From Good Intentions to Behaviour Change

Probabilistic Feature Diagrams for Behaviour Support Agents

Malte S. Kließ[1], Marielle Stoelinga[2], and M. Birna van Riemsdijk[1,2(✉)]

[1] Delft University of Technology, Delft, The Netherlands
m.s.kliess@tudelft.nl
[2] University of Twente, Enschede, The Netherlands
{m.i.a.stoelinga,m.b.vanriemsdijk}@utwente.nl

Abstract. Behaviour support technology assists people in organising their daily activities and changing their behaviour. A fundamental notion underlying such supportive technology is that of *compliance* with behavioural norms: do people indeed perform the desired behaviour? Existing technology employs a rigid implementation of compliance: a norm is either satisfied or not. In practice however, behaviour change norms are less strict: E.g., is a new norm to do sports at least three times a week complied with if it is occasionally only done twice a week? To address this, in this paper we formally specify probabilistic norms through a variant of *feature diagrams*, enabling a hierarchical decomposition of the desired behaviour and its execution frequencies. Further, we define a new notion of *probabilistic norm compliance* using a formal *hypothesis testing* framework. We show that probabilistic norm compliance can be used in a real-world setting by implementing and evaluating our semantics with respect to an existing daily behaviour dataset.

1 Introduction

Behaviour support technology [7] is aimed at assisting people in organising their daily activities and changing their behaviour, for example to adopt a healthier lifestyle. While numerous behaviour support frameworks have been developed, they typically focus on a specific domain or type of behaviour, such as monitoring our diet, emergency monitoring, or forgetting to perform certain tasks [12]. In our work we aim to develop a *generic* framework for representing and reasoning about people's (desired) daily behaviour in order to allow an electronic partner (epartner for short) to provide personalised behaviour support [19]. A generic framework facilitates application across domains, and development of expressive representation and reasoning techniques in a principled way.

This work is partially financed by the Netherlands Organisation for Scientific Research (NWO) under the research programmes CoreSAEP (639.022.416), SEQUOIA (15474) and StepUp (628.010.006), as well as by the EU under the project 102112 SUCCESS.

M. Baldoni et al. (Eds.): PRIMA 2019, LNAI 11873, pp. 354–369, 2019.
https://doi.org/10.1007/978-3-030-33792-6_22

A central task an epartner needs to be able to do in order to provide personalised behaviour support, is determine whether the user is *complying* with the desired target behaviour. The challenge we address in this paper is to formally define the fundamental components that are (at least) required for an epartner to perform this task, namely:

1. a description of the desired user behaviour, which can be self-reported, prescribed by a caregiver, or otherwise recorded;
2. a record of the actual daily routine or behaviour of the user;
3. a measure of compliance of what is actually being done to what is expected/desired to be done.

Inspired by research on normative multiagent systems [2], we refer to expressions of desired user behaviour as *(behaviour) norms* that may or may not be complied with by the user of the epartner.

Providing a comprehensive formal framework for representation of daily user behaviour for the purpose of behaviour support is a non-trivial task due to the potential complexity of this behaviour and the many facets that may be considered, such as temporal aspects [6] and user values [17]. In this paper we focus on two key characteristics. First, representing the potentially complex *structure of daily behaviour* requires a way to decompose behaviour into its constituting parts [18]. Not all of these parts need to always be executed, some are optional while others are mandatory, and sometimes a choice needs to be made. Second, the nature of (desired) daily behaviour is often *habitual* [5,9], i.e., it concerns the frequency of a user's repeated behaviour over time. For example, a user may want to change his habit of having a late breakfast such that at least 80% of his breakfasts are early breakfasts, and needs to do work in the evening on four out of five workdays, i.e., 80% of the workdays.

Defining when a user's behaviour is compliant with such a specification of desired behaviour requires first of all a definition of compliance with respect to the basic specified behaviour structure. Second, in order to define compliance with respect to the frequency of performed behaviour, we propose a statistical approach (*probabilistic norm compliance*). This is because there will typically be some variation in user behaviour over time, which might lead to some deviations from the precise desired behaviour frequencies. The question we need to answer is when these deviations are still "ok", i.e., when we can consider the user to have adopted the specified habit. For example, if we consider the past 20 workdays out of which the user has worked 17 evenings. Is the user compliant with the behaviour norm, i.e, can we say the user has adopted the specified habit? What if we consider the past 6 days out of which the user has worked 5 evenings?

To address these challenges, this paper provides the following contributions:

1. We propose to use the well-studied formalism of *Feature Diagrams* [11] for daily behaviour representation. Feature Diagrams have been used widely in software engineering for modelling Software Product Lines [15]. In that context Feature Diagrams represent the different parts of a software product, and how they fit together to compose the overarching concept or final product.

We observe that this formalism also provides a natural way of representing the hierarchical structure of daily behaviour. (Sections 2 and 3)
2. We introduce a novel extension of Feature Diagrams called probabilistic Feature Diagrams in order to represent behaviour frequencies. We provide a formal semantics by means of hypothesis testing [10]. Hypothesis testing is a type of statistical model checking, normally used to verify performance characteristics of software. (Section 4)
3. We perform an experimental evaluation of our framework by implementing the Feature Diagram semantics with respect to an existing daily behaviour dataset [14], and show that our notion of probabilistic norm compliance can be used in this real-world setting. (Section 5)

2 Behaviour Hierarchies

Psychological research [18] has shown that people think about their behaviour in a hierarchical fashion, from abstract to more concrete. We have proposed to formalise hierarchical behaviour structures with the aim of allowing a behaviour support agent to represent the (actual and desired) daily behaviour of its user in a way that matches how the user thinks about their behaviour [6,8,9,17].

Behaviour hierarchies can be represented as trees, with an abstract behaviour as the root, and its nodes and leaves decomposing this behaviour into its more concrete parts and sub-behaviours. The leaves will consist of behaviours that do not need to be decomposed further. This type of decomposition is comparable to Goal Plan Trees (GPTs) [16]. The main difference is in the semantics: our structures are used to *describe* desired behaviour, yielding a logic-based semantics to assess whether a structure is satisfied with respect to a user's actual behaviour. In contrast, GPTs are used to *generate* (software) agent behaviour. Furthermore, our hierarchies should include relative frequencies to indicate how often a sub-behaviour should be performed with respect to its parent behaviour.

Example 1. Suppose that a user, let us call him John, decides that they want to change their daily routine at home: John has realized that he very often has a late breakfast and thus starts his workday rather late as well. As part of improving his daily routine for a workday, he commits to having an early breakfast most of the time (at least 80%) – on the days that he has time for breakfast at home. To help him achieve this, John also commits to do some work at home in the evenings on most days (4 days per week, i.e., 80%), so he can have a breakfast at home as well as get the work done he committed to for his job. John also needs to take some prescribed medication several times per week (3–4 days, i.e., 60–80%). This desired behaviour can be represented as follows:

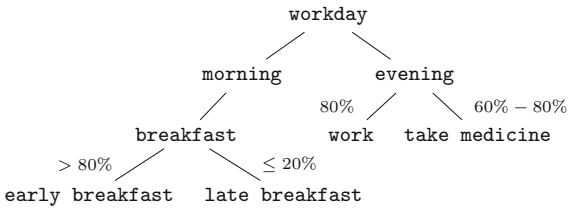

Fig. 1. Tree representation of Example 1

Note that this tree representation is lacking some information about the behaviour structure: namely, the sub-activities `early breakfast` and `late breakfast` form two *alternatives* of the activity `breakfast` (i.e., they cannot both occur on the same workday), while the sub-activities `work` and `take medicine` form two *options* of activities that may be done during the evening. Since for the evening activities, the rates of 80% and 60–80%, respectively, do not sum to (less than) 100%, it is easy to spot that there may be a different characterisation to these sub-activities as compared to the breakfast activities. However, should John plan to take medicine in the evening only once a week, changing the 60–80% to 20%, we need to express explicitly how these sub-behaviours should be interpreted. Similarly, we need to indicate whether an activity is supposed to be carried out as an *optional* or *mandatory* activity. For instance, John might occasionally skip eating breakfast at home in the morning, and take it to work, making this an *optional* activity.

In order to express such structural properties, one needs a more expressive syntactical framework, for which we propose to use Feature Diagrams.

3 Representing Daily Behaviour with Feature Diagrams

In previous work we have already proposed to formalise hierarchical behaviour structures for behaviour support agents [6,8,9,17]. Most of these works however do not provide formal semantics that expresses when such a structure is satisfied, or they do so for a structure with limited expressivity. Through our insight that the well-studied Feature Diagram formalism is suitable for representing behaviour hierarchies, in this paper we are able to propose both an expressive representation framework (Sect. 3.1) as well as an accompanying formal semantics (Sect. 3.2). In this section we provide a definition of Feature Diagrams that represents the behaviour structure. In the next section we add frequencies.

3.1 Syntax

The formal definition of a Feature Diagram we use here is based on Definition 3.2 of [11] and definitions for node types provided in [13]. In particular, the Feature Diagrams we present here are trees; nodes in our Feature Diagram represent (parts of) behaviours. Each node has a type associated with them from the set

$NT = \{\texttt{or}, \texttt{xor}, \texttt{option}\}$. The `optional` node type has edges that can either be of `mandatory` or `optional` type. That is, the decomposition of behaviour into parts is such that the sub-behaviours are either all independent parts of the behaviour (mandatory or optional), or they are options, possibly mutually exclusive ones (`xor`). The `xor` node is also referred to as an `alternative` node in the literature.

In Example 1, `early breakfast` is mutually exclusive with `late breakfast`, but taking them to be optional sub-nodes of the node `breakfast` would still allow both of them to be present. In this situation, we say that `breakfast` is of node type `xor`, meaning that precisely one of the sub-nodes are to be realised. Similarly, the node type `or` allows for at least one of the sub-nodes to be realised. For instance, taking `evening` in the example above to be of type `or` would mean that on any given evening of the workweek, John either takes medicine, or does some work, or both, but there is never (supposed to be) an evening on which he does not do either of these. Taking `workday` as an `option` node with `mandatory` links specifies that any workday requires something to be done in the morning and in the evening.

We formally define Feature Diagrams as follows, using a standard definition of the notion of a tree:

Definition 1 (Tree). *Let N be a set of behaviours and $E : N \times N$ a relation on N. We say that $\langle N, E\rangle$ is a* tree*, if E is antisymmetric, irreflexive and such that for any a, b, $c \in N$, if $(a, b) \in E$ and $(c, b) \in E$, then $a = c$. We use r to denote the root of a tree, i.e., the node $m \in N$ such that there is no $n \in N$ with $(n, m) \in E$. There can be precisely one such root node in any tree.*

Definition 2 (Feature Diagram, FD). *A Feature Diagram D is a structure $D = (N, E, \lambda, \mu)$ such that*

- *N is the set of* nodes*;*
- *$E \subseteq N \times N$ is the set of* decomposition edges *and $N^* \subseteq N$ is the set of nodes that are not leaves, i.e. $\forall n \in N^* \, \exists m \in N \, (n, m) \in E$;*
- *$\langle N, E\rangle$ is a tree;*
- *$\lambda : N^* \rightarrow NT$ is a labelling of the nodes, where $NT = \{\texttt{or}, \texttt{xor}, \texttt{option}\}$ is the set of* node types*;*
- *Let $N^{opt} \subseteq N$ be the set of nodes with label `option`, i.e., $\{n \mid n \in N, \lambda(n) = \texttt{option}\}$, and E^{opt} be the set of edges emerging from these nodes, i.e., $\{(n, m) \mid (n, m) \in E, n \in N^{opt}\}$. Then $\mu : E^{opt} \rightarrow \{\texttt{mandatory}, \texttt{optional}\}$ is a labelling of these edges.*

For a given node n, we will write $n \in D$ as shorthand for $n \in N$.

Usually, the formal Feature Diagrams are provided in graphical form; the relationships `or`, `xor`/`alternative`, `optional`, and `mandatory` are expressed using the following graphical representation:

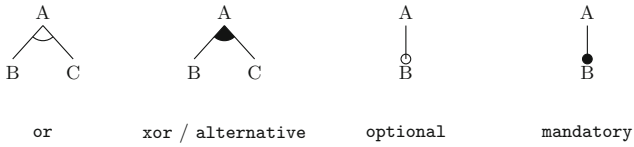

Fig. 2. Common representation of feature diagrams

Example 2. On th basis of the tree given in Example 1, we define the following Feature Diagram for the workday of John:

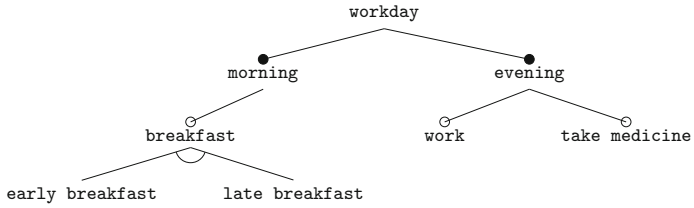

Fig. 3. Feature diagram representing the workday from Example 1

Note that the definition of Feature Diagrams as provided by Definition 2 does not allow for the representation of frequencies of Example 1, and thus we omit them here.

3.2 Semantics

The behavioural norms represented by Feature Diagrams are the ideal that the actual behaviour of the user will be compared to. We will therefore need to introduce what we mean by an *observation* or *model* of behavioural norms represented by a Feature Diagram. We can limit the observation to those behaviours that have a corresponding node in the Feature Diagram: behaviours that do not get mentioned in the diagram can be considered irrelevant for the question of whether a norm is complied with – any behaviour that is relevant for norm compliance should be recorded in the Feature Diagram right from the start.

Definition 3. *[Model/valid model] Let $D = (N, E, \lambda, \mu)$ be a Feature Diagram. A* model *of D is a subset $M \subseteq N$ of the nodes of D.*

A valid model *is a subset $M \subseteq N$ such that*

- *the root $r \in M$;*
- *if $n \in M$ and $\lambda(n) =$ `or`, then for at least one $m \in N$ with $(n, m) \in E$, $m \in M$;*
- *if $n \in M$ and $\lambda(n) =$ `xor`, then for precisely one $m \in N$ with $(n, m) \in E$, $m \in M$;*

- *if* $n \in M$ *and* $\lambda(n) =$ `option`*, then for all* $m \in N$ *with* $(n, m) \in E$ *and* $\mu((n, m)) =$ `mandatory`*,* $m \in M$*;*
- *if* $m \in M$ *with* $m \neq r$*, then also* $n \in M$ *for the unique* n *with* $(n, m) \in E$*.*

We will write $M \models D$ *to indicate that* M *is a valid model of* D*.*

We have omitted mentioning the `optional` edge type, since optional nodes need not be realized. Furthermore, the last point closes the model under predecessors in the tree: it guarantees that, e.g., if `late breakfast` as a subnode of `breakfast`, which in turn is a subnode of `morning` is present in a given model, then `breakfast` and `morning` are both guaranteed to be present.

A model of a Feature Diagram expresses the satisfaction of behaviour norms for a single instance of the behaviour represented by the tree. To formalise realisation of daily routines, i.e., satisfaction of Feature Diagrams over time, we introduce *traces* of models: each point in the trace will represent a single instance of user behaviour. E.g., if the behavioural norm the user wants assistance with is having early breakfasts on workdays, then a trace will consist of a sequence of valid models of the Feature Diagram representing this routine, one for each workday of the week. Note that the root of the Feature Diagram will be present in each sequent of the trace. Thus if the Feature Diagram represents a routine that is not done every day, we can either introduce a new root representing the day, which then occurs in every sequent of the trace – sometimes without any other element – or the sequents of the trace represent only those days on which the routine is – at least partially – executed in accordance with the specified Feature Diagram.

Definition 4 (Trace). *Let* D *be a Feature Diagram and let* σ_i *for* $i \in \mathbb{N}$ *be models of* D*. A* trace *on* D *is a sequence* $\vec{\sigma} = \langle \sigma_0, \sigma_1, \ldots, \sigma_n, \ldots \rangle$*. A trace* $\vec{\sigma}$ *on* D *satisfies* D *if for each* $i \in \mathbb{N}$*,* $\sigma_i \models D$*.*

There is an implied temporal ordering in this notion of trace: viewing the trace as a recording of observed behaviour, one can see the first sequent as the earliest observation, etc. Although we do not explicitly associate each index with a specific time, in most cases of monitoring daily behaviour it will be convenient to assume that each index stands for a specific day. Furthermore, we take traces in the formal definition to be countably infinite. In all practicality, we will then only be dealing with finite initial parts of traces. However, since we do not want to specify a maximal length, nor limit the number of times a specific behaviour can be recorded, we opt for $\mathbb{N}$ as the index set.

4 Probabilistic Feature Diagrams

The next step is to add frequencies into the Feature Diagrams. We do this by extending Definition 2 by a corresponding new component (Sect. 4.1). Then we define the semantics of these probabilistic Feature Diagrams through hypothesis testing (Sect. 4.2) by providing our new notion of probabilistic norm compliance (Sect. 4.3).

4.1 Syntax

Frequencies apply to edges of a Feature Diagram individually. An edge (n, m) with frequency p represents the norm to execute the behaviour represented by m with frequency p, relative to behaviour n. Frequencies may not only be seen as a point $p \in [0, 1]$, but could also refer to an interval in $[0, 1]$, e.g. $(1/2, 1]$ or $[1/3, 2/3]$, representing that the corresponding behaviour should be performed within this range. Edges are not required to have a frequency attached.

Definition 5 (probabilistic Feature Diagram, pFD). *Let $D = (N, E, \lambda, \mu)$ be a Feature Diagram. Let $freq : E \to I([0, 1])$ be a partial function assigning (relative) frequency intervals to edges in E.*

A probabilistic Feature Diagram *$\mathcal{D} = \langle D, freq \rangle$ then is a Feature Diagram with the additional frequencies on the edges given by the function freq.*

In case that q is either a singleton $[p, p]$, or of the form $[0, p)$, $(p, 1]$ (or their corresponding closed variants), we will simply denote these as p, $< p$, $> p$ (resp., $\leq p$, $\geq p$).

We impose a number of restrictions on frequencies, to avoid introducing contradictory information into the Feature Diagram. In particular, for `xor` nodes, we need to impose the restriction that the lower bounds of frequency intervals of its children add up to at most 1, and the upper bounds add up to 1. We can see frequencies as a normalised measure on the subnodes, relative to that node. The children of an `xor` node can be seen as a disjoint partition of the node, and thus frequencies summing up to some value larger than 1 would contradict this partition of the node. Since the frequency of the subnodes of some `xor` node are recording relative occurrence of the subnodes, having this restriction on the upper bounds guarantees that precisely one subnode will be done whenever the parent node is done. Furthermore, we do not allow a frequency $[0, 0]$ to be specified for an edge. This would indicate that the corresponding sub-behaviour should never be executed, which could contradict what are considered valid models according to Definition 3: a valid model might include the behaviour m of an edge (n, m), while adding a frequency $[0, 0]$ to this edge would express the contradictory information that this model is actually invalid. Third, we require that `mandatory` edges have frequency $[1, 1]$, as any other frequency would be contradicting the mandatory nature of the edge. We call probilistic Feature Diagrams that adhere to these restrictions *well-formed*.

Definition 6 (Well-formed probabilistic Feature Diagram, wpFD). *Let $\mathcal{D} = \langle D, freq \rangle$ with $D = (N, E, \lambda, \mu)$ be a probabilistic Feature Diagram. We say that $\mathcal{D}$ is a* well-formed probabilistic Feature Diagram *iff it satisfies the following constraints:*

- *if $\lambda(n) =$ `xor` for some $n \in D$, then $\sum_{(n,m)\in E} \inf freq((n, m)) \leq 1$ and $\sum_{(n,m)\in E} \sup freq((n, m)) = 1$;*[1]

[1] Note that we need to use the infimum here instead of the minimum, since the interval might be left-open.

– *There is no* $e \in E$ *such that* $freq(e) = [0, 0]$.
– *If* $e \in E$ *and* $\mu(e) =$ **mandatory**, *then* $freq(e) = [1, 1]$.

Example 3. Revisiting the Feature Diagram of Example 2, we are now able to work the frequencies back in, as given in Fig. 1, replacing the percentages given above by the corresponding frequency intervals:

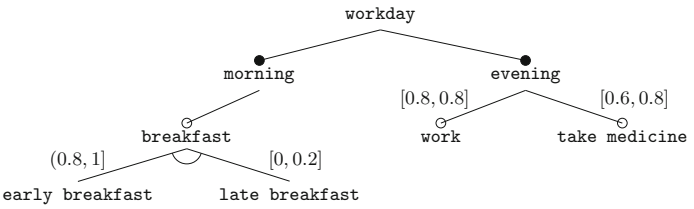

Fig. 4. Well-formed probabilistic feature diagram for the workday example

4.2 Hypothesis Testing for Probabilistic Feature Diagrams

Defining a semantics for the frequencies of a (well-formed) probabilistic Feature Diagram $\mathcal{D} = \langle D, freq \rangle$ requires a specification of the satisfaction of an edge (n, m) of D with frequency p with respect to a trace on D that represents the recorded behaviour of the user of the epartner over time. If these traces were infinite, we could calculate exactly whether the user behaviour indeed complies with the specified frequency by taking the ratio of the occurrence of m relative to n in the limit.

In practice however we need to evaluate compliance over varying finite time horizons, for example one week after the user has specified a new behaviour norm, but also after one month of trying to adopt a new habit, and possibly many other times. The observed frequencies will rarely be exactly equal to the desired frequency[2], since habitual user behaviour will often vary somewhat over time. In addition our sample size may prevent the possibility of exact compliance, e.g., if the desired frequency is 0.8 but we evaluate compliance over a trace of length 7. Nevertheless we want our epartner to be able to assess compliance in these cases.

To address these challenges, we employ a statistical technique called *hypothesis testing* [10]. Hypothesis testing is a type of statistical model checking to verify whether a system model satisfies a property of interest with a probability above or below a certain threshold value: the hypothesis. The core idea of statistical model checking is to use a computer program to repeatedly simulate

[2] At least when this frequency is a point. However also in case of an interval we need to ask whether it is justified to conclude (non-)compliance if the observed behaviour frequency is close to the edges of the interval.

the behaviour of the system model. For each of these simulations (samples), one can check whether or not the property of interest holds. One might see each such sample as a coin toss for which we can check whether it satisfies a certain property (let's say 'heads'). Using statistical techniques one can then determine whether it is justified to reject or accept the stated hypothesis, i.e., whether the true probability of the system exhibiting the property of interest can be assumed to be as stated by the hypothesis. For example, whether we can accept the hypothesis that the probability of the coin turning up heads is bigger than 0.7.

The idea of using hypothesis testing for defining the semantics of a probabilistic Feature Diagram now is to treat each state of a trace on the Feature Diagram as one possible sample: each state represents one instance of the user executing the behaviour specified by the Feature Diagram. Thus instead of repeatedly simulating a system model, we use repeated observations of the type of user behaviour expressed by the Feature Diagram. Recalling the user's intended behaviour of Example 1, the idea is that the epartner will construct a sequence $\langle \sigma_0, \sigma_1, \ldots, \sigma_t \rangle$ of length t after running for t days, monitoring only behaviour that is recorded in the Feature Diagram and thus relevant for monitoring norm compliance, and recording a separate model σ_j for each new day. The property of interest in our case is the occurrence of a behaviour m for a link (n, m) with some frequency, e.g., the user having `early breakfast`, in those states where n (`breakfast`) occurs (the sample size). This means we assume that these models are obtained via independent, identically distributed random processes as described above. Investigating to what extent we need to address possible dependencies between the creation of these models (e.g., once a user starts exhibiting non-compliant behaviour it is more likely that it will continue to do so) is left for future work.

Since hypothesis testing can only be used to verify whether the true probability is above or below a certain threshold value, for a frequency p that is a point we cannot conclude that the user behaviour is compliant. However, we *can* conclude that it is non-compliant, if it is (sufficiently) above or below p.

Hypothesis testing has previously been applied in the context of multi-agent systems to let agents hypothesise the likelihood that other agents will choose certain actions, based on their interaction history [1]. Instead, our work allows a behaviour support agent to assess whether observed user behaviour complies with given behaviour norms.

4.3 Semantics

In this section we formally define the semantics of probabilistic Feature Diagrams through hypothesis testing. Along the lines of [3,4], we first introduce the notion of a *j-sample* that takes the first j elements of the trace under consideration, allowing to select the sample we want to assess. Here, it is important that we do not 'mix and match' any specific parts of the trace, but pick j consecutive elements, without any discrimination. In a second step, the sample is processed through a statistic function T, which counts the number of times a node is included in the states of the selected part of the trace.

Definition 7. (j-Sample and Statistic). *Let $\mathcal{D} = \langle D, freq \rangle$ be a probabilistic Feature Diagram with $D = (N, E, \lambda, \mu)$, and let $\vec{\sigma}$ be a trace on D that satisfies D. The j-sample of $\vec{\sigma}$ is the initial sequence $\langle \sigma_0, \sigma_1, \ldots, \sigma_{j-1} \rangle$ of $\vec{\sigma}$ of length j. We denote the j-sample by $\vec{\sigma}(j)$.*

Let $\Sigma(D)$ be the set of traces on D and $\Sigma^{(j)}(D)$ be the set of j-samples of the traces. We define the statistic T on $\Sigma^{(j)}(D)$ and the nodes N of D by

$$T : \Sigma^{(j)}(D) \times N \to \mathbb{N}$$
$$T(\vec{\sigma}(j), m) = \sum_{s=0}^{j-1} \mathbb{1}_m(\sigma_s),$$

where

$$\mathbb{1}_m(\sigma) = \begin{cases} 1 & \text{if } m \in \sigma, \\ 0 & \text{otherwise.} \end{cases}$$

We can now use the statistic T to test for the hypothesis that the trace generated by the user's behaviour is compliant with the information in the probabilistic Feature Diagram $\mathcal{D}$ provided by the user. We opt here to use a test based on constructing confidence intervals for standard normally distributed random variables, which we will call Gauss-CI test, following the reasoning given by [10]. Given the properties of various tests described in [10], we opted for the Gauss-CI test since it works with a fixed sample size; in contrast to the model checking discussed there, the behavioural traces we deal with in this situation are indicating *past behaviour*, and we need our test to provide us with some answer towards (non-)compliance, so that a support system using the test can respond appropriately. This comes at the trade-off of drawing the wrong conclusion, or no conclusion at all, should the actual frequency of an activity be very close to the desired frequency. Since we would argue that a support system should not need a large sample of past behaviour before it can operate, we deem this acceptable.

Definition 8 (Gauss-CI test). *Let $\vec{\sigma}$, $\vec{\sigma}(j)$, $\mathcal{D}$ and T be as in Definition 7 above and let $\alpha \in [0, 1]$.*

Let $(n, m) \in E$ with $freq((n, m)) = p_m = [p_{0,m}, p_{1,m}]$. Let $T(\vec{\sigma}(j), n) = k$,

$$S_l(\vec{\sigma}(j), (n, m)) = (T(\vec{\sigma}(j), m) - k \cdot p_{0,m}),$$
$$S_u(\vec{\sigma}(j), (n, m)) = (T(\vec{\sigma}(j), m) - k \cdot p_{1,m}).$$

- *Let $l = l(\alpha, p_{0,m}) = \Phi^{-1}(\alpha) \cdot \sqrt{k \cdot p_{0,m} \cdot (1 - p_{0,m})}$ and $u = u(\alpha, p_{1,m}) = \Phi^{-1}(1 - \alpha) \cdot \sqrt{k \cdot p_{1,m} \cdot (1 - p_{1,m})}$, where Φ is the cumulative distribution function of the standard normal distribution.*
  We say that with confidence $(1 - \alpha)$, we reject the hypothesis that $p \geq p_{0,m}$ *if $S_l(\vec{\sigma}(j), (n, m)) < l$, and we* reject the hypothesis that $p \leq p_{0,m}$ *if $S_l(\vec{\sigma}(j), (n, m)) > -l$.*

We say that with confidence $(1-\alpha)$, we reject the hypothesis that $p \geq p_{1,m}$ *if* $S_u(\vec{\sigma}(j),(n,m)) < -u$, *and we* reject the hypothesis that $p \leq p_{1,m}$ *if* $S_u(\vec{\sigma}(j),(n,m)) > u$.
- *We say that the test is* inconclusive *in all other cases.*

We will use the test defined above to give a formalization for our notion of *probabilistic norm compliance*. In essence, given some interval $[p_{0,m}, p_{1,m}]$, we want to be certain that the frequency p in our sample is not too low or too high, i.e. we want to rule out that $p < p_{0,m}$ or $p > p_{1,m}$. For this, we obtain 'confidence intervals' $[l, -l]$ and $[-u, u]$[3] for the values of $p_{0,m}$ and $p_{1,m}$, respectively. That is, if the statistic S_l is larger than $-l$, then we may assume – with error level α – that $p > p_{0,m}$, and similarly, we may assume $p < p_{1,m}$ if $S_u < -u$. If both inequalities hold, we can safely assume that the norm of doing the specified activity m with a frequency in the interval $[p_{0,m}, p_{1,m}]$ is complied with. Furthermore, if $S_l < l$, we may safely assume that the norm is not complied with, with a frequency that is too low, or similarly, we may assume that the frequency is too high in case $S_u > u$. In all other cases, the frequency p is too close to one of the endpoints $p_{0,m}$, $p_{1,m}$ to be certain that it is on the right side of the endpoint, and therefore the test will be inconclusive.

Definition 9 (Probabilistic Norm Compliance). *Let* $\mathcal{D} = \langle D, freq \rangle$ *be a probabilistic Feature Diagram with* $D = \langle N, E, \lambda, \mu \rangle$, $\vec{\sigma}$ *a trace on* D *satisfying* D *and* $\alpha \in [0,1]$ *an error level. Let* $(n,m) \in E$ *be an edge with* $freq((n,m)) = [p_0, p_1]$, *we say that*

- $\vec{\sigma}$ *is* compliant with $\mathcal{D}$ for (n,m), *if the test defined in Definition 8 rejects the hypotheses* $p \leq p_0$ *and* $p \geq p_1$;
- $\vec{\sigma}$ *is* non-compliant with $\mathcal{D}$ for (n,m), *if the test either does not reject* $p \leq p_0$ *or* $p \geq p_1$;
- $\vec{\sigma}$ *is* inconclusive *for* (n,m) *otherwise.*

With $S_l(\vec{\sigma}(j),(n,m))$ *and* $S_u(\vec{\sigma}(j),(n,m))$, $l(\alpha,p_0)$, $u(\alpha,p_1)$ *given as above, let the* compliance function *be the function* $R(\mathcal{D},\vec{\sigma},j,\alpha,(n,m))$ *defined by*

$$R(\mathcal{D},\vec{\sigma},j,\alpha,(n,m)) = \begin{cases} \texttt{compliant} & \text{if } S_l(\vec{\sigma}(j),(n,m)) > -l(\alpha,p_0) \\ & \text{and } S_u(\vec{\sigma}(j),(n,m)) < -u(\alpha,p_1), \\ \texttt{non-compliant-too-high} & \text{if } S_u(\vec{\sigma}(j),(n,m)) > u(\alpha,p_1), \\ \texttt{non-compliant-too-low} & \text{if } S_l(\vec{\sigma}(j),(n,m)) < l(\alpha,p_0), \\ \texttt{inconclusive} & \text{otherwise.} \end{cases}$$

Note that in the special case of $p_0 = p_1 = \bar{p}$, i.e. the interval is a singleton, the compliance function can never provide the value `compliant`, since we need to reject both $p \geq \bar{p}$ and $p \leq \bar{p}$, and thus in particular reject $p = \bar{p}$.

[3] Note that we will have $l < 0 < u$, so the intervals are indeed sound.

5 Experimental Evaluation

5.1 Experimental Setup – Obtaining the Feature Diagram and Models

We will now proceed to put the formal definitions of the previous sections to the practice. Namely, we will evaluate an existing daily behaviour dataset [14] with our compliance function R given in Definition 9. The dataset consists of data about the execution of activities of daily living – e.g., eating and drinking, sleeping, working, watching tv, taking medicine, etc. – of several individuals (workday and weekend), over about 2 months. For this paper we have used the data in the file `data/edited_hh104_labour.xes.gz`, which has workday data of 43 days of user `hh104`. A typical entry for a single activity consists of a `start` event and a corresponding `complete` event (not shown here):

```
<event>
    <string key="concept:name" value="eatingdrinking"/>
    <string key="lifecycle:transition" value="start"/>
    <date key="time:timestamp"
        value="2011-06-15T07:11:45.000+02:00"/>
    <string key="work" value="eatingdrinking"/>
</event>
```

Since the dataset does not provide the Feature Diagrams corresponding to the desired behaviour of the user, we have reconstructed a possible Feature Diagram from the events given in the dataset. The sample entry above, for instance, indicates that the user had a meal on the morning of 15 June 2011, between 7:11 and 7:23. Thus we can take this entry as representing an instance of `breakfast`. We would separate the breakfasts into `early breakfast` in case the time of day is between 6 a.m. and 9 a.m., and classify a meal in the morning as `late breakfast` in case it takes place later than that but before noon. For this classification we only consider the start times of events.

Note that not all event entries of the dataset have been represented in the FD: for instance, we did not take any patterns for `sleep` into account here. We have picked values for the frequencies that might be considered desired behaviour for this user. For example, we noticed this user typically has breakfast rather late after doing some other activities, while it may be considered more healthy to start the day with breakfast. The resulting reconstructed probabilistic Feature Diagram is then that of Fig. 3.

To obtain the models, i.e., states of our trace and number of occurrences of the nodes in our Feature Diagram over this trace (the function T of Definition 7) we have made an implementation in the knowledge graph language Grakn (version 1.5.3 for Mac). The language allows to define an expressive schema in graph form over a given dataset. The tree structure of our Feature Diagrams lends itself well to implementation using knowledge graphs. We implement the Feature Diagram syntax and semantics as a Grakn schema. In order to obtain the

models to make up our trace, we need to define when the nodes of our example Feature Diagram are satisfied with respect to the dataset, e.g., `early breakfast` holds on a certain date if the above event occurs in the dataset for that date. We specify this using Grakn rules. We obtain the number of occurrences of nodes in our trace using a query over our schema and the imported dataset.[4] With $j = 43$, i.e., taking all workdays present in our dataset, we obtain the following number of occurrences of nodes of our Feature Diagram: `workday`, `morning`, `evening`, `breakfast`: 43; `early breakfast`: 11; `late breakfast`: 32; `work`: 38; `take medicine`: 33.

5.2 Results in Probabilistic Norm Compliance

Given the numbers of occurrences, and the desired frequencies, we apply the testing framework given above. First we apply the statistic of Definition 7. Note that in this case, we have $j = k = 43$, since both the `breakfast` and `evening` nodes occur 43 times. Note that the nodes for `workday` and `morning` are left out here, since our Feature Diagram only specifies frequences for the leaf nodes with respect to their parents. We use Definition 8 to obtain confidence intervals for the values of the statistics. We calculate those values to two decimal places, using the `norm.ppf` function of Python's `scipy.stats` package to obtain values for Φ^{-1}. Finally, we use the Probabilistic Norm Compliance function R from Definition 9 in order to determine whether the sample data indicates compliance with the relative frequencies of the Feature Diagram. As an error level for our test, we pick a value of $\alpha = 0.05$, or 5%.

Node	k	T	α	p_0	p_1	l	S_l	u	S_u	R(esult)
`early breakfast`	43	11	0.05	0.8	1	−4.31	−23.4	0	−32	`non-compliant-too-low`
`late breakfast`	43	32	0.05	0	0.2	0	32	4.31	23.4	`non-compliant-too-high`
`work`	43	38	0.05	0.8	0.8	−4.31	3.6	4.31	3.6	`inconclusive`
`take medicine`	43	33	0.05	0.6	0.8	−5.28	7.2	4.31	−1.4	`inconclusive`
`take medicine`	43	33	0.3	0.6	0.8	−1.68	7.2	1.36	−1.4	`compliant`

We can see that for `breakfast`, the test result clearly indicates that the ratio of `early breakfast` is far too low, and symmetrically, `late breakfast` occurs too often in the data. However, for `work` and `take medicine` the test result indicates an inconclusive result. For `take medicine` the values indicate that this node is realized with a ration of at least 60%, but the actual rate is likely too close to the upper bound of 80% for the test to return meaningful results. In fact, as the last line in the table above shows, we can obtain a compliant result if the error level is substantially increased (to 30% in this case).

[4] The code is available from GitHub repository [20].

6 Conclusion

We have introduced Feature Diagrams as a way of expressing and assessing compliance with desired user behaviour. In order to represent relative frequencies we proposed a probabilistic extension of Feature Diagrams. Interpreting such a probabilistic Feature Diagram as a record of behavioural norms, we can use Hypothesis Testing methods to monitor compliance with these norms in daily behaviour. The methods demonstrated here allow not just to measure compliance itself, but also allow to give an estimate of whether non-compliant behaviour occurs with too low or too high a frequency, compared to the recorded values in the Feature Diagram.

Our experimental evaluation is based on a dataset that consists of such records of daily behaviour. While the Feature Diagram corresponding to this data was a reconstruction, the results presented in Sect. 5 nonetheless demonstrate that the concepts and methods used in this paper provide meaningful answers to the question of whether a pre-recorded behavioural norm is complied with by a user. The framework of probabilistic Feature Diagrams and Hypothesis testing methods can also provide us with meaningful results in the presence of a rather small set of data points.

Expanding from this proof of concept, we intend to investigate formal properties of this framework. We further plan a user study, investigating ease of use of the framework in the intended field of application, and assessing intuitiveness of the notion of probabilistic norm compliance for assessing satisfaction of behaviour norms.

References

1. Albrecht, S.V., Ramamoorthy, S.: Are you doing what i think you are doing? criticising uncertain agent models. In: Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, Arlington, Virginia, United States, UAI 2015, pp. 52–61. AUAI Press (2015)
2. Andrighetto, G., Governatori, G., Noriega, P., van der Torre, L. (eds.) Normative Multi-Agent Systems, vol. 4 of Dagstuhl Follow-Ups. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2013)
3. Cheung, L., Stoelinga, M., Vaandrager, F.W.: A testing scenario for probabilistic processes. J. ACM **54**(6), 29 (2007)
4. Gerhold, M., Stoelinga, M.: Model-based testing of probabilistic systems. Formal Asp. Comput. **30**(1), 77–106 (2018)
5. Hull, C.L.: Principles of Behavior: An Introduction to Behavior Theory. Appleton-Century, New York (1943)
6. Kließ, M.S., Jonker, C.M., van Riemsdijk, M.B.A.: Temporal logic for modelling activities of daily living. In: Alechina, N., Nørvåg, K., Penczek, W. (eds.) 25th International Symposium on Temporal Representation and Reasoning, TIME 2018, vol. 120 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 17:1–17:15. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)

7. Oinas-Kukkonen, Harri: Behavior change support systems: a research model and agenda. In: Ploug, Thomas, Hasle, Per, Oinas-Kukkonen, Harri (eds.) PERSUASIVE 2010. LNCS, vol. 6137, pp. 4–14. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13226-1_3
8. Pasotti, P., Jonker, C.M., van Riemsdijk, M.B.: Action identification hierarchies for behaviour support agents. In: Third Workshop on Cognitive Knowledge Acquisition and Applications, Cognitum 2017 at IJCAI 2017 (2017)
9. Pasotti, P., van Riemsdijk, M.B., Jonker, C.M.: Representing human habits: towards a habit support agent. In: Proceedings of the 10th International Workshop on Normative Multiagent Systems, NorMAS 2016 (2016)
10. Reijsbergen, D., de Boer, P.-T., Scheinhardt, W., Haverkort, B.: On hypothesis testing for statistical model checking. Int. J. Softw. Tools Technol. Transfer **17**(4), 377–395 (2015)
11. Schobbens, P., Heymans, P., Trigaux, J.: Feature diagrams: a survey and a formal semantics. In: 14th IEEE International Requirements Engineering Conference, RE 2006, pp. 139–148, September 2006
12. Shafti, Leila S., Haya, Pablo Alfonso, García-Herranz, Manuel, Alamán, Xavier: Personal ambient intelligent reminder for people with cognitive disabilities. In: Bravo, José, Hervás, Ramón, Rodríguez, Marcela (eds.) IWAAL 2012. LNCS, vol. 7657, pp. 383–390. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35395-6_52
13. Sun, J., Zhang, H., Fang, Y., Wang, L.H.: Formal semantics and verification for feature modeling. In: 10th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2005. IEEE, pp. 303–312 (2005)
14. Sztyler, T.T, Carmona, J.J.: Activities of daily living of several individuals (2015). https://data.4tu.nl/repository/uuid:01eaba9f-d3ed-4e04-9945-b8b302764176
15. ter Beek, M.H., Legay, A., Lafuente, A.L., Vandin, A.: Statistical analysis of probabilistic models of software product lines with quantitative constraints. In: Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, New York, NY, USA, pp. 11–15. ACM (2015)
16. Thangarajah, J., Padgham, L., Winikoff, M.: Detecting & exploiting positive goal interaction in intelligent agents. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2003, pp. 401–408. ACM (2003)
17. Tielman, M.L., Jonker, C.M., van Riemsdijk, M.B.: What should i do? deriving norms from actions, values and context. In: Cassens, J., Wegener, R., Kofod-Petersen, A. (eds.) Proceedings of the Tenth International Workshop Modelling and Reasoning in Context, MRC 2018, no. 2134, pp. 35–40. CEUR Workshop Proceedings (2018)
18. Vallacher, R.R., Wegner, D.M.: What do people think they're doing? action identification and human behavior. Psychol. Rev. **94**(1), 3–15 (1987)
19. van Riemsdijk, M.B., Jonker, C.M., Lesser, V.: Creating socially adaptive electronic partners: interaction, reasoning and ethical challenges. In: Proceedings of the Fourteenth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, pp. 1201–1206. IFAAMAS (2015)
20. van Riemsdijk, M.B.: Feature diagrams for behaviour support agents in Grakn. https://github.com/mbirna/feature-diagrams-grakn (2019)