Formal Verification of an Autonomous Personal Robotic Assistant

Matt Webster, Clare Dixon and Michael Fisher
Centre for Autonomous Systems Technology
University of Liverpool, Liverpool, L69 3BX, UK
matt@liverpool.ac.uk

Maha Salem, Joe Saunders, Kheng Lee Koay and Kerstin Dautenhahn
Adaptive Systems Research Group
University of Hertfordshire, Hatfield, Hertfordshire, AL10 9AB, UK
m.salem@herts.ac.uk

Abstract

Human–robot teams are likely to be used in a variety of situations wherever humans require the assistance of robotic systems. Obvious examples include healthcare and manufacturing, in which people need the assistance of machines to perform key tasks. It is essential for robots working in close proximity to people to be both safe and trustworthy. In this paper we examine formal verification of a high-level planner/scheduler for autonomous personal robotic assistants such as Care-O-bot®. We describe how a model of Care-O-bot and its environment was developed using Brahms, a multi-agent workflow language. Formal verification was then carried out by translating this to the input language of an existing model checker. Finally we present some formal verification results and describe how these could be complemented by simulation-based testing and real-world end-user validation in order to increase the practical and perceived safety and trustworthiness of robotic assistants.

1 Introduction

Robotic assistants — robots which help people in particular tasks — are likely to be used for a variety of applications including personal healthcare, exploration within remote environments, and manufacturing. These robots will operate in close proximity to their human operators and therefore must be safe and trustworthy in their operations. One of the aims of the EPSRC-funded Trustworthy Robotic Assistants (TRA) project[1] is to develop tools and techniques for the verification and validation of robotic assistants. The TRA project uses three different methodologies for this: formal verification, simulation-based testing and end-user validation. In this paper we consider the application of formal verification to the Care-O-bot®: an autonomous robotic assistant deployed at the University of Hertfordshire's Robot House (see Figure 1).

[1] http://www.robosafe.org/

Figure 1: The Care-O-bot® Robotic Assistant operating in the University of Hertfordshire's Robot House.

Formal verification is the application of formal (i.e., mathematical) methods to the verification of systems. The approach used in this paper is based on model checking (Clarke, Grumberg, and Peled 1999), in which a model of a program or process is constructed. This model is typically non-deterministic, so that each "run" (or simulation) of the model can be different from the last. A program called a model checker exhaustively analyses all possible executions of the model in order to establish that some property holds. These properties can be derived from system requirements. Therefore it is possible, for example, to use a model checker to formally verify that in every execution of a given program, the program will always do something desirable. In other words, we can formally verify that a given requirement holds.

The model checker used in this paper is called SPIN (Holzmann 2003). SPIN has been publicly available since 1991 and has been used for the formal verification of a wide variety of systems, including flood control barriers, telecommunications switches and several space missions (Holzmann 2013). SPIN, which stands for Simple PROMELA Interpreter, verifies programs and processes written in PROMELA, the Process Meta-Language. Rather than writing in PROMELA directly, we utilise an intelli-

gent agent modelling language and simulation environment called Brahms (Sierhuis and Clancey 2002) to develop models of the Robot House and Care-O-bot. Brahms can be used to develop detailed models of systems with multiple interacting agents and has been used to model human–robot teams (Clancey et al. 2002) and complex workflows (Sierhuis and Clancey 2002) for space exploration. We translate the Brahms models automatically into PROMELA using the *BrahmsToPromela* translator software developed by Stocker et al. (2012), which is based on a formal semantics for Brahms (Stocker et al. 2011). As we shall see in Section 2, the autonomous control systems used in the Robot House were written at a similar level of abstraction to constructs in the Brahms modelling language. Therefore it was sensible to encode these autonomous systems using Brahms to minimise modelling errors due to changes in abstraction level, as well as to minimise the cost of developing the models.

In the remainder of this section we examine the Care-O-bot and the Robot House in more detail. In Section 2 we describe the way in which a model of the Care-O-bot's high-level planner/scheduler for autonomous decision-making system was developed using Brahms. Then, in Section 3, we show how that model was formally verified using the SPIN model checker. In Section 4 we compare our approach to related work, and in Section 5 we offer conclusions and directions for future research.

1.1 The Robot House and Care-O-bot®

The University of Hertfordshire's Robot House is a typical UK-suburban three-bedroom house near Hatfield in Hertfordshire, UK (see Figure 1). While offering a realistic domestic environment along with typical house furnishings and décor, the Robot House is further equipped with more than 50 sensors which provide real-time episodic information on the state of the house and the individuals occupying it. These sensors include electrical (e.g. refrigerator door open/closed sensor), furniture (e.g. cupboard drawers open), services (e.g. detect when toilet flush is being used) and pressure (e.g. chair sensors to detect when someone is seated) devices (Saunders et al. 2013; Duque et al. 2013).

The Robot House hosts a number of different robots that are frequently used to conduct Human-Robot Interaction (HRI) experiments in a setting that is more natural and realistic than a university laboratory (e.g. (Syrdal et al. 2013)). One of these robots is the commercially-available Care-O-bot® robot manufactured by Fraunhofer IPA (Reiser et al. 2009). 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." (Reiser et al. 2013) The Care-O-bot robot is equipped with a 7 degrees-of-freedom manipulator arm extended with a gripper with three fingers and is further comprised of an articulated torso, stereo sensors serving as "eyes", LED lights and a tray. Accordingly, the robot's sensors include its current location, the state of the arm, torso, eyes and tray. By means of a text-to-speech synthesising module, the robot is also capable of expressing given text as audio output.

The robot's software is based on the Robot Operating System (ROS) and a number of ROS packages (e.g., drivers, navigation and simulation software) are available online[2]. For example, to navigate to any designated location within the house, the robot uses the ROS navigation package[3] 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 commands are sent to the robot via the ROS *script server* mechanism which are then interpreted into low-level commands by the robot's software. For example, these high-level commands can take the form "raise tray", "move to location x", "grab object on tray", "say hello", etc.

The Care-O-bot's high-level decision making is determined by a set of rules which are stored in a MySQL database. Rules take the form:

```
Guard
==
RobotAction*
```

where `Guard` is a sequence of propositional statements that are either true or false, linked by Boolean AND (`&`) and OR (`|`) operations. `RobotAction*` is a sequence of actions which the Care-O-bot will perform only if the `Guard` is true. In practice, the `Guard` is implemented as a set of SQL queries and the `RobotActions` are implemented through the ROS-based `cob_script_server` package, which provides a simple interface to operate Care-O-bot. For example, take the following rule which lowers the Care-O-bot's tray:

```
SELECT * FROM Sensors WHERE sensorId=500
   AND value = 1     &
SELECT * FROM Sensors WHERE sensorId=504
   AND value = 1
==
light,0,yellow
tray,0,down,,wait
light,0,white,,wait
cond,0,500,0
cond,0,501,1
```

Here, the guard checks whether sensors 500 (which monitors whether the Care-O-bot's tray is raised) and 504 (which monitors whether the tray is empty) have the value of 1 or not by performing the SQL SELECT queries. (The Sensors table stores the values of all sensors in the Robot House.) If both sensors have the value 1 (i.e., "true"), then the five prescribed actions will be performed and will have the following effects: (i) turn the Care-O-bot's light to yellow; (ii) lower the tray and wait; (iii) turn the light to white; (iv) set sensor 500 to false; and (v) set sensor 501 to true.

The Care-O-bot's rule database is composed of multiple rules for determining a variety of autonomous behaviours, including checking and answering the front doorbell, reminding a person to take their medication, and so on. The full Robot House rule database used for this paper (which

[2] http://wiki.ros.org/care-o-bot

[3] http://wiki.ros.org/navigation

includes a set of 31 default rules) can be obtained from the EU ACCOMPANY project's Git repository[4].

2 Modelling the Care-O-bot using Brahms

The high-level autonomous decision making within the Robot House and Care-O-bot® at the University of Hertfordshire is carried out by a high-level planning/scheduling system described in the previous section. The code base includes a database of 31 default rules for the Robot House and Care-O-bot to follow. Careful examination of these rules revealed that the they are similar in structure to the various constructs within the Brahms multi-agent workflow programming language.

The first step in modelling was to convert the full set of Care-O-bot rules into a more convenient if-then rule representation. For example, the rule in the previous translation was rewritten as:

```
IF tray_is_raised AND tray_is_empty
THEN set_light(yellow)
     move_tray_and_wait(lowered_position)
     set_light(white)
     wait()
     set(tray_is_raised,false)
     set(tray_is_lowered,true)
```

Once translated into this format, these rules could then be straightforwardly translated into Brahms. A key concept in Brahms is the workframe, which specifies a set of things to be done when a given condition holds. The Robot House rules were translated into Brahms workframes within the Care-O-bot agent, with the `IF a THEN b` rules translated into the `when a do b` construct in Brahms. For example, the rule above was translated into a Brahms workframe called `wf_lowerTray`:

```
workframe wf_lowerTray {
 repeat: true;
 priority: 10;

 when(knownval(current.trayIsRaised = true)
   and
   knownval(current.trayIsEmpty = true))
 do{
   conclude((current.lightColour =
      current.colourYellow));
   lowerTrayAndWait();
   conclude((current.lightColour =
      current.colourWhite));
   waitForLightColourChange();
   conclude((current.trayIsRaised =
      false));
   conclude((current.trayIsLowered =
      true));
   }
}
```

This workframe is set to repeat, which means that it can be used more than once by the agent in which it exists. Multiple workframes can be eligible for execution by the Brahms interpreter at the same time, so the `priority` sets the importance of the workframe relative to other workframes (with larger numbers being more important). The `when a do b` construct is intuitive, and says that when a set of conditions are true, in this case the `trayIsRaised` and `trayIsEmpty` variables are true, then the agent should do the actions that follow. In the action list, the `conclude()` construct is used to determine when beliefs should be updated within the Brahms agent. The `lowerTrayAndWait()` and `waitForLightColourChange()` are programmer-defined primitive actions, whose function is to denote that something has happened and has taken a certain amount of time, e.g.:

```
primitive_activity lowerTrayAndWait(){
   max_duration: 4;
}
```

In general, Robot House rules were translated into Brahms workframes on a one-to-one basis. However, in some cases it was necessary to use more than one Brahms workframe for a rule. This generally happened when a rule contained interaction with the user via the GUI. For instance, when the person sits down and watches television (detected via sensors in the sofa seats and the television power outlet) the Care-O-bot approaches the person and asks whether they would like to watch the television together. At this point the person has three options which are presented to the person by the Robot House via a GUI on a tablet computer: to tell the Care-O-bot to watch television, return to its charging station, or continue with its current task. This behaviour is modelled using a Brahms workframe within the Care-O-bot agent, in which these options are communicated to the person using the `announceQueryToUser_ThreeOptions()` activity:

```
workframe wf_watchTV {   // cob rule 28
 repeat: true;
 priority: 10;
 when(knownval(robotHouse.sofaSeatOccupied =
    true)
   and
  knownval(robotHouse.televisionWattage > 10)
   and  ...)
 do{
  conclude((current.queryUserOption1 =
     current.activityWatchTV));
  conclude((current.queryUserOption2 =
     current.activityReturnHome));
  conclude((current.queryUserOption3 =
     current.activityContinue));
  conclude((current.userQueried = true));
  conclude((current.queryUser_ThreeOptions =
      true));
  announceQueryToUser_ThreeOptions();
  conclude((current.askedToWatchTV = true));
 }
}
```

The "person" agent then selects a response and sends it back to the Care-O-bot model. This then causes one of three workframes to trigger: which one depends on the user response. For example, the following workframe is triggered if the person agent tells the Care-O-bot to watch television:

```
workframe wf_optionSelectedWatchTV {
```

[4] https://github.com/uh-adapsys/accompany

```
 repeat: true;
 priority: 10;
 when(knownval(Person.userRespondedToQuery =
    true)
   and
  knownval(Person.queryResponse =
     current.activityWatchTV))
 do{
  conclude((Person.userRespondedToQuery =
     false));
  conclude((Person.guiSetWatchTV = true));
  conclude((current.userQueried = false));
 }
}
```

2.1 Modelling a Scenario

After translating the Robot House rules into Brahms it was necessary to set up a model of the Robot House environment, or *scenario*. The scenario determines the range of possibilities within the Robot House environment. This environment consists primarily of the Care-O-bot®, the person being assisted by Care-O-bot, and the Robot House itself. For example, the scenario consists of a model of the person and the Robot House, where each is defined as an agent within Brahms. Another agent, the "Campanile Clock", measures the passage of time in the model and keeps the other agents updated of the current time.

Our scenario lasts from 12pm to 6pm on a typical day in the Robot House. At any given point in the day the person may choose to sit down and watch TV, move into the living room area, or move into the kitchen (e.g., to prepare food to eat), or may choose to send the Care-O-bot into the kitchen or the living room. At 5pm the person will need to take their medication.

The person in the scenario can act non-deterministically, that is, they can behave in a manner which is unpredictable within the model of the scenario. This non-determinism is implemented in Brahms as a set of five workframes within the person agent, all of which will fire at a given point. Each workframe has a priority. The highest priority workframe is executed, and a belief is modified within the agent (using the "conclude" keyword in Brahms). This belief is modified with a level of certainty (known as the belief-certainty) which states that the belief will be updated with a given probability. If the belief is updated, this information is communicated to the Care-O-bot agent or the Robot House agent (depending on the workframe) which causes these agents to know that the person has done something, e.g., sent the Robot to the kitchen via the GUI, or that the person has moved into the living room. If the belief is not updated, then the next workframe fires, and so on. It is possible for none of the five workframes to update a belief within the person agent, and this special case models the ability of the person to do nothing.

Based on this simple scenario we can establish a number of high-level requirements of the Care-O-bot. For example, the Care-O-bot should remind the person to take their medication at 5pm. Another requirement is that the Care-O-bot should go into the living room if it is told to go into the living room by the person. As we shall see in the next section, we can formalise these kinds of requirements using temporal logic and verify them using the SPIN model checker.

3 Formal Verification of Brahms Models Using *BrahmsToPromela* and SPIN

Brahms refers to a multi-agent workflow specification language, as well as a software package consisting of an agent simulation toolkit and an integrated development environment. The Brahms software does not come with formal verification tools built-in, so for formal verification we used the *BrahmsToPromela* translator developed by Stocker et al. (2012) at the University of Liverpool. *BrahmsToPromela* allows models written using a subset of Brahms to be automatically translated into Promela, the process meta-language used by the SPIN model checker. Once translated, SPIN can be used for the automatic formal verification of the Brahms model with respect to particular requirement. In our case, we formalise these requirements using linear temporal logic (LTL), which allows the formalisation of concepts relating to time, e.g., "now and at all points in the future" ($\Box$), "now or at some point in the future" ($\Diamond$) and "in the next state" ($\bigcirc$) (Fisher 2011). This enables formalisation of safety requirements (something bad never happens, $\Box\neg$bad), liveness properties (e.g, something good always happens, $\Box$good), fairness properties (e.g., if one thing occurs infinitely often so does another, e.g., $\Box\Diamond$send $\implies$ $\Box\Diamond$receive) and reachability properties (e.g., eventually something happens, $\Diamond$something).

To explore possibilities, the following sample requirements were translated and their formalised properties verified using SPIN for the Brahms model described in Section 2. Within these, the *belief* operator, "B", is parameterised by the agent that holds the belief — so $\mathsf{B}_{\text{Care-O-bot}}x$ means that the Care-O-bot believes x is true.

1. It is always the case that if the Care-O-bot believes that the person has told it to move into the kitchen, then it will eventually move into the kitchen.

$$\Box\left[\begin{array}{c}\mathsf{B}_{\text{Care-O-bot}}(\mathsf{B}_{\text{Person}}\text{guiGoToKitchen})\\ \implies \Diamond\mathsf{B}_{\text{Care-O-bot}}(\mathsf{location} = \text{Kitchen})\end{array}\right]$$

2. It is always the case that if the Care-O-bot believes that the person has told it to move to the sofa in the living room, then it will eventually move into there.

$$\Box\left[\begin{array}{c}\mathsf{B}_{\text{Care-O-bot}}(\mathsf{B}_{\text{Person}}\text{guiGoToSofa})\\ \implies \Diamond\mathsf{B}_{\text{Care-O-bot}}(\mathsf{location} = \text{LivingRoomSofa})\end{array}\right]$$

3. It is always the case that if the Robot House believes that the sofa seat is occupied, and if the Robot House believes that the television wattage is higher than 10 watts, then eventually the Care-O-bot will move to the living room sofa and ask the person if they want to watch the television with the Care-O-bot.

$$\Box\left[\begin{array}{c}\mathsf{B}_{\text{RobotHouse}}\text{sofaSeatOccupied} \wedge\\ \mathsf{B}_{\text{RobotHouse}}\text{televisionWattage} > 10\\ \implies \Diamond\left(\begin{array}{c}\mathsf{B}_{\text{Care-O-bot}}\text{location} = \text{LivingRoomSofa} \wedge\\ \mathsf{B}_{\text{Care-O-bot}}\text{askedToWatchTV}\end{array}\right)\end{array}\right]$$

4. It is always the case that if the time is 5pm, then the Care-O-bot will believe that the medicine is due.

$$\Box \left[\begin{array}{c} \mathsf{B}_{\text{Campanile\_Clock}}\text{time} = 5\text{pm} \implies \\ \Diamond \mathsf{B}_{\text{Care-O-bot}}\text{medicineDue} \end{array} \right]$$

The first two requirements are derived from a higher-level requirement that, in general, the Care-O-bot should follow instructions given to it by the person. The third property is important for maintaining the social activity of the person within the Robot House, who is temporarily under the care of the Care-O-bot. The fourth property is derived from a higher-level requirement that the Care-O-bot should remind the person to take their medication at the correct time.

The computational resources of the formal verification of the four properties above are as follows:

Prop.	States	Depth	Memory (MB)	Time (s)
1	652,573	46,617	10,132	20.7
2	652,573	46,617	10,132	20.7
3	746,497	53,009	11,596	30.7
4	652,573	46,617	10,132	20.3

The formal verification was carried out on an eight-core Intel® Core™ i7-3720QM CPU (2.60GHz) laptop with 16 GB of memory running Ubuntu Linux 12.04 LTS.

In each case the same Promela model was used. Therefore any difference in the number of states or time taken is due to the complexity of the property being verified and the resulting automaton used by the model checker. It can be seen that for requirements 1, 2 and 4 the resources used were almost identical, and this is to be expected as these properties were similar in structure and produced similar automata. Property 3 produced a slightly more complex automaton and therefore required more resources to verify.

4 Comparisons With Related Work

Stocker et al. (2012) describe the development of the *BrahmsToPromela* software tool, which is used in this work to automatically translate a model of the Care-O-bot® written in Brahms into a PROMELA specification which can then be formally verified using the SPIN model checker. The authors examine an assisted living scenario similar to the Robot House system tackled in this paper. However, the work in this paper expands on the work of Stocker et al. by modelling a real-life robotic system and scenario where the rules are directly derived from actual code used in practice. This further demonstrates the validity of formal verification of autonomous robotic assistants using Brahms, *BrahmsToPromela* and SPIN.

Saunders et al. (2012) and Duque et al. (2013) described the University of Hertfordshire Care-O-bot and Robot House systems which are used as a basis for this work. More information on the development of these systems can be found on the ACCOMPANY project website[5].

Formal verification has been used before for robotic systems. For example, Mohammed, Furbach, and Stolzenburg (2010) used hybrid automata and hybrid statecharts for formal modelling and verification of multi-robot systems; Cowley and Taylor (2011) used dependent type theory and linear logic for the formal verification of assembly robots; and Kouskoulas et al. (2013) formally verified control algorithms for surgical robots. However, very little work has been conducted to formally verify the safety and trustworthiness of robotic home companions. This is where our work aims to complement existing research in the area of formal verification of autonomous and robotic systems.

[5] http://accompanyproject.eu

5 Conclusions and Future Work

We have shown how it is possible to formally verify an autonomous decision making planner/scheduler system for the Robot House assisted living environment and the Care-O-bot® robotic assistant. This was done by converting the Robot House planner/scheduler rules into Brahms: a workflow language for defining the behaviour of multiple agents. Crucially, these rules matched the Brahms style very closely. Brahms was also used to model the Robot House environment, including the Care-O-bot, the Robot House and a person living within the Robot House. The Brahms model was then translated into the Promela language using the *BrahmsToPromela* tool developed by Stocker et al. (Stocker et al. 2012). Once in Promela, the Brahms workflows (and, by implication, the Care-O-bot's decision making system) was formally verified. As proof-of-concept, a small selection of key properties were formally verified, demonstrating that this process can be used for verification of autonomous decision making systems for robotic systems.

5.1 Future Work

A common challenge when using model checkers is scalability. Model checking can be resource-intensive, especially when models are complex. For instance, in this paper around 10^4 MB of memory was needed per property verified. It is likely that increasing the complexity of the model (to add more Care-O-bot® rules, for example) would result in the model checker running out of memory, therefore preventing formal verification using the model checker. However, it is likely that we can reduce the complexity of these models through improvements to the *BrahmsToPromela* translator, which is at an early stage of development and has not been optimised to a high degree.

As described in Section 1, the formal verification of the Robot House Care-O-bot is taking place within a wider project aimed at developing a practical approach to verifying and validating robotic assistants with regard to safety and trustworthiness. Model checking — the kind of formal verification used here — is exhaustive and automatic, and can therefore provide a level of assurance that a computer system is safe and trustworthy. However, formal verification is often based on models of the system being verified, as well as models of its environment. In order to trust the results of formal verification one needs to validate the system and environment models to make sure that they are accurate. For this reason the TRA project will use simulation-based testing (Wile, Goss, and Roesner 2005) as well as real-world testing by end users in order to complement the formal verification approach. Furthermore, simulation-based testing and real-world testing will complement each other: the

former providing large numbers of automatically-generated tests that would be impossible in the real world, and the latter providing the essential grounding in reality that is ultimately essential to develop trust in autonomous systems.

One way to improve the validity of the formal verification results presented in this paper is to increase the fidelity of the Robot House model by making it more realistic. This could be done in a number of ways. For example, the person agent model at present is basic: the person can do any one of five actions at any given point. However, a real person within the Robot House can do much more than this. A realistic model of the behaviour of the person within the Robot House would be a significant step forward with respect to verifying the safety of the Robot House and the Care-O-bot. The fidelity of the model could also be increased by modelling environmental changes such as the doorbell ringing or another person entering the house. Another way in which the formal verification results could be improved is in the development of interesting properties to verify. The properties in this paper were derived from the engineering requirements of the Care-O-bot and the Robot House. However, it is likely that many requirements will come from the end-users of robotic systems. Further research is needed to determine ways in which the requirements of users (rather than engineers) can be obtained and formalised.

Simulation-based testing could be used to analyse systems which are difficult to formally verify, e.g., control systems which control the Care-O-bot's arm. A simulation-based model of the arm's control system and environment could be developed to enable verification of arm movements within safe parameters. For instance, it could be determined that the arm would not collide with objects during the handover of a object from the Care-O-bot to the person.

Real-world tests could involve human–robot interaction experiments in which human participants interact with the Care-O-bot in the Robot House by exploring its functionalities in the dedicated home environment. For example, participants could add new robot behaviours to the database, such as "remind me to take my medicine at 10pm", or call pre-existing robot commands such as "move to the living room." The system would then need to verify that a given behaviour or command does not violate the set of existing rules. As a long-term goal of this project, formally verified and validated robotic assistants will not only result in safer machines, but also in more trustworthy robot companions.

Acknowledgments The authors were partially supported by EPSRC grants EP/K006193, EP/K006320, EP/K006509, and EP/K006223.

References

Clancey, W. J.; Sierhuis, M.; Kaskiris, C.; and van Hoof, R. 2002. Brahms mobile agents: Architecture and field tests. In *Human–Robot Interaction: Papers from the AAAI Fall Symposium*. ISBN 978-1-57735-174-0.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. USA: MIT Press.

Cowley, A., and Taylor, C. J. 2011. Towards Language-based Verification of Robot Behaviors. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4776–4782.

Duque, I.; Dautenhahn, K.; Koay, K. L.; Willcock, I.; and Christianson, B. 2013. Knowledge-driven User Activity Recognition for a Smart House — Development and Validation of a Generic and Low-Cost, Resource-Efficient System. In *Proc. Sixth International Conference on Advances in Computer-Human Interactions*.

Fisher, M. 2011. *An Introduction to Practical Formal Methods Using Temporal Logic*. Wiley.

Holzmann, G. J. 2003. *The SPIN Model Checker: Primer and Reference Manual*. USA: Addison-Wesley.

Holzmann, G. 2013. Inspiring applications of Spin. http://spinroot.com/spin/success.html. Accessed 2013-10-10.

Kouskoulas, Y.; Renshaw, D. W.; Platzer, A.; and Kazanzides, P. 2013. Certifying the Safe Design of a Virtual Fixture Control Algorithm for a Surgical Robot. In Belta, C., and Ivancic, F., eds., *Proc. Hybrid Systems: Computation and Control (HSCC)*, 263–272. ACM.

Mohammed, A.; Furbach, U.; and Stolzenburg, F. 2010. Multi-robot systems: Modeling, specification, and model checking. *Robot Soccer* 241–265.

Reiser, U.; Connette, C.; Fischer, J.; Kubacki, J.; Bubeck, A.; Weisshardt, F.; Jacobs, T.; Parlitz, C.; Hägele, M.; and Verl, A. 2009. Care-O-bot® 3: Creating a Product Vision for Service Robot Applications by Integrating Design and Technology. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1992–1998.

Reiser, U.; Jacobs, T.; Arbeiter, G.; Parlitz, C.; and Dautenhahn, K. 2013. Care-O-bot® 3 — Vision of a Robot Butler. In Trappl, R., ed., *Your Virtual Butler — The Making-of*, volume 7407 of *LNAI*. Springer. 97–116.

Saunders, J.; Burke, N.; Koay, K. L.; and Dautenhahn, K. 2013. A User Friendly Robot Architecture for Re-ablement and Co-learning in A Sensorised Home. In *Proc. 12th European AAATE Conference*.

Sierhuis, M., and Clancey, W. J. 2002. Modeling and Simulating Work Practice: A Method for Work Systems Design. *IEEE Intelligent Systems* 17(5):32–41.

Stocker, R.; Sierhuis, M.; Dennis, L. A.; Dixon, C.; and Fisher, M. 2011. A Formal Semantics for Brahms. In *Proc. 12th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, volume 6814 of *LNCS*, 259–274. Springer.

Stocker, R.; Dennis, L. A.; Dixon, C.; and Fisher, M. 2012. Verification of Brahms Human–Robot Teamwork Models. In *Proc. 13th European Conf. on Logics in Artificial Intelligence*, volume 7519 of *LNCS*, 385–397. Springer.

Syrdal, D.; Dautenhahn, K.; Koay, K. L.; Walters, M.; and Ho, W. 2013. Sharing Spaces, Sharing Lives – The Impact of Robot Mobility on User Perception of a Home Companion Robot. In *Proc. Fifth International Conference on Social Robotics (ICSR)*, volume 8239 of *LNAI*. Springer.

Wile, B.; Goss, J. C.; and Roesner, W. 2005. *Comprehensive Functional Verification*. Morgan Kaufmann.