Verifying properties of activities of daily living

Verifying properties of Activities of Daily Living

Ricardo Contreras, Filip Smola, Jiawei Zheng, Jane Hillston, and Jacques D. Fleuriot

School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
{r.contreras,f.smola,jw.zheng,jane.hillston,jdf}@ed.ac.uk

Abstract. The monitoring of Activities of Daily Living (ADLs) for older adults can support their well-being since deviations in behaviour may indicate cognitive or physical decline, and reduced quality-of-life. In this work we present a new methodology that combines the spatial context with the observed behaviour of the person to enable a rigorous analysis of their daily activities. Tailored models are constructed that capture the layout of the individual's home and incorporate timestamped data from unobtrusive sensors attached to everyday objects. We specify expected behaviours as properties encoded in linear temporal logic and use model checking to assess whether sensor-captured behaviours align with expectations. Importantly, our methodology is agnostic to different forms of ADLs and behaviour specifications.

1 Introduction

The life expectancy of adults has risen over the past decades thanks to advancements in medical science and care-related technologies [4]. Although people are living longer, it is increasingly common for them to experience an array of long-term health conditions in their later years. Consequently, if we are to optimise quality-of-life, the care for adults in later life needs to balance independence and support [3].

Activities of Daily Living (ADLs) are self-care tasks that are essential for maintaining health and well-being. Unexpected behaviour during ADLs may relate to a decline in cognitive or physical abilities, that in turn can result in unsafe conditions [16].

By modelling the various components involved in carrying out ADLs as a finite state model, we can form an abstraction of an individual's behaviour. This enables the application of formal methods to verify whether the captured behaviour satisfies a set of logical formulae representing desired properties. Model checking [7], in particular, facilitated by tools such as NuSMV [6] and PRISM [13], provides a powerful way of ensuring the correctness and reliability of systems in various domains.

In this work we present a methodology for representing and reasoning about ADLs. We construct a model of an older adult's home, which includes the layout

and sensors they interact with. We incorporate the observed behaviour of the person as part of the model and use the symbolic model checker NuSMV to evaluate whether the behaviour conforms to properties encoded in a dialect of Linear Temporal Logic (LTL) [15]. The properties are extracted from existing health and safety guidelines from a variety of different institutions, including governments [8] and private institutions [9]. To manage the computational cost, we use bounded model checking to limit the number of system states explored during verification and we improve the performance of our approach by pruning non-relevant states, based on the observed behaviour.

The main contribution of this paper are as follows:

- We develop a methodology for monitoring ADLs that combines the spatial context with the observed behavior of an individual to create a comprehensive discrete-time model of an older adult's home and their physical interactions with objects and appliances.

- We introduce symbolic model checking in the domain of ADLs, using NuSMV as an example tool to verify observed behaviours, captured using unobstrusive sensors, against specified properties. When a violation happens, we use the counter-example generated by the model checker to visualise where and how it occurred.

- We formalise a set of ten care-related properties from established guidelines. While space constraints prevent the inclusion of all properties, the examples demonstrate the applicability of our methodology. Additionally, we include properties related to physical constraints, which are used for the validation of the data model.

- We evaluate the methodology on real data from another study, demonstrating its applicability and providing a basis for creating more accurate models of a home environment.

- We demonstrate that the process of verification can be made faster by using bounded model checking and pruning our discrete-time model to remove irrelevant sections based on the observed behaviour.

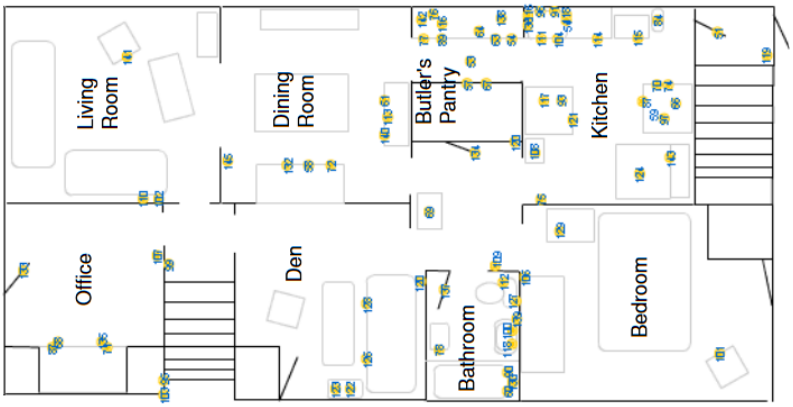
This methodology is specifically aimed at people living alone with the overall goal of identifying non-compliant behaviours of the individual over time.

This paper is structured as follows: Section 2 describes the preliminaries. Section 3 introduces our methodology for the development of personalised models. Section 4 describes the implementation and presents the results of evaluating our approach on an existing dataset. Section 5 discusses how other approaches relate to our work. We conclude in Section 6 with our final remarks.

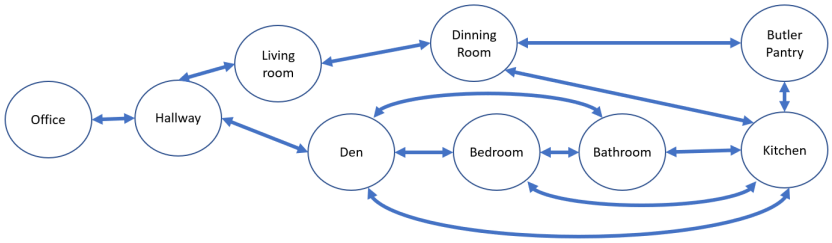
2 Preliminaries

In this section, we outline the preliminary aspects. This includes the description of the dataset and the assumptions made about the data. Our methodology assumes that we have access to data captured by different ambient sensors affixed to everyday home objects in various locations [21], which can be mined to elicit the activities the subject engages in.

In this paper, to illustrate our approach, we use a real-world dataset published by Munguia et al. [17, 22] involving an 80-year-old woman, from now on referred to as Subject A, living alone in a house fitted with 70 ambient sensors. Note that there are no motion detection sensors. The sensor data was collected over a period of two weeks and is timestamped. We use these timestamps to determine the order in which sensors are activated and deactivated. Note that we do not currently consider duration between events – this will be part of future work (see Section 6).



(a) Floorplan



(b) Graph representing the layout, with the Hallway sitting between the Office, Den and Living Room shown explicitly

Fig. 1: Concrete and abstract layouts of Subject A's house

We assume that our data is a time series of events $e_1, \ldots, e_n$, also called a *trace*, and that these events can be partitioned into activation and deactivation ones, denoted by e_i and e_j respectively. Let $t(e)$ denote the time and $s(e)$ denote the sensor for a given event e. We make the following further assumptions about the data:

- For every activation event e_i there is a corresponding deactivation one, e_j and no subsequent activation can occur before this deactivation.
- For corresponding activation and deactivation events e_i and e_j, activation always precedes deactivation. That is, $t(e_i) < t(e_j)$.
- For all sensors, the initial state is off.

Quick Overview of LTL

As we are dealing with time series data, we are generally concerned with modelling behaviour over time. This makes Linear Temporal Logic (LTL) [15] apt for our work as it allows us to specify properties we are interested in through the use of various temporal operators.

LTL is a formalism used in formal verification to reason about temporal properties in transition systems. We use LTL extended with past-time operators [1], henceforward referred as past Linear Temporal Logic (pLTL). In Section 3.4 we show how pLTL can be used to specify health and safety properties, which are then checked against a model of an individual's ADLs and their home.

In pLTL we construct formulas and then check whether all execution paths (sequences of states) of the system satisfy those formulas, where each state in the path is labelled with the atomic propositions that it satisfies. A formula in pLTL consists of atomic propositions, logical operators and temporal operators. Formally, the formulas are generated according to the following BNF grammar:

$$\phi, \psi ::= p \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \rightarrow \psi \mid X\phi \mid F\phi \mid G\phi \mid \phi\, U\, \psi \mid \phi\, R\, \psi \mid O\phi$$

Satisfaction of a formula ϕ by a path σ at time index i is denoted by $\sigma, i \vDash \phi$. An atomic proposition is satisfied if the state at that index of the trace is labelled with it. The non-temporal logical operators $\neg$, $\vee$, $\wedge$ and $\rightarrow$ have the same meaning as in propositional logic. For instance, we have $\sigma, i \vDash \phi \wedge \psi$ if and only if both $\sigma, i \vDash \phi$ and $\sigma, i \vDash \psi$.

Some of the temporal operators of pLTL, relevant to the current paper, are described in Table 1. This selection does not preclude the use of other operators; rather, it reflects those that are specifically chosen for the properties discussed here, recognising that different properties may require different operators.

An example property, expressed in PLTL, is given by (1). This specifies physical constraints based on the layout in Figure 1(b) and states that if the person is currently in the living room (pos_{LR}) and was previously in the kitchen ($O\, pos_{Kitch}$), then they must have also been to the hallway ($O\, pos_{Hall}$) or the dining room

Operator	Interpretation
Next (X)	$X\phi$ is satisfied if ϕ is satisfied at next time step That is: $\sigma, i \vDash X\phi$ if $\sigma, i+1 \vDash \phi$
Eventually (F)	$F\phi$ is satisfied if ϕ is satisfied at *some* possibly future time step That is: $\sigma, i \vDash F\phi$ if $\sigma, j \vDash \phi$ for some $j \geq i$
Globally (G)	$G\phi$ is satisfied if ϕ is satisfied at *every* possibly future time step That is: $\sigma, i \vDash G\phi$ if $\sigma, j \vDash \phi$ for every $j \geq i$
Until (U)	$\phi\, U\, \psi$ is satisfied if ψ is satisfied at some future time step and ϕ is always satisfied until then That is: $\sigma, i \vDash \phi\, U\, \psi$ if $\sigma, k \vDash \psi$ for some $k \geq i$ and $\sigma, j \vDash \phi$ for every $i \leq j < k$
Release (R)	$\phi\, R\, \psi$ is satisfied if ϕ is satisfied up to the next time ψ is satisfied That is: $\sigma, i \vDash \phi\, R\, \psi$ if either $\sigma, k \vDash \psi$ for some $k \geq i$ and $\sigma, j \vDash \phi$ for every $i \leq j \leq k$, or if $\sigma, j \vDash \phi$ for every $j \geq i$
Once (O)	$O\phi$ is satisfied if ϕ is satisfied at some past time step That is: $\sigma, i \vDash O\phi$ if $\sigma, j \vDash \phi$ for some $j \leq i$

Table 1: Sample of temporal operators used in pLTL

($O\, pos_{Din}$). This is, in the states representing the positions between the living room and the kitchen.

$$G(pos_{LR} \wedge O\, pos_{Kitch} \rightarrow O((\neg pos_{LR} \wedge \neg pos_{Kitch})\, U\, (pos_{Hall} \vee pos_{Din}))) \quad (1)$$

3 Methodology

In this section, we outline our methodology which is centred on developing personalised models, using contextual and collected data about an individual's ADLs and their home. The constructed model is amenable to formal verification using model checking against care guidelines formally encoded in temporal logic. Figure 2 illustrates the main components of our approach, which we detail next.

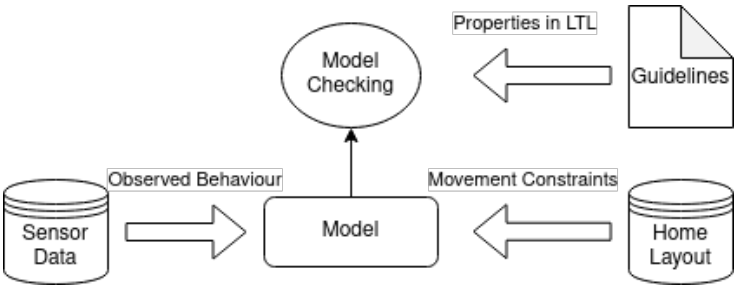


Fig. 2: Framework of our approach

3.1 Layout and Subtraces

The contextual information about the layout of the house plays a crucial role since it relates to the movement of the individual. We model the floor plan as a directed graph, where each node corresponds to a room and the edges represent the connections between rooms. Figure 1(b) depicts the graph for the home being considered in our dataset [17, 22].

Subtraces and states Typically there will be a large number of sensors generating data into the trace of activities (70 in the case of Subject A) but most properties will be focused on a small set or single sensors. We use subtraces to talk about events occurring between the activation and deactivation of a sensor. Consider a trace π for a set of sensors S. Given a pair of corresponding activation e_i and deactivation e_j for some sensor $s \in S$, the subtrace for that activation period is the set of all events, chronologically ordered, from the activation until the deactivation:

$$\text{subtrace}(\pi, s, e_i, e_j) = \left\{ e_k \in \pi \;\middle|\; \begin{array}{l} t(e_i) \leq t(e_k) \leq t(e_j), \\ s(e_i) = s(e_j), \\ s(e_k) \neq s(e_i) \text{ for all } e_k \notin \{e_i, e_j\} \end{array} \right\}$$

Note that our already mentioned assumptions about the data ensure that an activation occurs before its corresponding deactivation, so every such subtrace is non-empty.

We collect subtraces for all activation periods of a given sensor $s \in S$ in a truncated trace, denoted by $trunc(\pi, s)$, which is used when constructing our model (see Algorithm 1).

We also define a vector $V(t)$ to represent the state of all sensors at time t. An activated sensor is taken to be on until it is deactivated (off). In line with our assumptions about the data, all states in $V(0)$ are initialised to off (False). States are updated according to the occurrence of activation and deactivation events. For example, in Figure 3 at $V(t(e_j))$ only sensor s_2 is on (True).

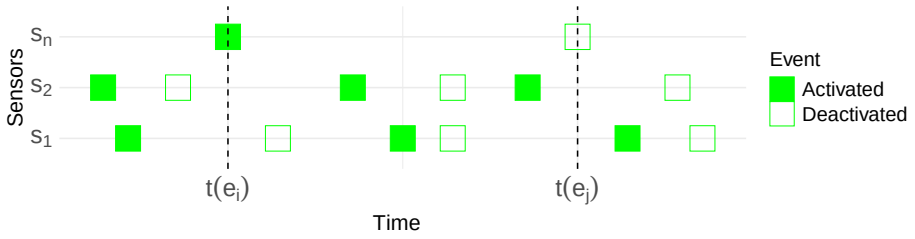


Fig. 3: Subtrace for the activation $t(e_i)$ and deactivation $t(e_j)$ of s_n

3.2 Static Model

We use our graph layout of the house and the assignment of sensors with respect to its rooms to build what we call the *static* model. We define the overall state of a room based on its sensors, which helps in the simplification and abstraction of the model. In what follows, we provide excerpts of our formalisation using NuSMV [6] as an example of the tools we could use to build the model. These specifications are provided for illustrative purposes; a comprehensive understanding of the model details is not required to understand the methodology. However, our approach is general and the model could be formalised and checked with other verification tools.

NuSMV can represent finite state systems and employs symbolic model checking techniques to explore their state space. In its setting, a model, i.e. a transition system, includes the structure and behaviour of the system. More specifically, a model consists of i) states, representing possible configurations during the execution of the system, ii) transitions, stating how the system evolves from one state to another, and iii) the initial conditions of the system. A model is verified against a set of logically-encoded properties.

When a property is violated, NuSMV generates a counter-example that is useful in explaining where and how the violation of a property can occur within the model. A counter-example is a specific sequence of states that leads to the failure of the property. It shows the exact conditions under which the property fails, making it easier to identify and understand the problem in the model.

Using model checking offers two distinct approaches i) it facilitates a thorough exploration of all potential states and executions within the model and ii) it allows us to verify a specific execution trace (observed behaviour) by constraining the model to follow particular execution paths. While the comprehensive analysis is crucial for ensuring that properties such as physical constraints are maintained at all times, the trace-based approach is beneficial for checking that the observed behaviour does not violate the encoded properties.

Movement In our model we need to capture the movement of a person within the house. We declare the variable `person_pos` to represent the position of the person within the house and specify their next possible positions based on the graph layout. Thus, from position x it is possible to move to position y if and only if there is an edge from node x to node y in the graph. In the model, the variable `person_pos` restricts the person to be in one room at a time. Listing 1 shows an excerpt of the code in NuSMV modeling the position of the person, based on the layout in Figure 1(b).

```
person_pos: {bathroom, bedroom, den, kitchen, ... , office};
...
next(person_pos) := case
  person_pos = bathroom :{bedroom, den, kitchen};
  person_pos = bedroom :{bathroom, den, kitchen};
  ...
esac;
```

Listing 1: Excerpt of the next possible positions from `bathroom` and `bedroom`

Sensors For each sensor we capture three variables: its state, activation and deactivation. In NuSMV, the `state` of a sensor is a boolean variable where True indicates the sensor is on (e.g. *light on*) and False indicates the sensor is off. In the component corresponding to a sensor, the `activation` boolean variable is set to `TRUE` at the time step when the `state` transitions from `FALSE` to `TRUE`, and it remains `TRUE` for one time step only. Note that while `activation` represents interaction with the person and thus serves as evidence of their activities, the state of a sensor *being* on — which can last for more than one time step — does not. The `deactivation` boolean variable is set to `TRUE` at the time step when the `state` transitions in the opposite direction, also remaining `TRUE` for one time step only. Similarly, while `deactivation` represents interaction by the user, the state of a sensor *being* off does not. Listing 2 shows an excerpt of the NuSMV code used to encapsulate these aspects of a sensor. Note that the variables are initialised to `FALSE`, which aligns with our assumption about each sensor's initial state.

```
ASSIGN
  init(state) := FALSE;
  init(activation) := FALSE;
  init(deactivation) := FALSE;
  next(activation) := ! state & next(state);
  next(deactivation) := state & ! next(state);
```

Listing 2: Sensor variables: state, activation and deactivation

We infer the position of a person from the activation or deactivation of sensors. To allow for this we assume that a sensor can only be triggered when the person is in the room where that sensor is located. In the model, we ensure this by interpreting a change in the sensor's `state` as an indication that the `person_pos` variable corresponds to the room where the sensor is located. For instance, a fridge sensor activation or deactivation in the kitchen would require `person_pos = kitchen` to hold. A sensor triggered in a different room from the one the person is in would require a change in `person_pos` to match that other room. It is also assumed that the person moves along the shortest path between two con-

secutive activations or deactivations of sensors located in different rooms. So, for example, a fridge sensor activation in the kitchen followed by a TV sensor deactivation in the living room would require the following: `person_pos = kitchen` to hold when the fridge is activated followed by `person_pos = dining room`, then `person_pos = living room` followed by the TV sensor being deactivated.

We also assume that only one sensor can be triggered at a time. To represent this, we further constrain changes to a sensor's `state` with the `state` of all other sensors in the same room remaining unchanged. Note that we only need to use sensors in the same room, because sensors in other rooms will not change due to `person_pos`.

Room states and changes In the model, sensors are grouped according to the room they belong to. We concatenate the states of all sensors within a room, and take this to represent the state of that room. We then compare this concatenated state over two consecutive steps and if it differs then there must have been a change in that room, meaning that the person is in that location. However, if over two consecutive steps the changes of state occur in different rooms, then there must have been movement of the person between those rooms. Listing 3 shows an excerpt of the code used to compare changes in a room based on its current and next configurations.

```
DEFINE
  config := room_configuration;
VAR
  change: boolean;
ASSIGN
  init(change) := FALSE;
  next(change) := config != next(config);
```

Listing 3: Changes in a room

We note here that our core model is generated automatically based on the layout and the positioning of the sensors within it. Room states and transitions are derived through the concatenation of sensors' states. The complete model also incorporates the collected sensor data.

3.3 Observed Behaviour

A subtrace corresponds to a chain of events captured by different sensors, which is then included in the static model to represent the behaviour of the person (see Algorithm 1 in Section 4). We do this to filter the possible execution paths to a single one that satisfies the constraints imposed by the subtrace.

Based on the length of a given subtrace, we specify a sequence in pLTL of the same length as the subtrace using the Next temporal operator. For instance,

sequences of length two and three are represented by pLTL statements shown in (2) and (3) respectively.

$$X\ \text{var1} \wedge XX\ \text{var2} \quad (2) \qquad X\ \text{var1} \wedge XX\ \text{var2} \wedge XXX\ \text{var3} \quad (3)$$

Two consecutive events in pLTL — Three consecutive events in pLTL

Using a built-in tool of NuSMV, **ltl2smv**, the pLTL sequence is converted into NuSMV syntax and included as a module in the model. Listing 4 shows an excerpt of the conversion of the sequence shown in (2) into NuSMV syntax.

```
MODULE sequence2(var1, var2)
VAR
  LTL_0_SPECF_2 : boolean;
  LTL_0_SPECF_1 : boolean;
DEFINE
  LTL_0_SPECF_0 := (!(!var1 | LTL_0_SPECF_2) | LTL_0_SPECF_1);
INIT
  !LTL_0_SPECF_0
TRANS
  next(LTL_0_SPECF_0) = LTL_0_SPECF_1
...
```

Listing 4: Module excerpt in NuSMV for instantiating two consecutive generic events

The sequence module will be instantiated with a subtrace when constructing the model (see Algorithm 1). For instance, the subtrace of length two capturing the kitchen light being turned on then off will result in the following module instance being added to the model: `sequence2(kitchen_light1.activation, kitchen_light1.deactivation)`.

3.4 Guidelines and pLTL specification

Given the variability among people and environmental contexts, establishing a standardised framework for the verification of ADLs becomes quite complex. We use health and safety guidelines to specify properties that dictate the expected behaviours during ADLs.

A variety of different institutions, from governments to private facilities, issue such guidelines for older adults at home and care in general. Health and safety guidelines, henceforward referred to as *guidelines*, are expressed in natural language, usually lack specificity and allow for multiple interpretations. In this work, we focus on guidelines that i) specify the behaviour within a home, and ii) describe behaviour that can be captured by unobtrusive sensors. We do not consider guidelines that are too vague or open to multiple interpretations, such as “pay attention when cooking”.

Our selected guidelines are manually encoded as pLTL properties, with Listing 5 showing an example about showering [20]. This advises against leaving a shower turned on and running without someone present to supervise it. Once encoded as a pLTL property, it is used to verify that when the shower faucet 2 (`bathroom_shower_faucet2`) is turned on there is no change in any other room until that faucet is off. The property includes an additional change restriction, `bathroom_excl_shower`, covering a subset of all sensors in the bathroom. This additional property condition allows the older adult to interact with the shower faucets, e.g. regulating temperature, while adhering to the guideline.

```
G (bathroom_shower_faucet2.activation →
   bathroom_shower_faucet2.deactivation R
    (!bedroom.change ∧ !butler_pantry.change ∧ !den.change ∧
    !dining_room.change ∧ !hallway.change ∧ !kitchen.change ∧
    !living_room.change ∧ !office.change ∧ !bathroom_excl_shower.change)
```

Listing 5: Guideline "Never leave a running shower unattended" encoded in pLTL, which will be referred to as Property Shower 2

We have identified additional guidelines that are used to define properties related to ADLs such as cooking [18], and the use of the television [8] and other appliances [9]. Due to space constraints, we do not elaborate further on these here. Similarly, the approach for choosing the properties to verify from the available guidelines is not discussed and is outside the scope of the present work.

In addition to the guidelines we also encode properties related to physical constraints such as (1), which we discussed in Section 3.3. These are used for model validation.

4 Implementation and Evaluation

We now present the implementation of our methodology and the results obtained when evaluating our model with the dataset from Munguia et al. [17, 22]. We made no alteration to the dataset apart from the removal of repeated fictitious dates[1].

4.1 Model construction

Subtraces within the dataset are identified based on a set of *sensors of interest*, which are chosen according to their relevance to a given property. For instance, the guideline in Listing 5 has `bathroom_shower_faucet2` as its only sensor of interest, because violations of the property can only occur when it is on. For each sensor of interest s we identify all its subtraces $trunc(trace, s)$ (see Section 3.1).

The main steps of adjusting and verifying a model are shown in Algorithm 1. For each subtrace, we add it to the model and then verify whether the model complies

[1] We identified repeated data entries with erroneous dates listed as April 31st.

Algorithm 1: Adjusting the model to observations and verifying a given property

```
   Input: staticModel, trace, property
1  allSubtraces ← {};
2  foreach s ∈ sensors of interest for property do
3      add trunc(trace, s) to allSubtraces;
4  foreach subtrace ∈ allSubtraces do
5      prunedModel ← prune(staticModel, subtrace);
6      instantiate sequence module with subtrace;
7      add instantiated sequence to prunedModel;
8      set initial sensor state in prunedModel;
9      verify property on prunedModel with bound of length(subtrace);
```

Algorithm 2: Pruning a Model, $prune(model, subtrace)$

```
Input: model, subtrace
prunedModel ← model;
foreach s ∈ sensors of model do
    if s ∉ sensors occurring in subtrace then
        Remove s from the variables, layout and room states of prunedModel;
return prunedModel;
```

with the given property. In the algorithm, the input *staticModel* corresponds to the model with all sensor variables, layout and room states (see Section 3.2). The pruning is specified in Algorithm 2. Lines 5 to 9 correspond to the following steps.

First, the model is pruned by removing sensors that have no events in a given subtrace leading to a simpler and more compact representation. We remove all such sensors from the variable definitions and room configurations, reducing the size of the model. Pruning improves performance during verification while obtaining the same results compared to the original model. Figure 4 shows the execution time of the model with all the sensors compared to the pruned model.

Second, the subtrace is used to instantiate the sequence module which is then integrated into the core model, see Section 3.3 for more details.

Third, the states of all sensors are identified at the time step prior to the start of the subtrace. For a subtrace starting just after event e, the sensor states would be represented as a vector $V(t(e))$ (see Section 3.1), which is used to initialise the sensor states in the model. This enables the verification to be conducted on a model that reflects the system's state at the start of the relevant subtrace.

Finally, when verifying the property we set a bound on the number of transitions of the model to match the length of the trace. We leverage NuSMV's bounded model checking [2] capability to address any potential state explosion

Sensors of interest	Subtraces	Comply (%)
`bathroom_shower_faucet1`	1	100
`bathroom_shower_faucet2`	189	98
`bedroom_tv`	17	94
`hall_door`	36	92
`kitchen_burner`	12	17
`kitchen_freezer`	20	80
`kitchen_refrigerator1`	16	88
`kitchen_refrigerator2`	247	100
`kitchen_toaster`	56	95
`living_tv`	57	93

Table 2: Sensors of interest and percentage of the models with added subtraces that comply with properties

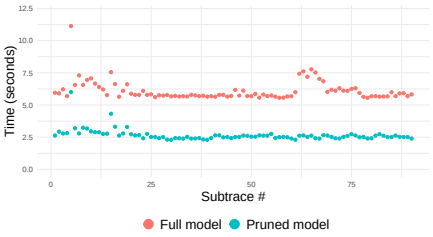


Fig. 4: Execution times for a sample of 90 subtraces involving sensors `kitchen_burner` and `bathroom_shower_faucet2` for the full and pruned models

problem since it then only exhaustively analyses possible execution paths up to the specified limit, rather than exploring the complete state space of the model.

4.2 Experiments and Results

For each property, a log is kept for all subtraces identified from the sensors of interest. For each subtrace it includes i) the sequence of events, ii) the vector $V(t(e))$ with the initial states of the sensors and iii) the output from NuSMV which indicates whether the model is compliant with the property.

We implement a graphical widget to visualise violations of the property being examined, as shown in Figure 5. This displays the counter-example generated by the model checker, including the subtrace and the violating event. In Figure 5, a violation of the property related to the shower running and being left unattended (see Listing 5) is triggered by turning the shower in the bathroom on and then opening the rubbish bin in the kitchen.

For the experiments, we select 10 properties identified from health and safety guidelines. These properties are formally represented in pLTL (just like the one shown in Listing 5) and from these the sensors of interest are identified. Based on the chosen properties we get 651 subtraces in total. Table 2 shows the selected sensors of interest along with the number of subtraces pertaining to each. In the table, the *Comply* column shows the percentage of subtraces for which the model is compliant with the property. While the properties concerning movement and layout constraints are not included in the table, we observed that all such properties were satisfied in the experiments.

In the experiments we observed variations in the satisfaction of the properties. The model consistently satisfied properties for some traces, e.g. 100% compliance for subtraces about sensor `kitchen_refrigerator2`, where the Property *Refrigerator 2* specified that the refrigerator door should be closed after it has

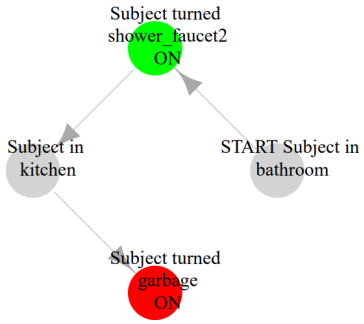


Fig. 5: Visualised counter-example for Property *Shower 2*

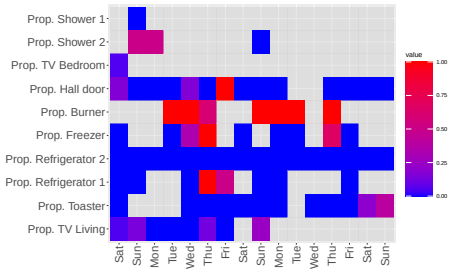


Fig. 6: Ratio of property satisfaction over the duration of the study

been opened. This indicates the person does not tend to forget to close it. In others, the compliance of the model with properties was rarely observed, e.g. 17% of subtraces for sensor `kitchen_burner`, where the Property *Burner* specified not leaving cooking unattended. To gain a more comprehensive understanding of the behaviour we explored the satisfaction of properties over the duration of the study. Figure 6 depicts the properties verified in our model over time where the x-axis denotes the chronological sequence of days and the y-axis denotes the properties related to the sensors previously shown in Table 2. This shows the percentage of time that a property is satisfied within subtraces for a particular day: blue indicates compliance of the model with the property, red indicates non-compliance and grey indicates the sensor of interest was not triggered on that day.

From Figure 6, we observe that some properties were satisfied more regularly than others. For example, for Property *Shower 2* (specified in Listing 5), its sensor of interest is only triggered on three out of sixteen days. While this may seem unexpected, it does align with the expected use of showers in older adults, which is about once a week [20]. An example of the behaviour of Subject A violating Property *Shower 2* is shown in Figure 5. In this case they violated the property because after turning the shower on, they moved to the kitchen and opened the garbage bin, thus leaving the shower unattended. Similarly, for cooking related activities (see Property *Burner* in Figure 6) the model also accurately represents the behaviour of the older adult whereby the property was violated most of the time. However, upon inspecting the subtraces, it becomes apparent that the violations lie in visiting the adjoining pantry and the dining room, or in switching on lights in the den and bathroom. This suggests the property may be too strong for this particular layout and individual.

Table 3 shows the verification times for the pruned model on different properties and subtraces. Note that the mean execution time of model checking a property does not depend on the length of the subtrace but does depend on how early the violation occurs in the trace. For example, a property checked on a subtrace

consisting of 10 events can be violated at the beginning of that subtrace. The same property on a different subtrace, also consisting of 10 events, can be violated at the end of that subtrace.

Property	#Subtraces	#events(min)	#events(max)	#sensors(max)	Mean time
Shower 1	1	4	4	2	3.21
Shower 2	189	2	73	14	3.10
TV Bedroom	17	2	274	29	2.88
Hall door	36	2	13	3	3.10
Burner	12	2	44	10	3.57
Freezer	20	2	559	33	4.01
Refrigerator 2	247	2	8	3	3.10
Refrigerator 1	16	2	220	34	3.25
Toaster	56	2	14	5	2.93
TV Living	57	2	145	20	3.03

Table 3: Pruned model timings, indicating the maximum number of sensors left after pruning (out of 70)

As can be seen, it takes on average under four seconds to verify a subtrace. For comparison, the time required to verify the *Burner* property with the full model is on average 6.12 seconds while with the pruned model drop to 3.57 seconds, meaning a reduction of nearly 50%. This is consistent among all the subtraces, as reflected in Figure 4.

5 Related Work

There has been some previous work on the identification of ADLs based on unobtrusive sensors using symbolic techniques. Chen et al. [5], for instance, introduce a knowledge-driven approach based on ontological modelling for ADL recognition in a smart home. Activity knowledge is modelled by properties describing the types of objects that can be used to perform an activity. Magherini et al. [14] use model checking and temporal logic for the recognition of ADLs in a smart home. Their work relies on past behaviour to build constraints on ADLs, such as specific steps to achieve the activity. Konios et al. [12] use Petri nets and model checking to identify normal and abnormal behaviour in data captured from different sensors. Garcia et al. [11, 10] collect data from ambient and wearable sensors to identify abnormal behaviour in the preparation of a beverage in the kitchen. Their approach relies on a sequence of steps that have to be performed sequentially. Compared to our approach, the strict adherence to the sequence of steps restricts the behaviour, e.g. a person should remain in the kitchen during tea brewing, which does not always align with the real world.

6 Conclusion

In this paper, we present a methodology for representing and reasoning about ADLs. Our approach relies on spatial contextual information and the sensor-observed behaviour of a person to build tailored models. These models are then checked for violations against encoded properties derived from health and safety guidelines, and the resulting counter-examples are visualised using a graphical widget. To show the applicability of our methodology, we conducted experiments on a real-world dataset about an older adult living independently. Results support the feasibility of our methodology for analysing ADLs, identifying potential deviations from expected behaviour. While some deviations seem genuinely undesirable, such as leaving the shower running and leaving the bathroom, others are more nuanced, such as leaving the kitchen while cooking. This suggests, aside from the need to tailor the model itself as we have done in this work, there is a further need to individualise the general properties from guidelines to the person and their context. Such context individualisation can help in dealing with rules that might be too strict, ensuring that the guidelines are both practical and adaptable to the unique needs of each individual.

Past operators in pLTL are currently used for properties related to physical constraints imposed by the layout. We anticipate that when expressing more complex properties, such as a person's preferences, additional usage of past operators will be required. As part of future work, we plan to extend our model to incorporate the time elapsed between events, allowing properties to account for duration. We also note that, while our current dataset is consistent with the assumptions set out in Section 2, others might not align with our current approach e.g. sensors could have several levels of activations or deactivations. We will explore how to adapt our methodology to accommodate particular dataset characteristics as parameters.

Finally, our current approach is tailored to cases involving a sole home occupant. Future research will extend our work to include multiple subjects, providing a more comprehensive understanding of ADLs within the context of shared living spaces. However, it should be emphasised that this restriction aligns with the living situations of a significant proportion of older adults across the world [19], where living on one's own is not uncommon, making the current work relevant in its own right.

Acknowledgements

This research was funded by the Legal & General Group (research grant to establish the independent Advanced Care Research Centre at University of Edinburgh). The funder had no role in conduct of the study, interpretation or the decision to submit for publication. The views expressed are those of the authors and not necessarily those of Legal & General.

References

[1] Marco Benedetti and Alessandro Cimatti. "Bounded Model Checking for Past LTL". In: *Proceedings of the Ninth International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Vol. 2619. Lecture Notes in Computer Science. Springer International Publishing, 2003, pp. 18–33. ISBN: 3-540-00898-5. DOI: 10.1007/3-540-36577-X_3.

[2] Armin Biere. "Bounded Model Checking". In: *Handbook of Satisfiability - Second Edition*. Vol. 336. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021, pp. 739–764. DOI: 10.3233/FAIA201002.

[3] Ailsa M. Cameron, Eleanor K. Johnson, and Simon Evans. "Older people's perspectives on living in integrated housing and care settings: the case of extra care housing". English. In: *Journal of Integrated Care* (2020). DOI: 10.1108/JICA-09-2019-0040.

[4] Chuanrui Chen, Shichao Ding, and Joseph Wang. "Digital health for aging populations". In: *Nature Medicine* 29.7 (July 2023), pp. 1623–1630. ISSN: 1546-170X. DOI: 10.1038/s41591-023-02391-8.

[5] Liming Chen, Chris D. Nugent, and Hui Wang. "A Knowledge-Driven Approach to Activity Recognition in Smart Homes". In: *IEEE Transactions on Knowledge and Data Engineering* 24.6 (2012), pp. 961–974. DOI: 10.1109/TKDE.2011.51.

[6] Alessandro Cimatti et al. "NuSMV 2: An OpenSource Tool for Symbolic Model Checking". In: *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 359–364. ISBN: 978-3-540-45657-5.

[7] Edmund M. Clarke et al. *Model Checking*. The Cyber-Physical Systems Series. MIT Press, 1999.

[8] Canada's Centre for Digital Media Literacy. *Managing Television in the Home*. https://mediasmarts.ca/tipsheet/managing-television-home. Accessed: 2024-04-06. 2016.

[9] FireAngel. *Fire Safety in the kitchen*. Accessed: 2024-04-06. 2021. URL: https://www.fireangel.co.uk/wp-content/uploads/2017/07/Fire-safety-in-the-kitchen-2024.pdf.

[10] Matias Garcia-Constantino, Alexandros Konios, and Chris Nugent. "Modelling Activities of Daily Living with Petri nets". In: *IEEE International Conference on Pervasive Computing and Communications Workshops*. 2018, pp. 866–871. DOI: 10.1109/PERCOMW.2018.8480225.

[11] Matias Garcia-Constantino et al. "Ambient and Wearable Sensor Fusion for Abnormal Behaviour Detection in Activities of Daily Living". In: *IEEE International Conference on Pervasive Computing and Communications Workshops*. 2020, pp. 1–6. DOI: 10.1109/PerComWorkshops48775.2020.9156249.

[12] Alexandros Konios et al. "Unifying and Analysing Activities of Daily Living in Extra Care Homes". In: *IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing(DASC)*. 2018, pp. 474–479.

[13] Marta Kwiatkowska, Gethin Norman, and David Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd International*

Conference on Computer Aided Verification. Vol. 6806. LNCS. Springer, 2011, pp. 585–591.

[14] Tommaso Magherini et al. "Using Temporal Logic and Model Checking in Automated Recognition of Human Activities for Ambient-Assisted Living". In: *IEEE Transactions on Human-Machine Systems* 43 (Nov. 2013), pp. 509–521. DOI: 10.1109/TSMC.2013.2283661.

[15] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specifications*. Springer New York, 1992.

[16] Michelle E. Mlinac and Michelle C. Feng. "Assessment of Activities of Daily Living, Self-Care, and Independence". In: *Archives of Clinical Neuropsychology* 31.6 (Aug. 2016), pp. 506–516. DOI: 10.1093/arclin/acw049.

[17] Emmanuel Munguia Tapia. "Activity recognition in the home setting using simple and ubiquitous sensors". MA thesis. Massachusetts Institute of Technology, 2003.

[18] Ali Akbar Razon and Ishtiaque Ahmad. "A Study on Fire Safety and Security at Kitchen in Apartment Buildings". In: *International Journal of Latest Engineering and Management Research* 2 (2017), pp. 62–71.

[19] David Reher and Miguel Requena. "Living Alone in Later Life: A Global Perspective". In: *Population and Development Review* 44.3 (2018), pp. 427–454.

[20] Robin Schiltz. *How To Help An Elderly Person To Shower Or Bathe: 7 Tips*. Accessed: 2024-04-06. 2023. URL: https://seniorsafetyadvice.com/how-to-assist-the-elderly-in-the-shower.

[21] Nikita Sharma et al. "Implementation of Unobtrusive Sensing Systems for Older Adult Care: Scoping Review". In: *JMIR Aging* 4.4 (2021). DOI: 10.2196/27862.

[22] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. "Activity Recognition in the Home Using Simple and Ubiquitous Sensors." In: vol. 3001. Lecture Notes in Computer Science. Springer, 2004, pp. 158–175.