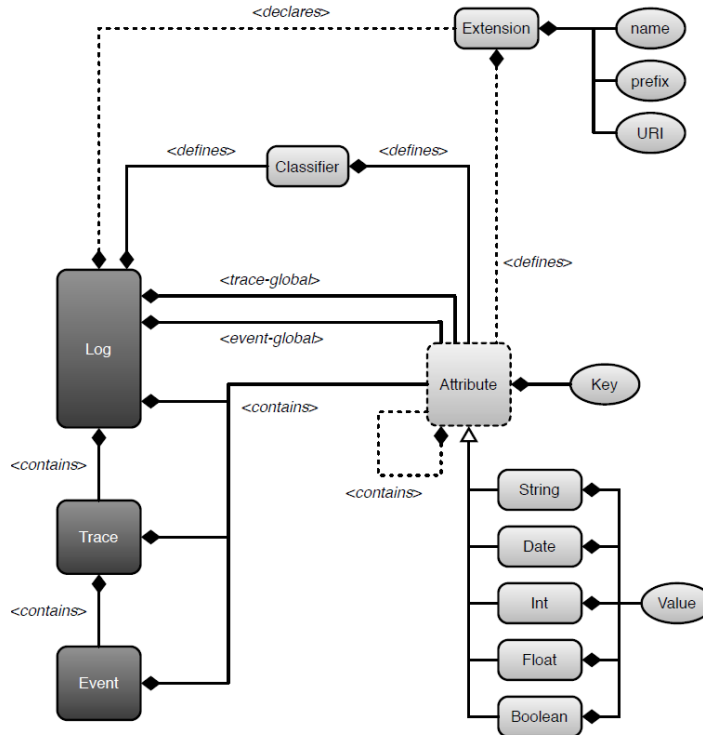


Fig. 2 XES event log meta-model, as defined in [12].

## 5 Event Abstraction as a Sequence Labeling Task

In this section we describe the framework for supervised abstraction of events based on Conditional Random Fields (CRFs). Additionally, we describe feature functions for event logs in a general way by using the IEEE XES standard [12]. XES, which is an abbreviation for *eXtensible Event Stream*, is the IEEE standard for process mining event logs. An overview of the XES file structure which is shown in Figure 2. An event *log* is defined as a set of *traces*, which in itself are a sequences of *events*. The log, traces and events can all contain one or more *attributes*, which consist of a *key* and a *value* of a certain type. Event or trace attributes may be *global*, which indicates that the attribute needs to be defined for each event or trace respectively. A log contains one or more *classifiers*, which can be seen as labeling functions on the events of a log, defined on global event attributes. *Extensions* define a set of attributes on log, trace, or event level, in such a way that the semantics of these attributes are clearly defined. One can view XES extensions as a specification of attributes that events, traces, or event logs themselves frequently contain. XES defines the following standard extensions:



- Concept** Specifies the generally understood name of the event/trace/log (attribute 'Concept:name').
- Lifecycle** Specifies the lifecycle phase (attribute 'Lifecycle:transition') that the event represents in a transactional model of their generating activity. The *Lifecycle* extension also specifies a standard transactional model for activities.
- Organizational** Specifies three attributes for events, which identify the actor having caused the event (attribute 'Organizational:resource'), his role in the organization (attribute 'Organizational:role'), and the group or department within the organization where he is located (attribute 'Organizational:group').
- Time** Specifies the date and time at which an event occurred (attribute 'Time:timestamp').

We introduce a special attribute of type *String* with key *label*, which represents the human activity level activity. The *concept* name of an event is then considered to be a sensor-level name of an event. The *label* attribute specifies the human activity level label for each event individually, allowing for example one sensor-level event of type *Dishes & cups cabinet* to be of human activity level type *Taking medicine*, and another sensor-level event of the same type to be of human level activity type *Eating*. Note that for some traces human level activity annotations might be available, in which case its events contain the *label* attribute, while other traces might not be annotated. Human activity level interpretations of unannotated traces, obtained by inferring the *label* attribute based on information that is present in the annotated traces, allow the use of unannotated traces for process discovery and conformance checking on a human activity level.

Figure 3 provides a conceptual overview of the supervised event abstraction method. The approach takes two inputs: 1) a set of annotated traces, which are traces where the human activity level event that each sensor level event belongs to (the *label* attribute of the sensor-level event) is known, and 2) a set of unannotated traces, which are traces where only the sensor-level events known. Conditional Random Fields (CRFs) are trained of the annotated traces to create a probabilistic mapping from sensor-level events to human activity level events. This mapping, once obtained, can be applied to the unannotated traces in order to estimate the corresponding human activity level event for each sensor-level event (its *label* attribute). Often multiple consecutive sensor-level events will have the same *label* attribute. We assume that multiple human activity level events cannot occur in parallel. This enables us to interpret a sequence of events with identical *label* values as a single human activity level event. To obtain a final human activity level log, we *collapse* sequences of events with the same value for the *label* attribute into two events with this value as *concept* name, where the first event has a *lifecycle* 'start' and the second has the *lifecycle* 'complete'. Table 1 and Table 2 illustrate this collapsing procedure through an input and output event log.

The method described in this section is implemented and available for use as package *AbstractEventsSupervised* in the ProM 6 [35] process mining toolkit and makes use of the GRMM [28] implementation of CRFs.

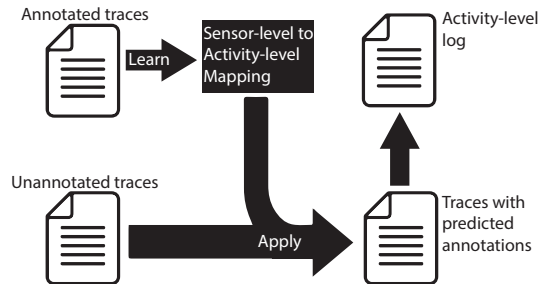
We now show for each XES extension how it can be translated into useful feature functions for event abstraction. Note that we do not limit ourselves to XES logs that

**Table 1** A trace with predicted human activity level annotations (*label*)

| Case | Time:timestamp      | Concept:name          | label           |
|------|---------------------|-----------------------|-----------------|
| 1    | 03/11/2015 08:45:23 | Medicine cabinet      | Taking medicine |
| 1    | 03/11/2015 08:46:11 | Dishes & cups cabinet | Taking medicine |
| 1    | 03/11/2015 08:46:45 | Water                 | Taking medicine |
| 1    | 03/11/2015 08:47:59 | Dishes & cups cabinet | Eating          |
| 1    | 03/11/2015 08:47:89 | Dishwasher            | Eating          |
| 1    | 03/11/2015 17:10:58 | Dishes & cups cabinet | Taking medicine |
| 1    | 03/11/2015 17:10:69 | Medicine cabinet      | Taking medicine |
| 1    | 03/11/2015 17:11:18 | Water                 | Taking medicine |

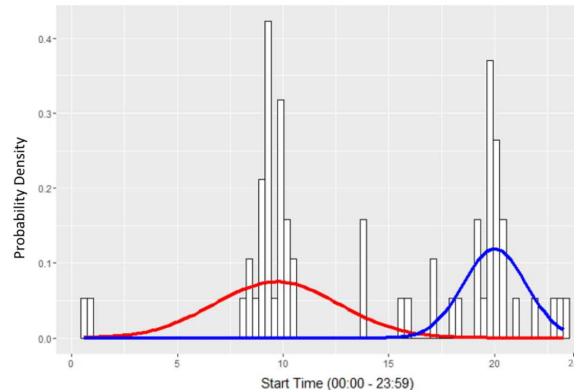
**Table 2** The resulting human activity level log after collapsing the consecutive identical label values of the trace in Table 1.

| Case | Time:timestamp      | Concept:name    | Lifecycle:transition |
|------|---------------------|-----------------|----------------------|
| 1    | 03/11/2015 08:45:23 | Taking medicine | Start                |
| 1    | 03/11/2015 08:46:45 | Taking medicine | Complete             |
| 1    | 03/11/2015 08:47:59 | Eating          | Start                |
| 1    | 03/11/2015 08:47:89 | Eating          | Complete             |
| 1    | 03/11/2015 17:10:58 | Taking medicine | Start                |
| 1    | 03/11/2015 17:11:18 | Taking medicine | Complete             |

**Fig. 3** Conceptual overview of Supervised Event Abstraction.

contain all XES extensions; when a log contains a subset of the extensions, a subset of the feature functions will be available for the supervised learning step. This approach leads to a feature space of unknown size, potentially causing problems related to the curse of dimensionality. To address this we use L1-regularized CRFs. In the training phase we search for values of weight vector  $\lambda$  that minimize the cross entropy between the ground truth target and the predicted label on the training data. L1-regularization adds a  $\lambda$  penalty terms to this minimization function that is proportionate to the size of the weight vector, giving the model an incentive not to use all of the available features (i.e., setting some features to zero weight). This results in prediction models that are sparse and therefore simpler, which helps to prevent overfitting.

**Fig. 4** The histogram representation and a Gaussian Mixture Model fitted to timestamps values of the plates cupboard sensor in the Van Kasteren data set.



### 5.1 From a XES Log to a Feature Space

We now discuss per XES extension how feature functions can be obtained.

**Concept extension** The sensor-level labels (concept names) of the preceding events in a trace can contain useful contextual information for classification into the correct human activity level event type. Based on the  $n$ -gram  $\langle a_1, a_2, \dots, a_n \rangle$  consisting of the sensor-level labels of the  $n$  last-seen events in a trace, we can estimate a categorical probability distribution over the classes of human activity level activities from the training log, such that the probability of class  $l$  is equal to the number of times that the  $n$ -gram was observed while the  $n$ -th event was annotated with class  $l$ , divided by the total number of times that the  $n$ -gram was observed. The CRF model requires feature functions with numerical range. A feature function based on the concept extension has two parameters,  $n$  and  $l$ , and is valued with the estimated categorical probability density of the current sensor-level event having human activity level label  $l$  given the  $n$ -gram with the last  $n$  sensor-level event labels. It can be useful to combine multiple features that are based on the concept extension, where the features have different values for  $n$  and  $l$ .

**Organizational extension** Similar to the concept extension feature functions, categorical probability distributions can be estimated on the training set for  $n$ -grams of *resource*, *role*, or *group* attributes of the last  $n$  events. Likewise, an organizational extension based feature function with three parameters,  $n$ -gram size  $n$ ,  $o \in \{\text{resource}, \text{role}, \text{group}\}$ , and label  $l$ , is valued with the probability density according to the estimated categorical probability distribution of label  $l$  given the  $n$ -gram of the last  $n$  event resources/roles/groups.

**Time extension** In terms of time, there are several potentially existing patterns. A certain type of human activity might for example be concentrated in a certain parts of a day, of a week, or of a month. This concentration can however not be modeled with a single Gaussian distribution, as it might be the case that a type of human activity has high probability to occur in the morning or in the evening, but low probability to occur in the afternoon in-between. A mixture distribution consisting of multiple components is therefore needed to model the probability distri-

bution over timestamps. The most well-known mixture distribution is the Gaussian Mixture Model (GMM), where each component of the mixture is defined by a normal distribution. The circular, non-Euclidean, nature of the data space of time-of-the-day, time-of-the-week, or time-of-the-month however introduces problems for the GMM, as, using time-of-the-day as an example, 00:00 is actually very close to 23:59. Figure 4 illustrates this problem. The Gaussian component with a mean around 10 o'clock has a standard deviation that is much higher than what one would expect when looking at the histogram, as the GMM tries to explain the data points just after midnight with this component. These data points just after midnight would however have been much better explained with the Gaussian component with the mean around 20 o'clock, which is much closer in time. Alternatively, we use a mixture model with components of the von Mises distribution, which is a close approximation of a normal distribution wrapped around the circle. To determine the correct number of components of such a von Mises Mixture Model (VMMM) we use Bayesian Information Criterion (BIC) [27], chooses the number of components which explains the data with the highest likelihood, while adding a penalty for the number of model parameters. A VMMM is estimated on training data, modeling the probabilities of each type of human activity based on the time passed since the start of the day, week or month. A time extension feature function with two parameters,  $t \in \{day, week, month, \dots\}$  and label  $l$ , is valued with the VMMM-estimated probability of label  $l$  given the  $t$  view on the event timestamp. An alternative approach to estimate the probability density on data that lies on a manifold, such as a circle, is described by Cohen and Welling [6].

**Lifecycle extension & Time extension** The XES standard [12] defines several lifecycle stages of process activities, which represent the transactional model of their generating activity. Lifecycle values that are commonly found in real life logs are *start* and *complete* which respectively represent when this activity started and ended. However, a larger set of lifecycle values is defined in the XES standard, including *schedule*, *suspend*, and *resume*. The time differences between different stages of an activity lifecycle can be calculated when an event log possesses both the lifecycle extension and the time extension. For example, when observing the *complete* of an activity, the time between this *complete* and the corresponding *start* of this activity can contain useful information for predicting the correct human activity label. Finding the associated *start* event becomes nontrivial when multiple instances of the same activity are in parallel, as it is then unknown which *complete* event belongs to which *start* event. We assume consecutive lifecycle steps of activities running in parallel to occur in the same order as the preceding lifecycle step. For example, when we observe two *start* events of an activity of type A in a row, followed by two *complete* events of type A, we assume the first *complete* to belong to the first *start*, and the second *complete* to belong to the second *start*.

The XES standard defines an ordering over the lifecycle values. For each type of human activity, we fit a Gaussian Mixture Model (GMM) to the set of time differences between each two consecutive lifecycle steps. A feature based on both the combination of the lifecycle and the time extension with activity label parameter  $l$  and lifecycle  $c$  is valued the probability density of activity  $l$  as estimated by the

GMM given the time between the current event and lifecycle value  $c$ . Bayesian Information Criterion (BIC) [27] is again used to determine the number of components of the GMM. Note that while these features are time-based, regular GMMs can be used instead of VMMs since time duration is a Euclidean, non-circular, space.

## 5.2 Evaluating Human Activity Level Event Predictions for Process Mining Applications

Existing approaches in the field of activity recognition take as input time windows where each time window is represented by a feature vector that describes the sensor activity or status during that time window. Hence, evaluation methods in the activity recognition field are window-based, using evaluation metrics like the percentage of correctly classified time slices [30, 13, 14]. There are two reasons to deviate from this evaluation methodology in a process mining setting. First, our method operates on events instead of time windows. Second, the accuracy of the resulting high level sequences is much more important for many process mining techniques (e.g. process discovery, conformance checking) than the accuracy of predicting each individual minute of the day.

A well-known metric for the distance of two sequences is the Levenshtein distance [19]. However, Levenshtein distance is not suitable to compare sequences of human actions, as human behavior sometimes includes branches in which it does not matter in which order two activities are performed. For example, most people *shower* and *have breakfast* after waking up, but people do not necessarily always perform the two in the same order. Indeed, when  $\langle a, b \rangle$  is the sequence of predicted human activities, and  $\langle b, a \rangle$  is the actual sequence of human activities, we consider this to be only a minor error, since it is often not relevant in which order two parallel activities are executed. Levenshtein distance would assign a cost of 2 to this abstraction, as transforming the predicted sequence into the ground truth sequence would require one deletion and one insertion operation. For example, most people *shower* and *have breakfast* after waking up, but people do not necessarily always perform the two in the same order. An evaluation measure that better reflects the prediction quality of event abstraction is the Damerau-Levenshtein distance [8], which adds a swapping operation to the set of operations used by Levenshtein distance. Damerau-Levenshtein distance would assign a cost of 1 to transform  $\langle a, b \rangle$  into  $\langle b, a \rangle$ . To obtain comparable numbers for different numbers of predicted events we normalize the Damerau-Levenshtein distance by the maximum of the length of the ground truth trace and the length of the predicted trace and subtract the normalized Damerau-Levenshtein distance from 1 to obtain Damerau-Levenshtein Similarity (DLS).

## 6 Case Studies

In this section we evaluate the supervised event abstraction framework on three case studies on real life smart home data sets.

### 6.1 Experimental setup

We include three real life smart home event logs in the evaluation: the Van Kasteren event log [14], and two event logs from a smart home experiment conducted by MIT [30]. All three event logs used in for the evaluation consist of multidimensional time series data with all dimensions binary, where each binary dimension represents the state of one in-home sensor. These sensors include motion sensors, open/close sensors, and power sensors (discretized to 0/1 states). We transform the multidimensional time series data from sensors into events by regarding each sensor change point as an event. Cases are created by grouping events together that occurred in the same day, with a cut-off point at midnight. High-level labels are provided for the event logs.

The following XES extensions can be used for these event logs:

**Concept** The sensor that generated the event.

**Time** The time stamp of the sensor change point.

**Lifecycle** *Start* when the event represents a sensor value change from 0 to 1 and *Complete* when it represents a sensor value change from 1 to 0.

Note that human activity level annotations are provided for all traces in the three event logs. To evaluate how well the supervised event abstraction method generalized to unannotated traces, we artificially use a part of the traces to train the abstraction model and apply them on a test set where we regard the annotations to be non-existent. We evaluate the obtained human activity labels against the ground truth labels in a Leave-One-Trace-Out-Cross-Validation setup where we iteratively leave out one trace to evaluate how well this mapping generalizes to unseen events and cases. We measure the accuracy of the human activity level traces compared to the ground truth human activity level traces in terms of Damerau-Levenshtein similarity [8].

Additionally, we evaluate the quality of the process model that can be discovered from the human activity level traces. To discover a process model from the human activity level event log we use the Inductive Miner [18]. There are several criteria to express the fit between a process model and an event log in the area of process mining. Two of those criteria are *fitness* [26], which measures the degree to which the behavior that is observed in the event log can be replayed on the process model, and *precision* [21], which measures the degree to which the behavior that was never observed in the event log cannot be replayed on the process model. Low precision typically indicates an overly general process model, that allows for too

much behavior. We compare the *fitness* and *precision* of the models produced by the Inductive Miner on the sensor-level log and the human activity level log.

## 6.2 Case Study 1: Van Kasteren Event Log

For the first case study we use the smart home environment log described in Van Kasteren et al. [14]. The Van Kasteren log contains 1285 events divided over fourteen different sensors. The log contains 23 days of data.

The average Damerau-Levenshtein similarity between the predicted human activity level traces in the Leave-One-Trace-Out-Cross-Validation experimental setup and the ground truth human activity level traces is 0.7516, which shows that the supervised event abstraction method produces traces which are fairly similar to the ground truth.

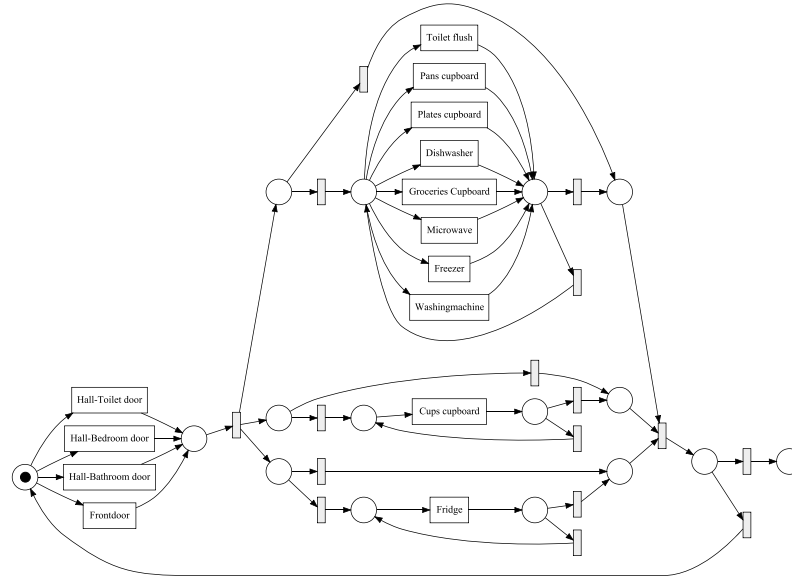
Figure 5 shows the result of the Inductive Miner [18] for the sensor-level events in the Van Kasteren data set. The resulting process model starts with a choice between four activities: *hall-toilet door*, *hall-bedroom door*, *hall-bathroom door*, and *frontdoor*. After this choice the model branches into three parallel blocks, where the upper block consists of a large choice between eight different activities. The other two parallel blocks respectively contain a loop of the *cups cupboard* and the *fridge*. This model closely resembles the flower model, which allows for all behavior in any arbitrary order. There seems to be very little structure on this sensor level of event granularity.

Figure 6 shows the result of the Inductive Miner on the aggregated set of predicted test traces. We can see that the main daily routine starts with *breakfast*, after which the subject *leaves the house* to go to work. After work the subject *prepares dinner* and *goes to bed*. The activities *use toilet* and *take shower* are put in parallel to this sequence of activities, indicating that they occur at different places in the sequences of activities.

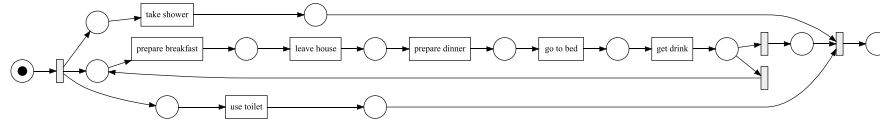
Table 3 shows the effect of the abstraction on the fitness and precision of the models discovered by the Inductive Miner. It shows that the precision of the model discovered on the abstracted log is much higher than the precision of the model discovered on the sensor data, indicating that the abstraction helps discovering a model that is more behaviorally constrained and more specific.

## 6.3 Case Study 2: MIT Household A Event Log

For the second case study we use the data of *household A* of a smart home experiment conducted by MIT [30]. Household A contains data of 16 days of living, 2701 sensor-level events registered by 26 different sensors. The human level activities are provided in the form of a taxonomy of activities on three levels, called *heading*, *category* and *subcategory*. On the *heading* level the human activities are very general in



**Fig. 5** Inductive Miner result on the sensor-level events of the Van Kasteren event log.



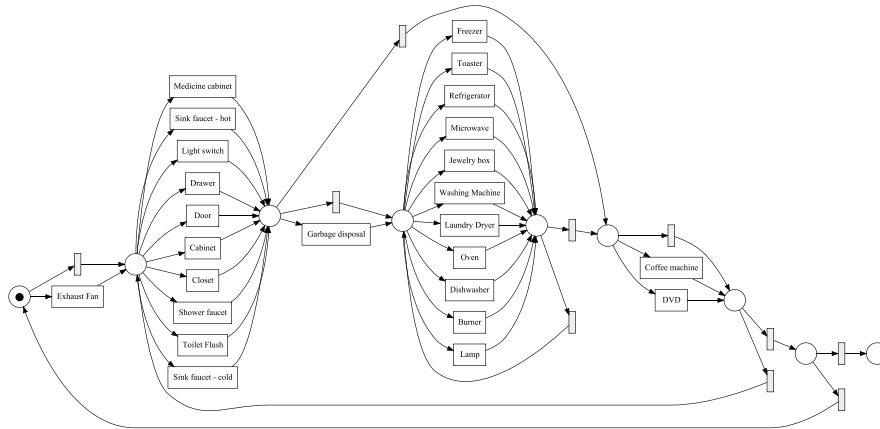
**Fig. 6** Inductive Miner result on the human activity level events discovered from the Van Kasteren event log.

**Table 3** Effect of abstraction on fitness and precision of the process model discovered by the Inductive Miner.

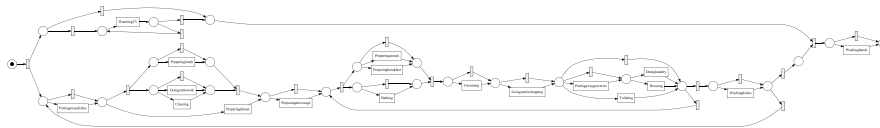
| Event log       | Abstraction     | Fitness | precision |
|-----------------|-----------------|---------|-----------|
| Van Kasteren    | No (Figure 5)   | 0.9111  | 0.3308    |
| Van Kasteren    | Yes (Figure 6)  | 0.7918  | 0.7804    |
| MIT household A | No (Figure 7)   | 0.9916  | 0.2289    |
| MIT household A | Yes (Figure 8)  | 0.9880  | 0.3711    |
| MIT household B | No (Figure 9)   | 1.0     | 0.2389    |
| MIT household B | Yes (Figure 10) | 0.9305  | 0.4319    |

nature, such as the activity *personal needs*. The eight different activities on the *heading* level branch into 19 different activities on the *category* level, where *personal needs* branches into e.g. *eating*, *sleeping*, and *personal hygiene*. The 19 *categories* are divided over 34 *subcategories*, which contain very specific human activities. At the *subcategory* level the *category meal cleanup* is for example divided into *washing dishes* and *putting away dishes*. At the *subcategory* level there are more types of





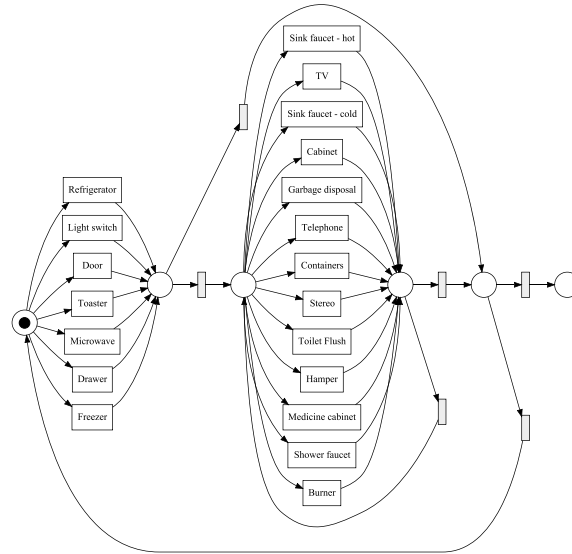
**Fig. 7** Inductive Miner result on the sensor-level MIT household A event log.



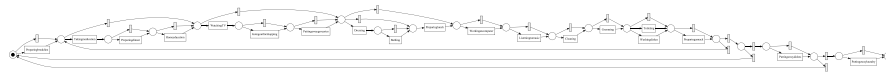
**Fig. 8** Inductive Miner result on the discovered human activity level events on the MIT household A log.

human activities than there are sensors-level activities, which makes the abstraction task very hard. Therefore, we set the target label to the *category* level.

Figure 7 shows the model discovered with the Inductive Miner on the sensor-level events of the MIT household A log. The model obtained allows for too much behavior, as it contains two large choice blocks. We found a Damerau-Levenshtein similarity of 0.6348 in the Leave-One-Trace-Out-Cross-Validation experiment. Note that the abstraction accuracy on this log is lower than the abstraction accuracy on the Van Kasteren event log. However, the MIT household A log contains more different types of human activity, resulting in a more difficult prediction task with a higher number of possible target classes. Figure 10 shows the process model discovered from the human activity level traces that we predicted from the sensor-level events. Even though the model is too large to print in a readable way, from its shape it is clear that the abstracted model is much more behaviorally constrained than the sensor-level model. The precision and fitness values in Table 3 show that indeed the process model after abstraction has become behaviorally more specific while the portion of behavior of the data that fits the process model remains more or less the same.



**Fig. 9** Inductive Miner result on the sensor-level MIT household B event log.



**Fig. 10** Inductive Miner result on the discovered human activity level events on the MIT household B log

### 6.4 Case Study 3: MIT Household B Event Log

For the third case study we use the data of *household B* of the MIT smart home experiment [30]. Household B contains data of 17 days of living, 1962 sensor-level events registered by 20 different sensors. Identically to MIT household A the human-level activities are provided as a three-level taxonomy. Again, we use the *subcategory* level of this taxonomy as target activity label.

The model discovered with the Inductive Miner [18] from the sensor-level events is shown in Figure 9. The model obtained allows for too much behavior, as it contains two large choice blocks. We found a Damerau-Levenshtein similarity of 0.5865 in the Leave-One-Trace-Out-Cross-Validation experiment, which is lower than the similarity found on the MIT A data set while the target classes of the abstraction are the same for the two data sets. This can be explained by the fact that there is less training data for this event log, as household B contains 1932 sensor-level events where household A contains 2701 sensor-level events. Figure 10 shows the process model discovered from abstracted log. Again this model is not readable due to its size, but its shape shows it to be behaviorally quite specific. The precision and fitness values in Table 3 also that process model after abstraction has indeed

become behaviorally more specific while the portion of behavior of the data that fits the process model decreased only slightly.

## 7 Conclusion

In this paper we presented a framework to abstract events using supervised learning which has been implemented in the ProM process mining toolkit. An important part of the framework is a generic way to extract useful features for abstraction from the extensions defined in the XES IEEE standard for event logs. We propose the Damerau-Levenshtein Similarity for evaluation of the abstraction results, and motivate why it fits the application of process mining. Finally, we showed on three real life smart home data sets that application of the supervised event abstraction framework enables us to mine more precise process model description of human life compared to what could be mined from the original data on the sensor-level. Additionally, these process models contain interpretable event labels on the human behavior activity level.

## References

- [1] van der Aalst WMP (2016) *Process Mining: Data Science in Action*. Springer
- [2] van der Aalst WMP, Weijters T, Maruster L (2004) Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 16(9):1128–1142
- [3] Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data. In: *Pervasive Computing, LNCS, Springer*, pp 1–17
- [4] Bose RPJC, van der Aalst WMP (2009) Abstractions in process mining: A taxonomy of patterns. In: *Business Process Management, LNCS, Springer*, pp 159–175
- [5] Chen L, Nugent C (2009) Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems* 5(4):410–430
- [6] Cohen T, Welling M (2015) Harmonic exponential families on manifolds. In: *Proceedings of The 32nd International Conference on Machine Learning, JMLR Workshop & Conference Proceedings*, pp 1757–1765
- [7] Conforti R, Dumas M, García-Bañuelos L, La Rosa M (2016) BPMN miner: Automated discovery of bpmn process models with hierarchical structure. *Information Systems* 56:284–303
- [8] Damerau FJ (1964) A technique for computer detection and correction of spelling errors. *Communications of the ACM* 7(3):171–176

- [9] van Dongen BF, Adriansyah A (2010) Process mining: Fuzzy clustering and performance visualization. In: Business Process Management Workshops, Springer, LNBIP, pp 158–169
- [10] Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. *Machine Learning* 29(2-3):131–163
- [11] Günther CW, Rozinat A, van der Aalst WMP (2010) Activity mining by global trace segmentation. In: Business Process Management Workshops, Springer, LNBIP, pp 128–139
- [12] IEEE (2016) IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016 pp 1–50, DOI 10.1109/IEEESTD.2016.7740858
- [13] van Kasteren T, Kröse B (2007) Bayesian activity recognition in residence for elders. In: Proceedings of the 3rd IET International Conference on Intelligent Environments, IEEE, pp 209–212
- [14] van Kasteren T, Noulas A, Englebienne G, Kröse B (2008) Accurate activity recognition in a home setting. In: Proceedings of the 10th International Conference on Ubiquitous Computing, ACM, pp 1–9
- [15] Kim E, Helal S, Cook D (2010) Human activity recognition and pattern discovery. *Pervasive Computing* 9(1):48–53
- [16] Kwapisz JR, Weiss GM, Moore SA (2011) Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter* 12(2):74–82
- [17] Lafferty J, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning, Morgan Kaufmann
- [18] Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs - a constructive approach. In: Application and Theory of Petri Nets and Concurrency, LNCS, Springer, pp 311–329
- [19] Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10:707–710
- [20] Mannhardt F, de Leoni M, Reijers HA, van der Aalst WMP, Toussaint PJ (2016) From low-level events to activities - a pattern-based approach. In: International Conference on Business Process Management, Springer, pp 125–141
- [21] Munoz-Gama J, Carmona J (2010) A fresh look at precision in process conformance. In: International Conference on Business Process Management, Springer, pp 211–226
- [22] Poppe R (2010) A survey on vision-based human action recognition. *Image and Vision Computing* 28(6):976–990
- [23] Rabiner LR, Juang BH (1986) An introduction to hidden Markov models. *ASSP Magazine* 3(1):4–16
- [24] Reisig W (2012) Petri nets: an introduction, vol 4. Springer Science & Business Media
- [25] Riboni D, Bettini C (2011) OWL 2 modeling and reasoning with complex human activities. *Pervasive and Mobile Computing* 7(3):379–395

- [26] Rozinat A, van der Aalst WMP (2008) Conformance checking of processes based on monitoring real behavior. *Information Systems* 33(1):64–95
- [27] Schwarz G (1978) Estimating the dimension of a model. *The Annals of Statistics* 6(2):461–464
- [28] Sutton C (2006) GRMM: Graphical models in Mallet. Implementation available at <http://mallet.cs.umass.edu/grmm>
- [29] Sztyley T, Carmona J, Völker J, Stuckenschmidt H (2016) Self-tracking reloaded: applying process mining to personalized health care from labeled sensor data. In: *Transactions on Petri Nets and Other Models of Concurrency XI*, Springer Berlin Heidelberg, pp 160–180
- [30] Tapia EM, Intille SS, Larson K (2004) Activity recognition in the home using simple and ubiquitous sensors. In: *Pervasive Computing, LNCS*, Springer, pp 158–175
- [31] Tax N, Alasgarov E, Sidorova N, Haakma R (2016) On generation of time-based label refinements. In: *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*, CEUR-WS.org, pp 25–36
- [32] Tax N, Sidorova N, van der Aalst WMP, Haakma R (2016) Heuristic approaches for generating local process models through log projections. In: *Proceedings of IEEE Symposium on Computational Intelligence and Data Mining*, To appear.
- [33] Tax N, Sidorova N, Haakma R, van der Aalst WMP (2016) Event abstraction for process mining using supervised learning techniques. In: *Proceedings of the SAI Intelligent Systems Conference*, IEEE, pp 161–170
- [34] Tax N, Sidorova N, Haakma R, van der Aalst WMP (2016) Log-based evaluation of label splits for process models. *Procedia Computer Science* 96:63–72
- [35] Verbeek HMW, Buijs JCAM, Van Dongen BF, van der Aalst WMP (2010) ProM 6: The process mining toolkit. In: *Proceedings of the Business Process Management Demonstration Track*, CEUR-WS.org, pp 34–39
- [36] Weijters AJMM, Ribeiro JTS (2011) Flexible heuristics miner (FHM). In: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*, IEEE, pp 310–317
- [37] van Zelst SJ, van Dongen BF, van der Aalst WMP (2015) Avoiding over-fitting in ILP-based process discovery. In: *International Conference on Business Process Management*, Springer International Publishing, pp 163–171