Requirements for a Temporal Logic of Daily Activities for Supportive Technology

Malte S. Kließ* and M. Birna van Riemsdijk*

Delft University of Technology
Delft, The Netherlands
m.s.kliess@tudelft.nl, m.b.vanriemsdijk@tudelft.nl

Abstract. Behaviour support technology is aimed at helping people organize their daily routines. The overall goal of our research is to develop generic techniques for representing people's actual and desired behavior, i.e. commitments towards themselves and others, and for reasoning about corresponding supportive actions to help them comply with these commitments as well as handle non-compliance appropriately. Describing daily behavior concerns representing the types of behaviour the user typically performs, but also when, i.e. we need to take into account *temporal dimensions* of daily behaviour. This paper forms a first requirements analysis of the types of temporal dimensions that are relevant for the purpose of supporting people's daily activities and how these may be formalized. This analysis forms the starting point for selecting or developing a formal temporal representation language for daily activities.

1 Introduction

Behaviour support technology [10] is aimed at helping people organize their daily routines and change their habits. The overall goal of our research is to develop generic techniques for representing and reasoning about people's actual and desired behavior for the purpose of providing support by means of technology [11, 15]. These expressions of desired behaviour can originate from users themselves or from others in their social context, such as caregivers. The idea underlying our approach is to model desired behaviour as norms [1] or commitments [12] regarding people's behaviour [9, 15, 11, 14]. Reasoning techniques are then aimed at deriving corresponding supportive actions to help people comply with these norms and commitments as well as handle non-compliance appropriately, i.e. with an understanding of the social relations and the values underlying the established agreements [11, 15, 8].

Describing (actual and desired) daily behavior concerns not only *which* types of behaviour the user typically performs, which can be represented in a behaviour hierarchy [11], but also *when* these activities are performed, i.e. we need to take into account temporal dimensions of daily behaviour. This paper forms a first requirements analysis of the types of temporal dimensions that are relevant for

* Supported by NWO Vidi project CoreSAEP.

the purpose of supporting people's daily activities. Once we have an understanding of which notions of temporality we want and need to capture for this type of technology, we can analyze which of many existing frameworks for representing and reasoning about norms and time we can build on, e.g. [6, 4, 2, 3, 7, 14]. This paper forms a preliminary exploration in this direction.

We present an illustrative scenario and preliminaries regarding temporal logic (Section 2). We identify key temporal dimensions of daily activities in Section 3. For each of these dimensions we provide an example formalization in temporal logic of an aspect of our scenario (Section 4). We conclude the paper in Section 5.

2 Scenario and technical preliminaries

2.1 Scenario

In this paper, we will explore examples involving temporal dimensions of daily activities based on the following scenario, where we intend to follow a few examples of daily activities in the fictitious life of Pedro, a young person with mental disabilities who can take care of most daily activities himself, but needs support in order to do them in a timely fashion.

Pedro can usually take care of everyday tasks himself, like getting ready in the morning, getting to work, going shopping and preparing meals. He needs support and regular reminders to schedule these activities. For this he has a support agent, which knows about Pedro's regular activities and preferences.

We intend to address the temporal dimensions of these routines. For example, when we describe our daily routines we would usually put these habits in some order in which we engage in these activities. We would certainly say that we get up before we wash ourselves, and that right next after that we have breakfast. However, there are a lot of subtleties involved when we want to formalize this description in a precise way so that an artificial agent is able to render efficient support for these activities: Is there a specific order in which we prepare the breakfast? Is it necessary to boil the water before we put in the tea bag, and when does this happen compared to preparing the bread we want to eat?

In order to deal with these kind of questions it is not enough for an agent to have an idea of the usual behaviour and habitual order. It also needs to ensure that actions are performed in a coherent way. For instance, putting the kettle on to boil water certainly needs to be done before we pour the water in a cup, but while we wait for the water to boil we can already toast the bread. Putting the kettle on before we go to the bathroom to wash, however, might not be a good idea: if we take too long in the bathroom the water will be too cold for tea, so we should start boiling the water only *after* we finished washing. This means that if we want an artificial agent to help users organize their day in an efficient way, then we need temporal reasoning with which not only the order in which activities are done can be stated, but also temporal 'closeness' to ensure that the user does not wait too long between finishing one step of an activity and starting to do the next step.

2.2 Temporal logic

We will formalize the logical framework in the following definition. We will roughly follow the definitions given in [11] for the HabInt habit support agent.

Definition 1. *Let $\mathcal{L}_{Str}$ be a propositional language over strings. Let $Act \subseteq \mathcal{L}_{Str}$ be the set of activities or actions. Let Part be a function $Act \to \mathcal{P}(Act)$, with $b \in Part(a)$ if the action b is a part*[1] *of doing action a. For the sake of readability we introduce predicates* `start`*,* `stop` *and* `doing` *on Act, signifying when an activity is started, stopped, or being done, but we will still treat Act as atomic. We close these atomic sentences as usual under $\neg$ and $\vee$, and interpret all other logical connectives in the usual way.*

As for the temporal dimensions, we will use notions from Linear Temporal Logic [5] as a way of sketching formalizations for the temporal dimensions discussed. In particular, we will have in mind trace semantics and the usual connectives $\Box$, $\Diamond$, $\mathcal{R}$, $\mathcal{U}$, $\bigcirc$ (to be read as *always*, *eventually*, *Release*, *Until* and *Next*, respectively). We will take this as a form of 'proof of concept', showing that some of the dimensions we have in mind may be expressed using notions of already existing frameworks.

3 Key temporal dimensions

In this section we will explore five temporal dimensions we believe are key to modelling human behaviour for supporting agents. We will provide examples demonstrating how each aspect covers part of an every-day behaviour pattern an agent should be able to support. Note that the examples are intentionally formulated very strictly: if a person states they are having dinner at 6 pm, then this should be formulated in the language just like that. Even though in reality, this will likely almost always be violated[2], it is important for the agent to recognize this. Whether an intervention is necessary, and if so, how the agent should intervene, should be delegated to the Norm Compliance Support system of the agent. Our intention is to provide the necessary formalization to state norms and thus give an agent the ability to detect deviance from these norms.

The five key dimensions are:

1. **Clocktime.** A supporting agent will no doubt have to be able to deal with clocktime. This is particularly important for keeping certain deadlines, as well as ensuring that scheduled actions are executed at the given time.
2. **Ordering.** Some behaviours require multiple distinct actions to be executed in a certain order, e.g. washing vegetables before cutting them. However, human behaviour is much more flexible than a deterministic routine; it is enough to know that vegetables should be cut *after* they have been washed,

[1] For example, brewing tea is an action in its own right, but may also be considered as part of 'preparing breakfast'.

[2] E.g., starting dinner five seconds after 6 pm.

but not necessarily directly after (partial order): it should be acceptable behaviour to wash all of the vegetables first, and then cut them.

3. **Coherence.** When describing behaviour with multiple parts it is not enough to state that they will all eventually be done. When cooking vegetables it is not good enough to wash them today, cut them tomorrow and finally cook them next week. Rather, these actions should be performed *coherently*, i.e. without too much delay. An immediate 'next' step, however, may be too rigid, so we need a notion that allows us to state that a set of actions form a coherent unit.
4. **Duration.** The duration of actions is a two-fold notion. On the one hand certain actions have a fixed or typical duration, e.g. 'pasta takes 8 minutes to cook'. On the other hand, an agent should be able to infer the duration of an action that is being performed, i.e. the actual duration of a specific instance of action execution, so that it can support the user in achieving the desired habitual goals. This becomes important, e.g. when supporting the user with keeping deadlines or making sure they do not exhaust themselves with certain exercises.
5. **Repetition.** The agent should be able to deal with repetition. This involves both regularly scheduled events and the cyclic nature of weeks and days. If a user wants to be supported having dinner at 6 pm every day, then this event regularly resets. The same is true for weekly exercises.

In the usual formalization, LTL semantics consists of a linear trace with one end point, namely the starting point of the trace. However, as one easily sees particularly in the clocktime example we have to deal with cyclic time: every night at 11.59 pm, when the time progresses one minute, the clock resets to 12.00 am, and a new day begins. Similarly, starting the week with Monday, the weekday variable changes each day until Sunday is reached, whereafter the variable resets to Monday again.

Linearity is not completely lost, though: we cannot repeat the actual day or week, but irrevocably progress into the future. The idea is, thus, to model this using two separate notions of time: *micro-time*, which is of cyclic nature and models the time passing each day, and *macro-time*, which is linear and models the fact that each day that has passed is certainly not coming back.

The way we suggest to model this is by using a two-dimensional trace index: we will write each separate point in the trace $\boldsymbol{s}$ as $s_{\langle a,b\rangle}$, where the first component refers to macro-time, and the second component refers to micro-time. We will use lexicographic ordering on the pairs $\langle a, b\rangle$: today at 5 pm is earlier than tomorrow 5 am.

While one can think of macro- and micro-time as separated into days passing and the time of the current day, respectively, this need not be the desired separation: there are larger cycles common in every-day life: weeks, months and years share a cyclic nature as well. It may depend on the specific task or use case how fine-grained one wants to make this separation. For the sake of this scenario, days and time of day are sufficient.

4 Scenario Formalization

In the following examples we will explore how the key dimensions identified above can be formalized in the framework outlined in Definition 1.

Example 1 (Clocktime).

Scenario: Pedro intends to have his dinner at 6 pm every day.

Formalization: Introduce a variable t ranging over time. Abusing notation, we will write $t = 6pm$ to mean that t has the value corresponding to real-world time of 6 pm. Then the statement above can be formalized as

$$\Box\Big(t = 6pm \rightarrow \texttt{start}(\mathit{HavingDinner})\Big).$$

Discussion: The statement above is very sharp: each day there is precisely one point in time when t has the value of 6 pm. So how do we then deal with situations when Pedro starts his dinner slightly earlier or later? We would argue here that this may be deferred to norm compliance: if we casually state 'I have my dinner each day at 6pm.', then we would argue this is meant precisely in the sense of the formal statement above.

Example 2 (Ordering).

Scenario: Pedro wants to bake a pizza for dinner. He already has a pizza dough prepared, and wants to decorate the pizza with tomato sauce, mushrooms and cheese.

Formalization: We want to express certain orders in the execution of an action, when that action is itself composed of smaller parts. As an example, take baking a pizza. For that we have to first prepare all ingredients, then put the toppings on the pizza, and finally put it in the oven. There is a strict ordering implicit here: we cannot[3] put the pizza in the oven and then afterwards put the (unprepared) toppings on the pizza dough, and finally prepare the toppings by washing and cutting them.

Take the LTL-formulation of *before*, as in e.g. [13], i.e. ϕ *before* $\psi \equiv \neg\,(\neg\phi\,\mathcal{U}\psi)$, where $\mathcal{U}$ is *Until*. Then the situation in the scenario above can be formalized by

$$\texttt{apply\_tomato\_sauce}\ \mathit{before}\ \texttt{decorate\_with\_mushrooms}, \tag{1}$$

$$\texttt{apply\_tomato\_sauce}\ \mathit{before}\ \texttt{decorate\_with\_cheese}, \tag{2}$$

$$\texttt{decorate\_pizza}\ \mathit{before}\ \texttt{bake\_pizza}. \tag{3}$$

Discussion: As an alternative formulation one could think of using *after*: ϕ *before* ψ should be (almost) the same as ψ *after* ϕ – both give a temporal link between ϕ and ψ, ϕ is the first statement that should come true, and ψ the second. Switching the roles of ϕ and ψ and negating the statement, however, will not turn *before* into *after*: we would be introducing the possibility of synchronicity, i.e. ϕ and ψ being executed simultaneously. Another problem is what to do once the second event, ψ, has occurred. From the definition using $\mathcal{U}$, this

[3] Or rather: should not attempt, as it gives undesired results.

does not prevent us from witnessing another ϕ later. While we certainly do not want to prevent Pedro from ever washing mushrooms again, the same mushroom should not be washed again once it has been cut.

Example 3 (Coherence).

Scenario: We will use the same scenario as for example 2, and focus on what it means to be 'coherent'.

Formalization: Using the example of preparing a pizza for dinner, what does it mean to 'coherently bake a pizza'? The dough has to be spread out, the ingredients put on it, and finally the pizza needs to be baked. While we also have the constraints that the toppings should first be washed, then cut, then put on the dough, we face the same issue of multiple instances as above: do we allow each mushroom to be washed, and then cut them all, or individually first cut, then wash, and repeat the process for each mushroom until we are done? We would argue here that this should not make a difference for the process of 'baking pizza', as long as we are not putting other activities in between, e.g. taking out the trash. So 'coherent' should mean that we do not engage in unrelated activities.

This seems to suggest an 'Axiom of Coherence':

$$(\text{Coh}) : \forall a \in \mathit{Act}.\ [\texttt{doing}(a) \to (\neg\texttt{wait}(a) \to \forall b \notin \mathit{Part}(a).\ \neg\texttt{start}(b))]\,.$$

This statement says that as long as we do not have some mandatory waiting time on our hand, e.g. when the pizza is in the oven, we should not start any other unrelated activity.

Discussion: 'Coherence' should also mean 'close together temporally', which is currently not encoded in the Axiom above. We can still have arbitrary idle time between washing the mushroom and putting it on the pizza. Setting a strict time limit for the distance between stopping some part of an activity and starting the next one may be too strict. In the pizza example, we may get some help via deadlines to solve this: if Pedro wants to have pizza for dinner, and have his dinner at 6 pm tonight, then this deadline already forces him to have shorter breaks. The question is then whether Coherence actually is a derived notion, and the Axiom stated above is implicitly given by the other temporal dimensions.

Example 4 (Duration).

Scenario: Pedro usually takes 20 minutes to prepare a pizza, and then puts it into the oven for 20 minutes. Thus the activity '*make_pizza*' has a duration of 40 minutes.

Formalization: We can equip any action a with an attributed duration $d \in \mathbb{N}$, given in minutes, to form the ordered pair $\langle a, d\rangle$. Assuming that no action can be done instantaneously, we allow $d = 0$ to indicate that no duration has been stated for the current action.

In the scenario above, we would then have $\langle \mathit{make\_pizza}, 40\rangle$ in the model, stating that preparing pizza has a duration of 40 minutes.

For the model, assume we are looking at traces[4] $\boldsymbol{s} = \langle s_0, s_1, \ldots, s_i, \ldots \rangle$, where s_i is the state of the model at minute i, and s_0 is the initial state. Then 'preparing pizza takes 40 minutes' would be modeled by

$$s_i \models \texttt{start}(\mathit{make\_pizza}) \Rightarrow \exists j \geq i + 40 \,.\, s_j \models \texttt{done}(\mathit{make\_pizza}).$$

Discussion: This example sees duration as an explicit notion present in the model. One could also see it as an implicit notion, where 'duration' is nothing but the difference of the two end-points of an activity, i.e. the difference between $\texttt{start}(a)$ and $\texttt{stop}(a)$. The explicit notion is needed for planning and norm compliance: if the agent knows that baking a pizza takes 40 minutes, then this needs to be started at least 40 minutes before the planned dinner time; similarly, the norm of 'having dinner at 6 pm' is certainly violated if Pedro starts preparing the pizza at 10 to 6. How does the implicit notion fit in here? Is it just used to update the explicitly given duration by experience, or as a monitoring tool to make sure the pizza stays in the oven for just the right time? Or does it have some importance in its own right, other than being checked against the recorded duration while doing an instance of an activity?

Example 5 (Repetition). Scenario: Pedro has a 'Pizza Day': every Monday he has pizza for dinner.

Formalization: Introduce a variable D for the weekday. Then we can formalize the above by:

$$\Box(D = \text{Monday} \;\rightarrow \texttt{pizza\_for\_dinner}).$$

Discussion: There seems to be nothing much needed for this except for variable types that allow access to the trace time. One could expand this example to 'pizza every other week', in which case one can then check against the past week. Letting W range over weeks, we then would obtain $(W = n \wedge \texttt{pizza\_for\_dinner}) \rightarrow (W = n + 1 \wedge \neg\texttt{pizza\_for\_dinner})$, and similarly swapping the negation in front of $\texttt{pizza\_for\_dinner}$ on both sides of the implication.

5 Discussion and future work

The examples above can all be expressed using LTL, using a two-dimensional index for the temporal trace; essentially, it is still discrete, linear, but it has the added benefit that we are now able to code daily routines using the second component of the trace index only. This suggests that LTL may be a suitable language for encoding temporal aspects of everyday activities. However, there are still many open questions such as those discussed in Section 4. In particular, we have not adressed the issue of expressing 'temporal closeness', or: 'how late is too late?'. A hard restriction on temporal distance between two activities that should be performed 'closely together' may be too strict for practical purposes. On the other hand, this may be an issue that can be dealt with by defining

[4] For the sake of simplicity, we omit the 'macro-time' component for the moment, looking at a one-dimensional trace.

norm compliance appropriately. Using a temporal language for reasoning about compliance support may require adaptations to the temporal language to make it tractable. As in [14] we use LTL without deontic operators. Whether this suffices for expressing norms regarding desired daily behaviour is also to be explored in future research.

Acknowledgements. We would like to thank two anonymous referees and the participants of the CARe-MAS workshop for insightful comments and discussions that helped improve the work presented in this paper.

References

1. G. Andrighetto, G. Governatori, P. Noriega, and L. van der Torre, editors. *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.
2. J. Broersen, F. Dignum, V. Dignum, and J.-J. Ch. Meyer. Designing a deontic logic of deadlines. In *Proceedings Seventh International Workshop on Deontic Logic in Computer Science (DEON'04)*, volume 3065 of *LNCS*, pages 43–56. Springer-Verlag, 2004.
3. S. Cranefield. A rule language for modelling and monitoring social expectations in multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems (ANIREM'05 and OOOP'05)*, volume 3913 of *LNCS*, pages 246–258, 2006.
4. F. Dignum and R. Kuiper. Specifying deadlines with dense time using deontic and temporal logic. *International Journal of Electronic Commerce*, 3(2):67–86, 1998.
5. E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 996–1072. Elsevier, Amsterdam, 1990.
6. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proceedings of the 20th Australian joint conference on Advances in artificial intelligence*, pages 486–496. Springer-Verlag, 2007.
7. K. V. Hindriks and M. B. van Riemsdijk. A real-time semantics for norms with deadlines. In *Proceedings of the twelfth international joint conference on autonomous agents and multiagent systems (AAMAS'13)*, pages 507–514. IFAAMAS, 2013.
8. A. Kayal, W.-P. Brinkman, R. Gouman, M. A. Neerincx, and M. B. van Riemsdijk. A value-centric model to ground norms and requirements for epartners of children. In *Coordination, Organizations, Institutions, and Norms in Agent Systems IX (COIN'13)*, volume 8386 of *LNCS*, pages 329–345. Springer, 2014.
9. A. Kayal, W.-P. Brinkman, H. Zoon, M. A. Neerincx, and M. B. van Riemsdijk. A value-sensitive mobile social application for families and children. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 22nd Conference on User Modeling, Adaptation, and Personalization (UMAP'14)*, volume 1181. CEUR, 2014.
10. H. Oinas-Kukkonen. Behavior change support systems: A research model and agenda. pages 4–14, 2010.

11. P. Pasotti, M. B. van Riemsdijk, and C. M. Jonker. Representing human habits: towards a habit support agent. In *Proceedings of the 10th International workshop on Normative Multiagent Systems (NorMAS'16)*, LNCS. Springer, 2016. To appear.
12. M. P. Singh. An ontology for commitments in multiagent systems: toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
13. M. B. van Riemsdijk, L. Dennis, M. Fisher, and K. V. Hindriks. Agent reasoning for norm compliance: a semantic approach. In *Proceedings of the twelfth international joint conference on autonomous agents and multiagent systems (AAMAS'13)*, pages 499–506. IFAAMAS, 2013.
14. M. B. van Riemsdijk, L. Dennis, M. Fisher, and K. V. Hindriks. A semantic framework for socially adaptive agents: Towards strong norm compliance. In *Proceedings of the fourteenth international joint conference on autonomous agents and multiagent systems (AAMAS'15)*. IFAAMAS, 2015.
15. M. B. van Riemsdijk, C. M. Jonker, and V. Lesser. 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'15)*, pages 1201–1206. IFAAMAS, 2015.