"The fridge door is open"–Temporal Verification of a Robotic Assistant's Behaviours

Clare Dixon[1], Matt Webster[1], Joe Saunders[2],
Michael Fisher[1] and Kerstin Dautenhahn[2]

[1] Dept. of Computer Science, University of Liverpool, L69 3BX, UK
[2] Adaptive Systems Research Group, University of Hertfordshire, AL10 9AB, UK

Abstract. Robotic assistants are being designed to help, or work with, humans in a variety of situations from assistance within domestic situations, through medical care, to industrial settings. Whilst robots have been used in industry for some time they are often limited in terms of their range of movement or range of tasks. A new generation of robotic assistants have more freedom to move, and are able to autonomously make decisions and decide between alternatives. For people to adopt such robots they will have to be shown to be both safe and trustworthy. In this paper we focus on formal verification of a set of rules that have been developed to control the Care-O-bot, a robotic assistant located in a typical domestic environment. In particular, we apply *model-checking*, an automated and exhaustive algorithmic technique, to check whether formal temporal properties are satisfied on all the possible behaviours of the system. We prove a number of properties relating to robot behaviours, their priority and interruptibility, helping to support both safety and trustworthiness of robot behaviours.

1 Introduction

Robot assistants are being developed to help, or work, closely with humans in industrial, domestic and health care environments. In these environments the robots will need to be able to act autonomously and make decisions to choose between a range of activities and yet will need to operate close to, or in collaboration with humans. We need to make sure that such robotic assistants are both safe and trustworthy. Safety involves showing that the robot does nothing that (unnecessarily) endangers the person. To this end the International Organization for Standardization (ISO) TC184/SC2 Technical Committee has been working on ISO 13482, a standard relating to safety requirements for non-industrial robots, i.e. non-medical personal care robots[3]. Trustworthiness involves social issues beyond pure safety. It is not just a question of whether the robots are safe but whether they are perceived to be safe, useful and reliable, and will not do anything we would consider unpleasant, unfriendly, or dangerous.

In this paper we consider the application of formal verification to the Care-O-bot® [10], an autonomous robotic assistant deployed in a domestic-type house

[3] `www.sis.se/popup/iso/isotc184sc2/index.asp`

at the University of Hertfordshire. Low-level robot actions such as movement, speech, light display, etc., are controlled by groups of high-level rules that together define particular behaviours; for example, a sequence of rules to let the user know that the fridge door is open. We apply *formal verification*, in particular *model-checking* [2], to such behaviours within the Care-O-bot. In model-checking, a mathematical model of *all* the possible executions of the system is constructed (often a finite state transition system) and then *all* possible routes through this model are checked against a required logical formula representing a desired formal property of the system. Model-checking is different from running robot test experiments as it can check that a property holds on *all* paths through the input model. Robot experiments remain useful, allowing the observation of what occurs on *particular* runs of the real system. Together, model-checking the behaviours for the robot alongside real robot experiments within the robot house gives us a stronger assurance that the robot behaves as desired and helps convince users of its trustworthiness.

In preliminary work [18] we modelled the Care-O-bot behaviours using a human-robot teamwork modelling language, known as Brahms [14], and applied formal verification to the resulting Brahms models. Brahms is a multi-agent modelling, simulation and development environment designed to model both human and robotic activity using rational agents and has been used at NASA for modelling astronaut-robot planetary exploration teams. As the Care-O-bot behaviours have similarities with Brahms constructs, and as a tool [15] has been developed to translate from Brahms models into the SPIN model-checker [5], this route was adopted as a preliminary, quick and systematic way to model-check Care-O-bot behaviours. Issues with the approach involve the necessity to first produce a Brahms model, the resulting models being unnecessarily large due to the need to translate aspects of Brahms unnecessary to the Care-O-bot rules, the length of time for verification of simple properties, and issues relating to the modelling of non-determinism. Additionally, as this was a preliminary attempt at model-checking robot behaviours, features such as selecting between alternative behaviours using inbuilt priorities and whether the behaviours could be interrupted were not modelled.

In this paper we provide a translation (by hand) of the behaviours for the Care-O-bot directly into input suitable for a particular model-checker. Whilst a hand crafted translation is both time consuming and potentially error prone it means that we have much greater control over the size of the models (and related verification times) in terms of the level of abstraction taken. Additionally we can allow much more non-determinism in the models (for example allowing the variables relating to sensor information such as "the television being on", "the fridge door being open", "the doorbell ringing", etc., to be set non-deterministically). A longer term aim is to develop an automated, direct translation from sets of Care-O-bot behaviours to one or more model-checkers and this "by-hand" translation gives us useful insight into how to achieve this.

This paper is organised as follows. In Section 2 we describe the Care-O-bot robot assistant and its environment. In Section 3 we give more details about

temporal logic and model-checking and explain how the translation into input to the model-checker has been carried out. In Section 4 we give results from formally verifying several properties via the model-checker. In Section 5 we discuss related work, while conclusions and future work are provided in Section 6.

2 The Robot House and the Care-O-bot

The University of Hertfordshire's "robot house" is a typical suburban house near Hatfield. While offering a realistic domestic environment along with typical house furnishings, the robot house is also equipped with sensors which provide information on the state of the house and its occupants, such as whether the fridge door is open and whether someone is seated on the sofa [11, 3].

The robot house can be used to conduct Human-Robot Interaction experiments in a setting that is more natural and realistic than a university laboratory (e.g. [12, 16]). One of the robots in the house is the (commercially available) Care-O-bot robot manufactured by Fraunhofer IPA [10]. It has been specifically developed as a mobile robotic assistant to support people in domestic environments, and is based on the concept of a robot butler. The Care-O-bot robot, shown in Figure 3, has a manipulator arm incorporating a gripper with three fingers, an articulated torso, stereo sensors serving as "eyes", LED lights, a graphical user interface, and a moveable tray. The robot's sensors monitor its current location, the state of the arm, torso, eyes and tray. The robot can "speak" in that it can express text as audio output using a text-to-speech synthesising module.

The robot's software is based on the Robot Operating System (ROS)[4]. For example, to navigate to any designated location within the house, the robot uses the ROS navigation package in combination with its laser range-finders to perform self-localisation, map updating, path planning, and obstacle avoidance in real-time while navigating along the planned route. High-level rules are sent to the robot via the ROS script server mechanism and these are then interpreted into low-level actions by the robot's software. For example, high-level rules can take the form "lower tray", "move to sofa area of the living room", "say 'The fridge door is open' ", etc. The Care-O-bot's high-level decision making is determined by a set of behaviours which are stored in a database. Behaviours (a set of high level rules) take the form:

```
Precondition-Rules -> Action-Rules
```

where `Precondition-Rules` are a sequence of propositional statements that are either true or false, linked by Boolean *and* and *or* operators. `Action-Rules` are a sequence rules denoting the actions that the Care-O-bot will perform only if the `Precondition-Rules` hold. The `Precondition-Rules` are implemented as a set of SQL queries and the `Actions-Rules` are implemented through the ROS-based `cob_script_server` package, which provides a simple interface to operate Care-O-bot.

[4] `wiki.ros.org/care-o-bot`

For example the rules for the behaviour `S1-alertFridgeDoor` are provided in Fig. 1. Here, the rule numbers from the database are given, where rules 27 and 31

```
27  Fridge Freezer Is *ON* AND has been ON for more than 30 seconds
31  ::514:: GOAL-fridgeUserAlerted is false
    32  Turn light on ::0::Care-o-Bot 3.2 to yellow
    34  move ::0::Care-o-Bot 3.2 to ::2:: Living Room and wait for
            completion
    35  Turn light on ::0::Care-o-Bot 3.2 to white and wait for completion
    36  ::0::Care-o-Bot 3.2 says 'The fridge door is open!' and wait for
            completion
    37  SET ::506::GOAL-gotoCharger TO false
    38  SET ::507::GOAL-gotoTable TO false
    39  SET ::508::GOAL-gotoSofa TO false
    40  ::0::Care-o-Bot 3.2 GUI, S1-Set-GoToKitchen, S1-Set-WaitHere
    41  SET ::514::GOAL-fridgeUserAlerted TO true
```

Fig. 1. The `S1-alertFridgeDoor` rules

represent the precondition-rules, rules 32, 34-36, 40 provide the (descriptions of the) action-rules while 37-39 and 41 initiate the setting of various flags. Rules 27 and 31 check whether the fridge door is open and `GOAL-fridgeUserAlerted` is false. If these hold (and the preconditions for no other behaviour with a higher priority hold) then this behaviour will be executed by setting the robot's lights to yellow, moving to the living room, setting the robot's lights to white, saying "The fridge door is open", setting various goals to be false, providing the user several options via the Care-O-bot's interface and finally setting `GOAL-fridgeUserAlerted` to be true.

The Care-O-bot's database is composed of multiple rules for determining a variety of autonomous behaviours, including checking the front doorbell, telling the person when the fridge door is open, and reminding them to take their medication, etc. The robot house rule database used for this paper (which includes a set of 31 default behaviours) can be obtained from the EU ACCOMPANY projects Git repository[5].

The robot can perform only one behaviour at a time. Each of the behaviours is given a priority between 0 and 90. If the preconditions for more than one behaviour hold at any moment then the behaviour with the highest priority is executed. For example the behaviour `S1-alertFridgeDoor` has a priority of 60 and `S1-gotoKitchen` has a priority of 40 so if the preconditions to both were true then the former would be executed. Priorities remain the same throughout the execution. If more than one behaviour has equal priority, then they are loaded in a random order and on execution whatever behaviour is first (in the set of equal priorities) will be executed first. Additionally each behaviour is flagged as

[5] `github.com/uh-adapsys/accompany`

interruptible (1) or not (0). In general, behaviours execute to completion, i.e. all the rules that are part of the behaviour are performed, even if the precondition to another behaviour becomes true during its execution. However, behaviours flagged as interruptible are terminated if the precondition of a higher priority behaviour becomes true whilst it is executing. The priorities and interruptible status (denoted *Int*) of behaviours are given in Table 1. Behaviours with both priority status and interruptible status set to zero are omitted from this table to save space (but are included in the model).

Name	Priority	Int	Name	Priority	Int
S1-Med-5PM-Reset	90	0	S1-gotoTable	40	1
checkBell	80	0	S1-kitchenAwaitCmd	40	1
unCheckBell	80	0	S1-sofaAwaitCmd	40	1
S1-remindFridgeDoor	80	0	S1-tableAwaitCmd	40	1
answerDoorBell	70	0	S1-WaitHere	40	1
S1-alertFridgeDoor	60	0	S1-ReturnHome	40	1
S1-Med-5PM	50	1	S1-continueWatchTV	35	1
S1-Med-5PM-Remind	50	1	S1-watchTV	30	1
S1-gotoKitchen	40	1	S1-sleep	10	1
S1-gotoSofa	40	1			

Table 1. Priority Table for Behaviours

3 Modelling the Care-O-bot Behaviours

Model-checking [2] is a popular technique for formally verifying the temporal properties of systems. Input to the model-checker is a model of all the paths through a system and a logical formula (often termed a *property*) to be checked on that model. A useful feature of model-checkers is that, if there are execution paths of the system that *do not* satisfy the required temporal formula, then at least one such "failing" path will be returned as a counter-example. If no such counter-examples are produced then all paths through the system indeed satisfy the prescribed temporal formula. Here we use the `NuSMV` [1] model-checker.

The logic we consider is propositional linear-time temporal logic (PTL), where the underlying model of time is isomorphic to the Natural Numbers, $\mathbb{N}$. A model for PTL formulae can be characterised as a sequence of *states* of the form: $\sigma = s_0, s_1, s_2, s_3, \ldots$ where each state, s_i, is a set of proposition symbols, representing those propositions which are satisfied in the i^{th} moment in time.

In this paper we will only make use three of temporal operators: '$\bigcirc$' (*in the next moment in time*), '$\Diamond$' (*sometime in the future*), and '$\Box$' (*always in the future*) in our temporal formulae. The notation $(\sigma, i) \models A$ denotes the truth of formula A in the model σ at state index $i \in \mathbb{N}$, and is recursively defined as

follows (where PROP is a set of propositional symbols).

$$
\begin{array}{ll}
(\sigma, i) \models p & \text{iff } p \in s_i \text{ where } p \in \text{PROP} \\
(\sigma, i) \models \bigcirc A & \text{iff } (\sigma, i+1) \models A \\
(\sigma, i) \models \diamondsuit A & \text{iff } \exists k \in \mathbb{N}.\ (k \geqslant i) \text{ and } (\sigma, k) \models A \\
(\sigma, i) \models \Box A & \text{iff } \forall k \in \mathbb{N}.\ (k \geqslant i) \text{ and } (\sigma, k) \models A
\end{array}
$$

Note that $\Box$ and $\diamondsuit$ are duals, i.e. $\Box\varphi \equiv \neg\diamondsuit\neg\varphi$. The semantics of Boolean operators is as usual.

First we identify the variables to use in the `NuSMV` representation of the model.

Booleans from the Care-O-bot rules: many of the Boolean values from the system can be used directly, for example goals `GOAL-fridgeUserAlerted`, or `GOAL-gotoSofa`

Robot actions: involving its location, the robot torso position, speech, light colour, the orientation of the tray or providing alternatives on the Care-O-bot display for the person to select between are modelled as enumerated types for example `location` could have the values `livingroom, tv, sofa, table, kitchen, charging`.

Scheduling behaviours: we use a variable `schedule` with an enumerated type for each behaviour, e.g. if `schedule = schedule_alert_fridge_door` holds this denotes that the preconditions to the `S1-alertFridgeDoor` behaviour have been satisfied and this behaviour has been selected to run having the highest priority.

Executing Behaviours: we use a variable called `execute` with an enumerated type for each behaviour involving more than one step eg `execute = execute_alert_fridge_door` denotes that the `S1-alertFridgeDoor` behaviour is executing. An enumerated type `execute_step` with values `step0, step1` etc keeps track of which part of the behaviour has been completed.

Fig. 2 gives the schema showing the changes to variables in subsequent states in the state transition diagram for the behaviour `S1-alertFridgeDoor` when its precondition holds and this behaviour is scheduled. This corresponds with the behaviour in Fig. 1. The first box shows the preconditions that must hold (i.e. `fridge_freezer_on` and ¬`goal_fridge_userAlerted`) and that the behaviour must be scheduled (`schedule = schedule_alert_fridge_door`) before the other variables are set. This behaviour cannot be interrupted (see Fig. 1) so once it is scheduled it will execute to completion. However other behaviours that are interruptible (eg `S1-gotoKitchen`) may not complete all their steps if the preconditions of another behaviour with a higher priority become true during its execution.

When the previous behaviour has completed a new behaviour is scheduled. To set the next value of `schedule` in the `NuSMV` input file, a list of cases are enumerated as follows

$$
\begin{array}{c}
condition_1 : schedule_1 \\
\ldots : \ldots \\
condition_n : schedule_n
\end{array}
$$

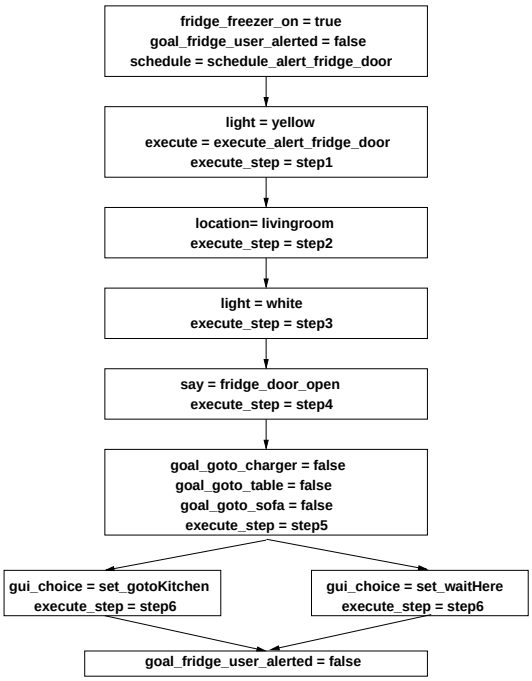

Fig. 2. Schema showing changes to variables for `S1-alertFridgeDoor` behaviour

where $condition_i$ represents the preconditions to activate the behaviour and $schedule_i$ is the behaviour selected to execute. The behaviours with higher priorities appear above behaviours with lower priorities and `NuSMV` selects the first case it encounters where the condition is satisfied.

We need to abstract away from some of the timing details included in the database to obtain a model that is discrete, for example, involving delays or timing constraints of 60 seconds or less. The behaviour `S1-watchTV` involves checking a goal has been been false for 60 minutes. To achieve this we use an enumerated type `goal_watch_tv_time` with values for every 15 minutes `m0, m15, m30, m45, m60`. We could increase the number of values to represent 5 minute intervals (or even less), for example, but this would increase the size of the model.

4 Verification Using Model-Checking

Here we provide the properties that we checked and the outcome from running these on `NuSMV`.

1. If the fridge door is open and $goal\_fridge\_user\_alerted$ is false then at sometime in the future the Care-O-bot will be in the living room and at some

time after that it will say the fridge door is open.

$$\Box((fridge\_freezer\_on \wedge \neg goal\_fridge\_user\_alerted) \Rightarrow \\ \Diamond(location = livingroom \wedge \Diamond say = fridge\_door\_open))$$

We expect this to be false as, even though the preconditions to the behaviour `S1-alertFridgeDoor` are satisfied, preconditions to a behaviour with a higher priority, namely `S1-answerDoorBell`, might hold and the other behaviour be executed instead of this.

2. If the fridge door is open and *goal_fridge_user_alerted* is false and the `S1-alertFridgeDoor` behaviour has been scheduled then at sometime in the future the Care-O-bot will be in the living room and at some time after that it will say the fridge door is open.

$$\Box((fridge\_freezer\_on \wedge \neg goal\_fridge\_user\_alerted \wedge \\ schedule = schedule\_alert\_fridge\_door) \Rightarrow \\ \Diamond(location = livingroom \wedge \Diamond say = fridge\_door\_open))$$

We expect this to be true as the `S1-alertFridgeDoor` behaviour is not interruptible so once it is scheduled it should execute to conclusion.

3. If the person selects *goto kitchen* via the Care-O-bot GUI then at sometime in the future its location will be in the kitchen.

$$\Box(gui\_choice = gui\_set\_gotoKitchen \Rightarrow \\ \Diamond location = kitchen)$$

We expect this to be false. Selecting $gui\_choice = gui\_set\_gotoKitchen$ sets the goal *goal_goto_kitchen* to be TRUE which is the precondition to the behaviour `S1-goToKitchen`. However the behaviour `S1-goToKitchen` may not be scheduled as the preconditions to higher priority behaviours may also be satisfied at the same time. Alternatively it may be scheduled but interrupted before completion.

4. If the person selects *goto kitchen* via the Care-O-bot GUI and this behaviour is scheduled then at sometime in the future its location will be in the kitchen.

$$\Box((gui\_choice = gui\_set\_gotoKitchen \wedge \\ \Diamond schedule = schedule\_goto\_kitchen) \Rightarrow \\ \Diamond(schedule = schedule\_goto\_kitchen \wedge \Diamond location = kitchen))$$

We expect this to be false. By selecting $gui\_choice = gui\_set\_gotoKitchen$ (as previously) the goal *goal_goto_kitchen* is set to be TRUE which is the precondition to the behaviour `S1-goToKitchen`. Also, here the behaviour `S1-goToKitchen` has been scheduled but it may be interrupted before completion.

5. If the sofa is occupied, the TV is on and the goal to watch TV has been false for at least 60 minutes, then at some time in the future the Care-O-bot will be located at the sofa and some time after that it will say *shall we watch TV*.

$$\Box((sofa\_occupied \wedge tv\_on \wedge \neg goal\_watch\_tv \wedge goal\_watch\_tv\_time = m60) \\ \Rightarrow \Diamond(location = sofa \wedge say = shall\_we\_watch\_tv))$$

We expect this to be false. Similar to (1) and (3) although the preconditions for behaviour `S1-watchTV` have been satisfied it may not be scheduled as the preconditions to higher priority behaviours may also be satisfied at the same time. Alternatively it may be scheduled but interrupted before completion so that the robot may not have moved to the sofa or may not have said *Shall we watch TV* (or both).

6. If the system attempts to execute a tray raise rule, at some point in the future the physical tray is raised (`physical_tray= raised`) and later the internal flag showing that tray is raised (`tray = raised`) holds.

$$\Box(execute\_raise\_tray \Rightarrow \Diamond(physical\_tray = raised \wedge \Diamond tray = raised))$$

   We expect this to be true as `raiseTray` cannot be interrupted.

7. The next property shows the interruption of the behaviour `S1-gotoKitchen` by a higher priority behaviour `S1-alertFridgeDoor`. If the `S1-gotoKitchen` behaviour is executing (and the Care-O-bot is not raising or lowering the tray which is not interruptible) and the preconditions to `S1-alertFridgeDoor` become true then in the next moment the behaviour `S1-gotoKitchen` will not be executing.

$$\Box((execute = execute\_goto\_kitchen \wedge \neg move\_tray \wedge fridge\_freezer\_on \wedge \neg goal\_fridge\_user\_alerted) \Rightarrow \bigcirc(\neg(execute = execute\_goto\_kitchen)))$$

   We expect this to be true due to the priority and interruption settings.

The results from verifying the above properties are given in Fig. 4. The outputs are produced by running `NuSMV` version 2.5.4 on a PC with a 3.0 GHz Intel Core 2 Duo E8400 processor, 4GB main memory, and 16GB virtual memory running Scientific Linux release 6.4 (Carbon) with a 32-bit Linux kernel. The timings below are carried out running `NuSMV` with the flags `-coi` (cone of influence) `-dcx` (disable generation of counterexamples) `-dynamic` (enable dynamic variable reordering). The size of the model generated by `NuSMV` has 130,593 reachable states.

The properties (3) and (5) above correspond with properties (1) and (3) from [18]. Note that we cannot prove *any* of the properties that were proved from [18]. One difference between the two models relates to the environment. In [18] a person is explicitly modelled and is the only cause of non-determinism in the model. The person can choose to do one of the following: "move to the kitchen", "move to the living room", "watch television", "send the robot to the kitchen", "send the robot to living room" or "do nothing". The doorbell was assumed not to ring. This means only one behaviour can be triggered at any one time, significantly reducing non-determinism in the previous work. In the current paper a person is not explicitly modelled, although this could be done, and sensors such as the television being on, the doorbell ringing, etc., are allowed to be arbitrarily set to be true or false at any point in the model so several behaviours might be triggered at once. Additionally, in the current work, priorities and

Fig. 3. Care-O-bot in the Robot House

Property	Output	Time (sec)
1	FALSE	11.1
2	TRUE	12.3
3	FALSE	7.7
4	FALSE	9.3
5	FALSE	11.6
6	TRUE	6.4
7	TRUE	6.9

Fig. 4. Model-checking Results

interruptions were modelled which were not considered in the previous work. The properties from [18] could not be proved here because the preconditions for higher priority behaviours might also hold and so be executed instead or they might be interrupted by higher priority behaviours during execution.

The size of the model here (130,593 reachable states) is much smaller than that in [18] (with 652,573 or more states). Additionally the verification times for the four properties in [18] took between 20-30 seconds, longer than the properties analysed in this paper. We believe this is because the direct hand-crafted translation we use here avoids the need for the translation of constructs required in the Brahms modelling language whilst still retaining the meaning of the Care-O-bot behaviours. Additionally, as mentioned previously here we have abstracted away from some of the timing details such as "wait for 5 seconds".

5 Related Work

This work applies model-checking to the behaviours of the Care-O-bot robot located in the University of Hertfordshire's robot house. Further details about robot house, robot architecture and control systems and experiments with participants in the robot house can be found, for example, in [3, 12, 16, 11, 13]. This paper builds on and extends the work described in [18]. As mentioned in Section 4 we use a different environment model allowing more non-determinism, model behaviour about priorities and interruptions and provide a direct translation into input to the `NuSMV` model-checker.

In [15] verification of a domestic home care scenario involving a person, a human carer, a robotic assistant and a house agent is considered. The scenario is modelled in Brahms and an automated translation from Brahms models into the SPIN model-checker is provided. However, here the scenario we analyse relates to real code used in practice in real life experiments in the University of Hertfordshire's robot house.

Other research using model-checking to verify aspects of robot behaviour include the verification of safety properties for a core of the Samsung Home Robot (SHR) [6], the analysis of robot motion [4], and the application of model-checking to the control system responsible for slowing down and stopping of a wheeled off-road vehicle [9]. Other formal methods used for robot verification include the application of quantified differential dynamic logic to prove physical location properties for algorithms controlling a surgical robot [7], the use of hybrid automata and statecharts to model and verify a multi-robot rescue scenario [8] and the application of an interactive theorem prover to specify and verify robot collision avoidance algorithms [17].

6 Conclusions and Future Work

We have modelled the behaviours of a robotic assistant in the model-checker `NuSMV` and proved a number of properties relating to this. The size of models and verification times were less than in previous work [18]. The priorities and interruptibility of the behaviours were modelled so that even if the satisfaction of the preconditions of behaviours such as `S1-alertFridgeDoor` or `S1-gotoKitchen` became true these behaviours might not be fully executed because of higher priority behaviours being scheduled instead. Many of the timing details were removed from the models but more detailed timings could be included at the expense of the size of models and verification times. The model of a person in the robot house was not represented but this could be incorporated showing their location for example. In future work we would like to provide an automated translation from a database of rules into a number of different model-checkers and we could experiment with larger databases of behaviours and different model-checkers. Note that we believe that verification of this or more complex systems should be targeted at an appropriate level of abstraction, in particular (as here), the decision making level or high level control rather than low level robot robot movement.

Returning to the issues of safety and trustworthiness, these verification results provide a route towards proving safety requirements and of using proofs to convince users of trustworthiness. These results should be used as a compliment to experiments with real people in the robot house relating to perceptions of trust to give more confidence in robotic assistants.

Acknowledgments The authors were partially supported by EPSRC grants EP/K006193 and EP/K006509.

References

1. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In: Proc. of Int. Conf. on Computer-Aided Verification (CAV) (2002)

2. Clarke, E., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (2000)
3. Duque, I., Dautenhahn, K., Koay, K.L., Willcock, I., Christianson, B.: Knowledge-driven User Activity Recognition for a Smart House. Development and Validation of a Generic and Low-Cost, Resource-Efficient System. In: Proc. of the Sixth Int. Conf. on Advances in Computer-Human Interactions (ACHI). pp. 141–146 (2013)
4. Fainekos, G., Kress-Gazit, H., Pappas, G.: Temporal Logic Motion Planning for Mobile Robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 2020– 2025. IEEE Computer Society Press (2005)
5. Holzmann, G.J.: The Spin Model Checker: Primer and Reference Manual. Addison-Wesley (2003)
6. Kim, M., Kang, K.C.: Formal construction and verification of home service robots: A case study. In: Automated Technology for Verification and Analysis (ATVA). LNCS, vol. 3707, pp. 429–443. Springer (2005)
7. Kouskoulas, Y., Renshaw, D., Platzer, A., Kazanzides, P.: Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In: Proc. of the 16th Int. Conf. on Hybrid Systems: Computation and Control. pp. 263–272. ACM (2013)
8. Mohammed, A., Furbach, U., Stolzenburg, F.: Multi-Robot Systems: Modeling, Specification, and Model Checking, chap. 11, pp. 241–265. InTechOpen (2010)
9. Proetzsch, M., Berns, K., Schuele, T., Schneider, K.: Formal Verification of Safety Behaviours of the Outdoor Robot RAVON. In: Fourth Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO. pp. 157–164. INSTICC Press (2007)
10. Reiser, U., Connette, C.P., Fischer, J., Kubacki, J., Bubeck, A., Weisshardt, F., Jacobs, T., Parlitz, C., Hägele, M., Verl, A.: Care-o-bot®3 - creating a product vision for service robot applications by integrating design and technology. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS). pp. 1992–1998 (2009)
11. Saunders, J., Burke, N., Koay, K.L., Dautenhahn, K.: A user friendly robot architecture for re-ablement and co-learning in a sensorised home. In: Assistive Technology: From Research to Practice (Proc. of AAATE). vol. 33, pp. 49–58 (2013)
12. Saunders, J., Salem, M., Dautenhahn, K.: Temporal issues in teaching robot behaviours in a knowledge-based sensorised home. In: Proc. 2nd International Workshop on Adaptive Robotic Ecologies (2013)
13. Saunders, J., Syrdal, D.S., Dautenhahn, K.: A template based user teaching system for an assistive robot. In: Proceedings of 3rd International Symposium on New Frontiers in Human Robot Interaction at AISB 2014 (2014)
14. Sierhuis, M., Clancey, W.J.: Modeling and simulating work practice: A method for work systems design. IEEE Intelligent Systems 17(5), 32–41 (2002)
15. Stocker, R., Dennis, L.A., Dixon, C., Fisher, M.: Verifying Brahms Human-Robot Teamwork Models. In: Proc. European Conference on Logics in Artificial Intelligence (JELIA). LNCS, vol. 7519, pp. 385–397. Springer (2012)
16. Syrdal, D.S., Dautenhahn, K., Koay, K.L., Walters, M.L., Ho, W.C.: Sharing spaces, sharing lives–the impact of robot mobility on user perception of a home companion robot. In: Proc. of 5th Int. Conf. on Social Robotics, (ICSR). pp. 321–330 (2013)
17. Walter, D., Täubig, H., Lüth, C.: Experiences in applying formal verification in robotics. In: 29^{th} Int. Conf. on Computer Safety, Reliability and Security (SafeComp). LNCS, vol. 6351, pp. 347– 360. Springer (2010)
18. Webster, M., Dixon, C., Fisher, M., Salem, M., Saunders, J., Koay, K.L., Dautenhahn, K.: Formal verification of an autonomous personal robotic assistant. In: Proc. of Workshop on Formal Verification and Modeling in Human-Machine Systems (FVHMS). pp. 74–79. AAAI (2014)