Representing human habits: towards a habit support agent

Pietro Pasotti, M. Birna van Riemsdijk, Catholijn M. Jonker

Abstract. Human behaviour is constrained by obligations on the one hand, by the routines and habits that constitute our normal behaviour on the other. In this paper, we present the core knowledge structures of HabInt, a Socially Adaptive Electronic Partner that supports its user in trying to adopt, break or maintain habitual behaviours. We argue that HabInt’s role is best conceived of as that of an extended mind of the user. Hence, we pose as requirements that HabInt’s representation of the relevant aspects of the user and her world should ideally correspond to that of the user herself, and use the same vocabulary. Furthermore, the knowledge structures of HabInt should be flexible and explicitly represent both its user’s actual habitual behaviours and her desired habitual behaviours. This paper presents knowledge structures that satisfy the aforementioned requirements. We interleave their syntactic specification with a case study to show their intended usage as well as their expressive power.

1 Introduction

Man is a creature of habit. While people display a fascinating variety of behaviours even across relatively simple domains, it is also true that from day to day most people are quite fixed in their ways. Carrying out habitual activities is mostly unproblematic and even desirable ([29]). However at times unforeseen circumstances make our habitual choices unavailable or their outcomes undesirable. Other times we wish to adopt or break a habit, and both are difficult enterprises. While many of us normally have little or no difficulty in dealing with these challenges ([29]), the actual amount of nuisance is subjective. To some, even small disruptions of daily routines may cause anxiety and distress ([12, 18]), whereas for others, such as people suffering from depression, breaking habits can be beneficial ([24]).

In this paper, we take the first steps in developing a concrete implementation of habits even across relatively simple domains, it is also true that from day to day most people are quite fixed in their ways. Carry-
The ultimate goal of HabInt is to help the user achieve her goals and promote her values. In this sense, an implementation of a HabInt is best conceived of as part of the extended mind (see [9]) of its user. This means that it must be trustworthy, reliable and accessible (cfr. [9]) and so must be its knowledge. To make HabInt accessible and trustworthy, we must provide knowledge structures that are as transparent for the user as possible. The knowledge must be readily available for the user not only to use, but also to expand, contract and otherwise modify in a way that matches the way behaviours are discovered, explored, abandoned by people. To further enhance trust and reliance, we limit the agent’s proactiveness to only those actions that are explicitly requested by the user. Accessibility, for one, means that the information/knowledge in the system must be easily accessible by the user. Consequently the HabInt has to store and manipulate its user model explicitly (unlike, for example, a neural network). For another thing, the user model should match the one the user has of herself as closely as possible, i.e. it should be a shared mental model (see [17]). Thus HabInt builds the vocabulary of goals, values, activities and actions from the user’s wording. Summing up, HabInt’s knowledge structures should be:

- **adaptable**: obtained by interacting with the user. This entails that they need to tolerate runtime updates and be built incrementally, while remaining meaningful at all intermediate stages of the construction process.
- **shared**: correspond as much as possible to the user’s conceptual structure and use the same vocabulary as the user does.5
- **explainable**: HabInt needs to be able to carry out reasoning and explain the reasons that led to its current beliefs, in a dialogue referring to goals, values, and situational aspects ([30]). For example, HabInt should be able to model and then explain back to the user as requested which values are positively or negatively affected by some activities, to which goals activities contribute, and which values motivate which goals.
- **expressive**: the structures need to accommodate uncertain, incomplete, and even inconsistent information. Finally, they must express the (context-dependent) behaviour enactment likelihood, for that is how HabInt can tell whether a behaviour is a habit or not, or whether it is becoming or ceasing to be one.

To show HabInt’s intended usage and the expressive power of its knowledge structures, we introduce a few snapshots of the life of a woman, Alice, as she interacts with Hal, her HabInt. Throughout the paper we will refer back to these scenarios and show how they are dealt with behind the scenes by Hal.

**SCENARIOS: Alice and Hal**

**S1** Alice has a new job and would like to form a robust routine for travelling there. Also she would like to stop oversleeping. To help her with these issues Alice buys an HabInt, which she calls Hal. After booting it, Alice explains to Hal that she has two goals: first, to ‘wake up’ and then to ‘get to work’. Hal asks what the options regarding the two goals are. It discovers that while there are a number of ways to get to the workplace, there is only one way of waking up, which requires remembering to set an alarm. Alice explains to Hal that the main ways of getting to work are 1) by car, and 2) by bike. Furthermore, one can go by bike in two ways, (2.1) via the fast but risky Route A, or (2.2) via the safer, but longer Route B.

**S2** Alice sometimes takes a cab to work. She feels no need for support in doing so, so when Hal reminds her to check the weather as she is leaving for work, she just says “well, actually today I’m going to work in some other way, so I won’t need it. You don’t need to worry about this.” HabInt does not know how Alice is going to work that day.

**S3** Alice now has the habit of setting the alarm every single day. However, exceptionally, on Mondays she forgets to set the alarm almost every other week (Probably this relates to her Sunday night’s Vodka Tasting Club meetings).

**S4** Alice tells Hal that of the two options to go to work by bike, she prefers the safe route (2.2) over the fast one (2.1). She explains that being fast is not as important to her as being safe.

**S5** Alice asked Hal to help her grow the habit of going to work by bike. Years later, however, Alice decides to stop biking to work and go by car instead. Thus specific habitual behaviours part of her previous biking-to-work-routine are no longer necessary. She tells Hal the following: “(Instead of going by bike) now I’d like to go work by car”, “I’ll also need to stop taking the raincoat as I go to work.”

**S6** Alice long ago told Hal that she dislikes ‘smoking’, an action, because it denotes ‘health’, which she greatly values. Consequently, she has not smoked a single cigarette for 10 years now. However, one day Hal learns that Alice is smoking. After inquiring, Hal is told simply: “I want to start smoking.”

### 3 The Actual Behaviour model (ABM)

There are habits regarding what activities we carry out daily; i.e. habits regarding, once something is done, what do we do next (next-actions). We model such activity patterns by capturing the sequential activation patterns of the goals that they purport to achieve.

Second, there are habits regarding the way in which we carry each activity out. We will call them conze-habits, for concretisation habits. The intuition is that just like the goal get home by car is intuitively more concrete that the goal get home, the activity of driving home, which achieves the former, is more concrete than the activity of going home, which achieves the latter. Achieving the former goal entails achieving the latter, but the converse does not hold. This is-a-way-of relation between goals is what we intend to capture with the notion of concretisation: we model habits regarding the way in which we do things by modelling the underlying goal concretisation patterns.

Finally, there are habits regarding what actions we perform as part of carrying out an activity (in a particular way): we call them Action-habits. For every activity, we represent the actions the user can perform when she tries to achieve its goal, and capture the likelihood that they are in fact performed.

In §3.1 and §3.2 we describe a knowledge representation language based on these three notions. Finally, exploiting our representation of the actions’ consequences, we can express the values that they affect, and hence talk about the motivation and preferences that underlie behaviour choice and change. This is done in §3.3.

The common basis of the language that the ABM is built upon is a language of alphanumerical strings. HabInt parses the User’s messages at the level of propositional logic operators and treats the remaining uninterpreted strings as atoms. For example, “[the user is] not eating” becomes ¬(‘eating’). A propositional language over Strings is the basis of the knowledge structures we define next. A logical consequence relation is defined on formulae over Latom in the standard way. We use , , as variables ranging over Latom.

#### 3.1 Activities: what we do and how we do it

Abstracting away the temporal features for the sake of simplicity, an Activity is informally understood as something which the user does to modify the current state of affairs. Most of our daily activities

---

5 I.e. if the user refers to its habit of ‘brushing teeth after every meal’, then that, literally, is the name HabInt stores.

6 We capitalise technical terms, to avoid confusion with common concepts.
are carried out with a purpose, which we call a Goal. Our working definition of Goal is: a declarative description of the state of affairs which the user would like to achieve by carrying out an Activity.

HabInt's most fundamental knowledge structure represents the user's daily activities' underlying goals, and what goals are concretisations of what other goals. We call this knowledge the Goal Base.

The relation conc defines a branching structure of Goals that represents the way the user conceptualises her daily goals in terms of more concrete versions of themselves. This is captured by the binary relation conc. A special role is played by toplevel Goals, which are not a concretisation of any other Goal. In other terms, those for which the user sees no need to provide a higher goal. Examples for Alice include being awake early, having breakfast, getting to work and back home again. Toplevel Goals are then linked to one another by the relation next, forming a separate branching structure. This structure represents the user's potential Goal activation sequences: information about what she might do after doing something else.

Definition 1 (Goals and Goal Base) The user's Goal Base is a triple $G := (G, conc, next)$, where:

- $G \subseteq L_{act}$, the current set of Goals of the user, with typical variables $g$ and $g'$.
- $conc : G \times G$ is a directed and acyclic concretisation relation, such that $\forall g, g' \in G : conc(g, g')$ if $g'$ is a concretisation of $g$.
- next $\subseteq top_G \times top_G$ is a directed, acyclic relation such that next$(g, g')$ if once she has satisfied goal $g$, the user may try to satisfy (i.e. adopt) $g'$ next.

Furthermore, $top_G$ is the set of toplevel goals of $G$:

$$top_G := \{ g \in G | \forall g' \in G : conc(g', g) \}$$

Note that the user can specify as many Goals as she likes, and can leave gaps and blanks. So, even if she habitually smokes at home, her HabInt may never know. It is up to the user to inform HabInt of alternative activities for the goals she mentioned to it, even if the user leaves these underspecified. Hence HabInt must assume that unknown, additional alternatives always exist.

By monitoring and interacting with the user, HabInt learns and keeps track of the Activities that she carries out as she tries to achieve unknown, additional alternatives always exist.

As we mentioned in § 1, a habit is not just a "frequent behaviour". However, frequency, automatism, ease of performance and other features of habitual behaviours are correlated. in particular frequency

3.2 Habits: the way we normally do what we do

As we mentioned in § 1, a habit is not just a "frequent behaviour". However, frequency, automatism, ease of performance and other features of habitual behaviours are correlated. in particular frequency
can serve as a predictor for the other features and it can be derived from observing and communicating with the user ([35]). Therefore, we chose to detect habits through the underlying behaviours’ enactment likelihoods. We describe a user’s day as a sequence of top-level goals (given by next). Each of those can then be concretised in different ways (as described by conc), and each goal can be assigned, via an Activity, a set of Actions that can be executed whilst achieving it. This is information about what the user is known to sometimes do: it defines the space of possible behaviours. Each one of these may in practice be enacted rarely or never, and both their content and their performance frequency can change over time. Consequently, we keep the representation of what the user knows she may do (next, conc, Activities) separate from the expectations regarding what she will do.

We have seen in § 1 that habits are cued by contexts. Hence we must keep track of those parts of the context that are believed by the user or by some internal learning algorithm to cue some behavioural response. We call them Triggers. Let $T \subseteq \mathcal{L}_{act}$ be a finite set of known Trigger. Let $\tau$ range over $T$.

However, reacting to Triggers is not automatic: even in the presence of a Trigger the cued behaviour may not follow. Then we must record, given the presence of a Trigger, the likelihood of the associated behaviour occurring. This is captured by $\text{prob}$. Formally, $\text{prob}$ is a function of type $(T \times G \times (G \cup \mathcal{L}_{act})) \rightarrow [0, 1]$. Intuitively:

$$\text{prob}(g' | \tau, g) = x,$$

top level goals $g$ and $g'$, means that given that $g$ has been just achieved and that the Trigger $\tau$ holds, then the user adopts $g'$ with likelihood $x$. If $g'$ is not top level, then it means that while she tries to achieve $g$, and given Trigger $\tau$ the user is expected to adopt $g'$ with likelihood $x \in [0, 1]$.

$$\text{prob}(\alpha | \tau, g) = x,$$

means that if the goal $g$ is adopted and $\tau$ holds, then the user is expected to execute action $\alpha$ with likelihood $x$.

Some behaviours’ Triggers can be unknown, or so frequent to be irrelevant. In that case the Trigger is true.

If, given a Trigger, the enactment likelihood of some behaviour is above a certain threshold $t \in [0, 1]$, HabitInf infers that the behaviour is a habit. We call $t$ the user’s habit threshold, which is the performance likelihood above which the user feels confident in calling something a habit. Given existing research (e.g. [36]), it is reasonable to assume that $t > 0.5$. The return values of $\text{prob}$ are estimated based on information from the user and/or sensor data (cf. [20]). Now we must keep in mind a key property of the notion of Goal we employ here: Goals that are concretisations of the same goal cannot be adopted concurrently. While this is not generally true for next-related Goals, here for simplicity we assume it is. For example, after waking up, Alice can go to work or go to the beach, not both. Also, as she goes to work, Alice can go ‘by bike’ or ‘by car’, not both. Therefore, no matter how $\text{prob}$ is calculated, the likelihoods must sum up to one on all outgoing next paths and on all outgoing conc paths too. Actions are executed independently from one another, so there is no such constraint there.

The data structures we have defined so far allow us to express the types of habits described at the beginning of this section: next-, conc- and action-habits. These correspond to transitions between next-related goals, conc-related goals, and $(g, \alpha)$ pairs respectively.

3.3 Values: why we do what we do as we do it

Even though a HabitInf having only the above structures can already be of use, in our opinion it needs to understand the motivations (based on values) for the choices the user makes to best support her. The user may need support in making satisfactory choices regarding her behaviour, which involves comparing competing Actions, based on their outcomes; competing Activities, based on the Actions they are associated with; and competing Goals, based on the Activities that they can be achieved by. For supporting the user, HabitInf needs to understand and reason with the motives of the user’s behaviour, and thus needs to know and understand her values.

Paraphrasing [19],

we understand values as a hierarchy of desirable, abstract, cross-situational goals. Ultimately, any activity is motivated by the pursuit of values. Still, all actions that we take as part of any activity end up affecting the same values (see Figure 3). So the user interface must be capable of value-based argumentation, and it is therefore natural to store also these knowledge structures in the unified world model we are describing here. As our HabitInf is a personal support agent, here we assume that every user has her own (hierarchy of) values and hence we ignore their often-alleged universality ([23]). With the help of the user, HabitInf learns what values she has, how important they are relative to each other, and what world features (literals from $\mathcal{L}_{act}$) can affect them. HabitInf reasons about Values using value-based argumentation frameworks (cfr. [5]).
Definition 5 (Values and Value Base) We define $V := (V, \preceq, \text{pro})$ to be the Value Base of the user, where

- $V \subseteq G$ denotes the set of given Values of the user.
- $\preceq \subseteq V \times V$ is a preorder, such that $\forall v, v' \in V : v \preceq v'$ holds if $v$ is less important than $v'$.
- $\text{pro} := L_{\text{var}} \times V \rightarrow \{\triangledown, \lhd, \triangleright\}$ is an injective function encoding the way literals $\alpha$ from $L_{\text{var}}$ promote $\triangledown$ ($\lhd$) or not affect ($\triangleright$) the user’s values.

Note that the default return value of the function $\text{pro}$ is $\triangleright$: we assume that the user does not know or does not care, until she says otherwise.

When the user and her $\text{HabInt}$ are reasoning about the best course of action to take, the postconditions of the actions involved play the fundamental role. Each postcondition expresses not only the goal its Action achieves but (in conjunctive normal form) a list of its effects. Exploiting this fact, $\text{HabInt}$ can infer from the Value Base the way Actions first, then Activities affects Values.

As abstract goals of activities, values may well be unspecified and in the background. But when it comes to evaluating the effects of the concrete Actions that together form an Activity, the importance and visibility of values become greater. Actions can be said to promote and denote values by bringing about their postcondition and, through the Actions that habitually achieve them, so can Activities. While Actions’ outcomes are stable, habits dictate which Actions are executed when carrying out an Activity. Therefore, to determine what values are affected by an Activity, one must factor in habits.

With the Value Base, all parts of the ABM have been discussed.

Definition 6 (Actual Behaviour model (ABM)) The Actual Behaviour Model is the tuple $\mathfrak{A} := (V, G, A, C, T, \text{prob}, l)$, where the elements are respectively, the Value Base, the Goal Base, the Activity Base, the Action Base, the set of Triggers, the conditional likelihood function, and the habit threshold.

Given $\text{pro}$, which tells how $L_{\text{var}}$ literals affect Values, we generalise it to $\text{pro}^\ast$, which also tells how Actions and Activities do.

Definition 7 (Promote) Given an ABM $\mathfrak{A}$, the function $\text{pro}^\ast := ((L_{\text{var}} \cup L_{\text{act}} \cup L_{\text{conc}}) \times V) \rightarrow \{\triangledown, \lhd, \triangleright\}$ is defined as follows:

- If $a$ is a literal from $L_{\text{var}}$, then $\text{pro}^\ast(a, v) = \text{pro}(a, v)$.
- If $\varphi \in L_{\text{act}}$, then we require all the disjuncts to ‘agree’ on $v$:
  
  $\text{pro}^\ast(\varphi \lor a, v) = \begin{cases} 
  \text{pro}(a, v) & \text{if } \text{pro}^\ast(\varphi, v) = \text{pro}(a, v) \\
  \triangleright & \text{otherwise}
  \end{cases}$

- Let $a$ be an action $[1 : a \ast b]$, and $\text{cnf}(a)$ denote the set of $b$’s conjunctive normal form’s conjuncts (with $\mathbb{B} \in \{\lhd, \triangledown\})$. Given:
  
  $C_{a}^{\mathbb{B}} := \{ \varphi \in \text{cnf}(a) : \text{pro}^\ast(\varphi, v) = \mathbb{B} \}$
  
  $\text{pro}^\ast(\varphi, v) = \uparrow$ if $|C_{a}^{\mathbb{B}}| > |C_{a}^{\lhd}|$

  (1)

  similar to [33], we say that $\alpha$ promotes a Value $v$ ($\text{pro}^\ast(\alpha, v) = \uparrow$) if it brings about more promoting than $v$-demoting postcondition. The conditions for $\text{pro}^\ast(\alpha, v) = \downarrow$ or $\mathbb{B} = \triangleright$ are very similar: change ‘$
  \ast$’ to ‘$
  \ast'$’ and ‘‘ in (1) respectively.

- Let $A = \{1, g, Act\}$, and $h(A) := \{ \alpha \in Act | \exists r, g : \text{hab}(\alpha, r, g) \in \mathfrak{A} \}$; then
  
  $D_{A}^{\mathbb{B}} := \{ \alpha \in h(A) : \text{pro}^\ast(\alpha, v) = \mathbb{B} \}$

  (2)

  $\text{pro}^\ast(\mathbb{A}, v) = \uparrow$ if $|D_{A}^{\uparrow}| > |D_{A}^{\triangledown}|$

8 Think about the habitual activity of going back home (after a day of work). The user can, but does not need to specify which values that macroscopic activity promotes.

This is crucial for $\text{HabInt}$ to represent inconsistences between what the user does, or wishes to do, and his Values (cf. §5 for an example).

[Behind the scenes of S4] Hal learns that Alice considers ‘be safe’ ($v_{1}$) and ‘be fast’ ($v_{2}$) as Values. Hence, it adds them to its previously empty Value Base, which now is $V = \{v_{1}, v_{2}\}$, $\emptyset$, $\emptyset$, $\emptyset$. Then it learns that biking through Route A (an Activity $A$) promotes ‘safety’, but it does not know what specific postcondition of what Action involved in the Activity promotes it. Hence, first Hal adds a dummy Action $\alpha =$ [‘something’ ‘something’ $\rhd$ ‘$a_{1}$’] to $A$ (where ‘$a_{1}$’ is a new atom), and then adds to its Value Base the fact that $\text{pro}(a_{1}, \langle ‘safety’ \rangle) = \uparrow$. Via the same process, it also records that Route B promotes ‘be fast’. From this, Hal can deduce that $\text{pro}(\langle ‘A’, ‘safety’ \rangle)$. Finally, it learns: ‘be fast’ $\lhd$ ‘be safe’. (Actually things are a bit more complicated, as promoting ‘safety’ seems to be a property the Activity always has, according to Alice. Hence, by chaining post- and pre-conditions appropriately, Hal must ensure that $\alpha$ is presumed executed by the user every time the Activity of ‘biking through route A’ is performed.)

4 The Desired Behaviour model (DBM)

People that are not quite satisfied with their actual behaviour may tell their $\text{HabInt}$ what is bothering them. Only then, can they describe how they would like to be supported in changing it.

While the ABM of a user describes what the user does (and how she does it) in specific situations, the Desired Behaviour Model (DBM) describes a set of Desired Habits to the ABM that reflect how the user would like her ABM to become. The key intuition here is that if conforming to a desired behaviour were not an issue under any circumstance, the user would not mention it to $\text{HabInt}$. Therefore, $\text{HabInt}$ treats each Desired Habit as a support request, which still does not convey any information about how the agent can in practice support the user. Later on, each Desired Habit can be linked to one or multiple ways in which the agent can support the user: for instance, instructions of when and how to produce a reminder, initiate a conversation, monitor some environmental variable, or ask what is going on. However, we do not discuss these in this paper.

In what follows, $\mathfrak{B}$ is an ABM, $\tau$ is a Trigger, $g, g' \in \text{Goals}$, and $\alpha$ is an Action (all from $\mathfrak{B}$). We consider Desired Habits of three types:

next-Desired Habits are structures of the form $(\tau, g, g')$, where next$(g, g')$ is part of the Goal Base of $\mathfrak{B}$. This Desired Habit type formalizes the user’s desires concerning her toplevel goal sequences. When she talks about what she should or would prefer to habitually do after doing something else, $\text{HabInt}$ will formalise that as a next-Desired Habit.

conc-Desired Habits are structures of the form $(\tau, g, g')$, where conc$(g, g')$. This Desired Habit formalizes habit change desires concerning the way the user achieves some goal (i.e. her concrétisation patterns). conc-Desired Habits formalise, for example, the user’s desired habitual way of achieving some toplevel Goal.

Action-Desired Habits are structures of the form $(\tau, g, \alpha)$, where there is some activity $A = \{1, g, Act\}$ in $\mathfrak{B}$’s Activity Base with $\alpha \in Act$. If the user wishes to change the actions she habitually performs as part of carrying out some Activity, that will be formalised as an Action-Desired Habit.

In a similar fashion we introduce undesired behaviours or the habits which the user wants to drop. We call them Undesired Habits. They are also expressed in $L_{\text{var}}$ but stored in a different set, undhab. Each Undesired Habit encodes the user’s desire to habitually not enact a behaviour (in some way) or perform an action (given some
Figure 4. An example Goal structure. The Goal \( \emptyset \) has two concretisations, \( \emptyset \) and \( \emptyset \). Also, after achieving \( \emptyset \), the user can try to achieve either \( \emptyset \) or \( \emptyset \).

Definition 8 (Desired Habits and Undesired Habits of the ABM) Given an ABM \( A = (V, G, A, C, T, \text{prob}, t) \), the set of Desired Habits is \( \text{Dhab} \subseteq L_{\text{dhab}} \), and the set of her Undesired Habits is \( \text{Undhab} \subseteq L_{\text{dhab}} \), where \( (g, g') \) are Goals in \( G \), \( \tau \in T \) and \( \alpha \in C \):

\[
L_{\text{dhab}} := \{ \langle \tau, g, g' \rangle | \langle \tau, g, \alpha \rangle \}
\]

With the difference between next- and conc-Desired Habits in mind, we can clarify their intended semantics by specifying the conditions under which they can be said to be compiled with. A Desired Habit points out a behaviour which should be habitual under some trigger; hence a Desired Habit is compiled with when that behaviour is indeed a habit (under that trigger). Similarly, a Undesired Habit is compiled with when the corresponding behaviour is not a habit.

Definition 9 (Compliance) Given a habit threshold \( t \) and an ABM \( A \), we say that \( A \) complies with

\[
\begin{align*}
\langle \tau, g, g' \rangle & \in \text{Dhab} \quad \text{iff} \quad \text{prob}(g'|\tau, g) > t & (2) \\
\langle \tau, g, \alpha \rangle & \in \text{Dhab} \quad \text{iff} \quad \text{prob}(\alpha|\tau, g) > t & (3) \\
\langle \tau, g, g' \rangle & \in \text{Undhab} \quad \text{iff} \quad \text{prob}(g'|\tau, g) < t & (4) \\
\langle \tau, g, \alpha \rangle & \in \text{Undhab} \quad \text{iff} \quad \text{prob}(\alpha|\tau, g) < t & (5)
\end{align*}
\]

Based on the known Triggers, \( \text{HabInt} \) keeps track of what behaviours the user wishes to change and stores them in its Dhab and Undhab. The Dhab, Undhab and ABM constitute the DBM.

Definition 10 (Desired Behaviour Model) Given the ABM \( A \), the Desired Habits Dhab, and the Undesired Habits Undhab, the Desired Behaviour Model is \( (A, \text{Dhab}, \text{Undhab}) \).

5 Violation, anomaly, and inconsistency monitor

The structures we have described so far capture (un)desired habits, one-off behaviours, existing habits, and also the user’s values, and all can be at odds with one another. Hence many types of conflict can be expressed in their language. Here we describe three: the most crucial ones to monitor for habit support. Namely we show that, given the DBM and ABM, \( \text{HabInt} \) can monitor whether an actual behaviour is anomalous, inconsistent with the user’s value-based preferences or whether it violates an existing Desired Habit. The examples point out how \( \text{HabInt} \)’s monitoring module can check the user’s ABM and DBM for such conflicts (all examples refer to Figure 4).

Behavioural anomaly: when the user does something unusual (or in an unusual way). For example, when \( \text{hab}(\tau, \emptyset, \emptyset) \), but the agent believes that the user is now doing \( \emptyset \) instead of \( \emptyset \).

When a behavioural anomaly is detected, \( \text{HabInt} \) can e.g. be instructed to investigate, remind the user of her habitual behaviour, or alert a supervisor. The ABM knowledge alone is sufficient for expressing this anomaly. Both ABM and DBM are needed, on the other hand, to express the following state of violation: when a Desired Habit is not a habit (or vice versa, when an Undesired one is).

\[
\langle \tau, g, g' \rangle \in \text{Dhab} \land \langle \tau, g, g'' \rangle \notin \text{Dhab} \land \langle \tau, g, g'' \rangle \in \text{hab}
\]

Undesired behaviour: when the user does something (in a way) she declared she does not want to (or should not). For example, if \( \emptyset, \emptyset, \emptyset \in \text{Dhab} \land \langle \tau, \emptyset, \emptyset \rangle \notin \text{Dhab} \), but the user believes that the user habitually does \( \emptyset \) after \( \emptyset \), when \( \tau \).

When undesired behaviour is detected, this means that the user is doing something she declared she wanted support in not doing (or vice versa). Many kinds of support can be associated with violations of this type. For example, the user may ask to be reminded of the values she invoked when she set the Desired Habit she is about to violate or to talk once more about the consequences of her behaviour. The same holds for one-off behaviours in place of habits.

Furthermore, using the notions introduced in § 3.3, \( \text{HabInt} \) can reason about which Values are affected by an Activity and know, for example, if an Activity \( A \) for the goal \( g \) denotes the user’s most important values: \( \forall v (v' \langle v < v' \Rightarrow \text{prob}(A, v) = 1) \)

If the user mentions that she is carrying out \( A \), or \( \langle \tau, g', g \rangle \in \text{Dhab} \), then her \( \text{HabInt} \) will detect a value inconsistency:

Value inconsistency: when the user’s Actions, Activities, or (Un)Desired Habits are not in line with her preferences. For example, when an Action denotes an important Value.

Other types of Desired Habits could in principle have been defined. For example, looking at Figure 4, one may wish to express Dhab(\( \emptyset, \emptyset, \emptyset \)). It could be read as requesting to form a habit of “instead of doing \( \emptyset \) by means of \( \emptyset \), stop doing \( \emptyset \) altogether and start doing \( \emptyset \) instead”. But this is rather convoluted, and we see little added value. Rarely we say things like: “if you see me go to work by bike, remind me I should stay home instead”. For similar reasons also the other possible Dhab-types require more far-fetched interpretations. So, we will not discuss them further.
deliver a warning, as previously instructed by the user. The value in-
consistency type of anomaly can be a symptom of inconsistencies in
the user’s motivation/intention/action structure, or irrational be-
aviour. To find out which one it is, HabInt can be instructed to ini-
tiate communication with the user.

6 Related work

Research on human-computer interaction has explored many ways
in which technology can be used to aid behaviour change (e.g. [25])
and support habit formation and breaking. In contrast with these ap-
proaches, our focus is not on the psychological aspects of behaviour
change in a specific domain and how to support this through tech-
nology. Rather, our conception of HabInt is as an extended mind of
the user: we focus on developing generic knowledge structures that
allow a HabInt to represent and construct user habits in a way that
corresponds to the her conception of her activities.

The field of Activity Recognition has developed machine learning
approaches to deduce what an observed human being is doing (and
her behavioural patterns too), based on raw sensor data. For exam-
pies and further references, see [25, 11, 20]. These techniques will
be used in the monitoring component, to update the prob function
and automatically determine what the user is doing. This reduces the
amount of information we need to get directly from the user. Re-
search such as [8, 10] on (often neural network-like) learners that
mimic the acquisition and monitoring of routine sequential action
in humans is related, but does not satisfy most of the requirements of § 2
due to its different purpose. Our knowledge structures are at a higher
level of abstraction, and capture the relations between activities as
well as their motivations to enable user support on the basis of these
higher-level concepts. However, the prob-based transition system is
inspired to Markov Models [13].

Another area has investigated agents that form habits and routines
of their own (see for example [2, 15]). Instead, our interest is in an
agent that supports a human user in dealing with her habits. Knowl-
edge structures oriented towards habit-learning agents need not be
shared or explainable, and consequently the models are not explicit.

The challenge of developing support agents capable of dynami-
cally interacting with humans in complex environments is not new
(e.g., [4, 6, 37]). We share with them the general vision of a support
agent, but the domain of support for dealing with individual habits
was still unexplored. Secondly, while existing support agents in this
tradition build on the notion of an agent whose primary goal is to take
over some of the tasks a human has, the HabInt we propose has no
such purpose. As a consequence, we face some different issues. For
example, the issue of Adjustable Autonomy, as in [22], disappears,
because HabInt does not take over human tasks or responsibilities
but simply automates some of them when requested. On the other
hand, the question of how to time the interventions remains.

In the area of multi-agent systems, knowledge structures for the
representation of goals and actions have been extensively studied
(e.g. [7]). In this paper we show how such structures can be used
as a basis for the representation of habits. In future work we will in-
vestigate to what extent BDI languages can be used as a basis for
implementing HabInt. A difference between HabInt and BDI agents
is that agents execute plans themselves while HabInt represents a
user’s goals and activities in order to support the user in executing
them. Thus a core challenge towards HabInt’s development will be
to study how to these knowledge structures can be constructed in in-
teraction with the user, and to develop further notions and reasoning
techniques for habit support on their basis.

7 Discussion

The literature agrees on habits being no longer consciously goal-
oriented: awareness of the goal has been lost in the process of habit
formation and is no longer an explicit motive, but at most a latent
justification. While in other situations the motive needs to be present
in order to motivate action, habitual behaviours often lose sight of
the motives as soon as they are no longer needed.9 Clearly, turn-
ing carefully deliberated-upon choices into habits is a value-driven
drive of its own, since in so doing we free up time and effort-
consuming deliberations by crystallising them into automatic cue-
response mechanisms (e.g. [21]). But by obscuring the values the
behaviours’ goals used to promote or demote, the process of habit
formation risks making us blind to better choices and pitfalls alike.
HabInt can help to counter this phenomenon by recording explicit
representations of the values that are promoted or demoted by cer-
tain behaviours. The habit-formation process weakens awareness of
how actions affect values and how values motivate goals (cfr. Figure
3) HabInt can help to close the loop, so that the values that are the
motivation and purpose of habitual behaviour can be made visible
again, revealing perhaps its normative aspects.

The dictionary definition of the word norm ([1]) includes: a) A
principle of right action binding upon the members of a group and
serving to guide, control, or regulate proper and acceptable be-


9 This is a simplification: see [34] for a more complete account.
with her desired behavior. In this way, *HabInt* will be able to determine when and how to give the user timely advice.

A key feature of *HabInt* is that it adheres strictly to the user’s vocabulary for expressing goals, activities, actions and values, and that the fine-grainedness of the concretisation relation and of the actions involved in the user’s activities are tailored to the user’s needs. If a reminder to “go by bike” is enough for the user to know what to do, then no additional information is stored by *HabInt*.

In future work we intend to investigate the resemblance to David Lewis’ perfect worlds semantics for deontic logic. We would like to study whether deontic logic techniques would allow the type of reasoning needed to differentiate between actual behaviours, already formulated desired behaviours, and tentative attempts of the user to formulate her actual or desired behaviours.

The current model does not address the temporal dimension of habits. As timely interventions are crucial, the temporal aspects will be addressed in a future paper. Furthermore, here we defined *prob* as a probability function, whereas we would like *HabInt* to reason qualitatively, using notions such as ‘always, often, at least biweekly…’. This challenge is to be addressed in future work, as will the other components of the agent architecture, the implementation and the verification of the validity, scalability and robustness of the approach.

**ACKNOWLEDGEMENTS**

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

**References**


