Petri Net Based Supervisory Control of a Social Robot with LTL Specifications

Bruno Lacerda, Pedro U. Lima, Javi Gorostiza and Miguel A. Salichs

***Abstract*— We describe the implementation of a method to control a social robot based on discrete event system supervisory control theory. The sensors and actuators of the robot are modelled as Petri nets, and the target behaviour is given as a set of rules written as linear temporal logic (LTL) formulas. The Petri net models and LTL rules are then used to build a Petri net realization of a supervisor that is guaranteed by construction to restrict the robot's behaviour such that the rules are fulfilled. This approach provides a close-to-natural-language description of the target behaviour, and can be the basis for a human-robot speech interaction device, where the rules for the robot to fulfil are described by the human in natural language.**

I. INTRODUCTION

Social robotics is an emerging field where a special attention is paid to the interaction between the robot and the human user(s). To be appropriate for this kind of interaction, a robot must fulfil several requirements, such as providing different forms of interacting (e.g., speech, touch), possessing a high level of autonomy, and being able to learn and to adapt to different situations.

In this work, we present the application of a method that implements linear temporal logic (LTL) [7] specifications in Maggie [14], a social robot developed at the RoboticsLab of Universidad Carlos III de Madrid with the main purpose of interacting with children. We will model a subset of Maggie's sensors and actuators as Petri nets (PN) [13], and define rules that restrict Maggie's behaviour in LTL. The PN and the LTL formulas are then used to build a PN realization of a supervisor [4] that forces the system to fulfil the formulas. The use of LTL provides a close-to-natural-language framework to define the rules for the robot, which can allow the development of a speech interaction module, where the user states the rules using a pre-defined subset of natural language, and the robot then shows the behaviour originated by those rules. This kind of interaction can be used as a game for children, where they can see the impact of the rules they state in the behaviour of the robot.

The work presented here can be seen as a first step to develop an alternative approach to the work presented in [9], where a method to teach action sequences by means of speech interaction is defined. This method relies on speech interaction with the user, who utters orders in a pre-defined subset of natural language. The orders are translated to a sequential function chart (SFC) [5], a graphical programming language based on binary PNs, used for programmable logic controllers. SFCs feature elements such as conditional choices, parallel execution of sequences and loops. In this approach, the user starts from an "empty plan", and builds a behaviour for the robot by fully stating all the actions, and in which order and in which conditions they are to be performed. Conversely, in our work, the user starts with all the possible actions for the robot being able to be executed randomly and restricts them to the desired behaviour by stating rules that the robot must fulfil. Hence, [9] can be seen as a direct approach to building sequences for the robot to execute where the whole behaviour must be specified step-by-step, while we propose a more abstract approach, where the desired behaviour is obtained from a set of general rules. This might lead to the appearance of "surprises" in the behaviour of the robot, in the sense that it may exhibit properties that were not thought of by the user when stating the rules.

Another technique used for humans teaching behaviours to robot systems, which has presented good results, is programming by demonstration, where the user shows the robot how to perform a given task, being imitated by the robot afterwards. Two examples of this approach are [1] and [3]. Due to its suitability to model concurrent systems and the wide range on analysis methods available, PNs are a widely used tool for the modelling of robot systems [15], [6]. Also, LTL has been successfully applied as a language to specify and synthesize correct by design admissible behaviours [12], [10]. Furthermore, a translation between a restricted subset of English and LTL to deal with motion planning problems for mobile robots is defined in [11].

The paper is outlined as follows: in Section II we provide a brief description of Maggie, the platform where we implemented the method. In Section III we show how to model Maggie's sensors and actuators as Petri nets where some places correspond to the truth value of binary variables used to describe the state of the system, followed by an explanation on how to write the LTL specifications, in Section IV. Section V describes the method for the construction of the supervisor and, in Section VI, a discussion about the approach and future developments is provided.

B. Lacerda and P. Lima are with the Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, Portugal {blacerda, pal}@isr.ist.utl.pt. The work of B. Lacerda was supported by the portuguese Fundação para a Ciência e Tecnologia through grant SFRH/BD/45046/2008. Most of this work was accomplished while they were visiting Universidad Carlos III de Madrid, in P. Lima's case under a Cátedra de Excelencia awarded by Banco Santander.

J. Gorostiza and M. Salichs are with the Robotics Lab, Universidad Carlos III de Madrid, Madrid, Spain {jgorosti, salichs}@ing.uc3m.es

Fig. 1. Maggie, the social robot, interacting with a child.

II. Maggie, The Social Robot

Maggie, depicted in Figure 1, is a social robot developed at the RoboticsLab of Universidad Carlos III de Madrid to interact mainly with children. In this Section, we provide a brief description of its capabilities, further details can be found in [14].

Maggie is a 1.35m tall girl-like doll. Her base is equipped with two differentially driven wheels and a caster wheel on each side. The arms and the eyelids have 1 DOF: up/down, while the neck has 2 DOF: up/down and left/right. She also possesses tactile sensors, including on the shoulders and on top of her head. These are the sensors and actuators that we took into account in our implementation. Maggie possesses many other capabilities, such as infrared and ultrasound sensors used for navigation, a color camera for people tracking and a mouth shape with invisible web-cam and coloured lights synchronized with the speech. These capabilities result in a platform well suited to study human-robot interaction and robot learning by training and teaching.

The control architecture is based on the automatic-deliberative (AD) control architecture proposed in [2]. In this architecture, the robot skills are divided in two levels. In the automatic level, we find the skills related with robot sensors and actuators. In the deliberative level, we find higher-level skills, such as a planner or our implementation of the feedback-loop of supervisory control. The skill we developed subscribes to events representing changes in sensor readings, for which an event handler is implemented. Also, it can order the execution of skills related to performing simple actions (e.g., raising an arm or start spinning).

III. Petri Net Modelling and Execution

We will use PNs to model Maggie behaving freely in the environment. This model is a building block of the supervisor, being used in conjunction with the LTL formula specifying the target behaviour to build it. In the PN models used here both places and transitions have a specific interpretation:

- *Places* represent the value of binary variables that are used to define the state of the system. For each variable, there is one place representing that is is *true* and one place representing that is is *false*. For all reachable markings of the Petri net, there is one token in one of these places and zero in the other. This means that, for a given marking, one can unambiguously extract the corresponding state. A state is the set of variables for which the places meaning that they are true have one token;
- *Transitions* represent orders to execute actions or changes in sensor readings, i.e., the firing of a transition represents a communication between our implementation and the automatic level of the architecture.

In Figure 2, the PN model for Maggie is depicted. The model is composed of several modules, one for each actuator and one of each sensor. We represent the interpretation of each place and transition as $\langle . \rangle$. For example, a token in place p_1 means that $moving\_forward$ is true and a token in place p_2 means that $moving\_forward$ is false. Furthermore, the firing of transition t_1 represents executing the $move\_forward$ action and the firing of transition t_{19} means that someone started touching the tactile sensor on the left shoulder.

To illustrate, the initial state for Maggie, given by the depicted marking, is the set:

$$\{base\_idle,\ head\_center,\ head\_down,\ left\_arm\_down,\ right\_arm\_down,\ left\_eyelid\_down,\ right\_eyelid\_down\}$$

We note that, for the base, when a *start* action is issued, the robot starts performing that action until a *stop* action is issued, that is, the action continues until it is explicitly stopped. For the other models, we assume that the actuator moves for a fixed amount and then stops, hence no *stop* action is required.

We implemented a *Petri net executor* in C++, so that the robot is able to run its system Petri net. We start by dividing the transitions into transitions corresponding to actions - t_1 to t_{18} - and transitions corresponding to changes in sensor readings - t_{19} to t_{24}. The implementation is a loop that, in each step, randomly selects one of the active transitions corresponding to actions and fires it, ordering the execution of the corresponding action and updating the marking. Also, when one of the sensors changes state, the handler interrupts the loop, and the transition corresponding to that sensor change is immediately fired and the marking is updated. By running this PN without supervising it, Maggie simply executes random actions, displaying an unrestricted behaviour.

IV. Writing the LTL Specification

We will write LTL formulas that restrict Maggie's behaviour. These formulas are written over the set Π, defined by the union of the set E of events (actions plus sensor readings) and the set D of variables that describe the states. Formulas

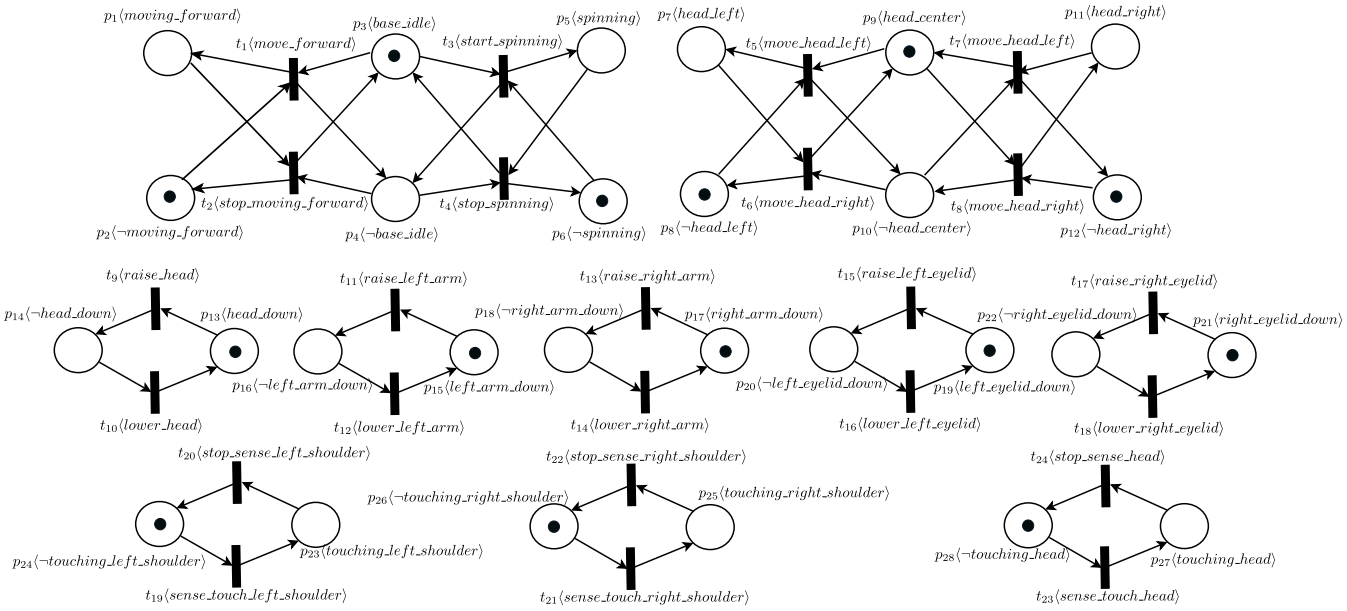

Fig. 2. The Petri net model for Maggie's different actuators and sensors.

LTL are written using the usual propositional connectives plus a set of temporal connectives, including the next (X), always (G) and until (U) connectives. LTL formulas are evaluated over infinite sequences of sets of propositional symbols $\sigma : \mathbb{N} \to 2^{\Pi}$. Intuitively, for $i \in \mathbb{N}$:

- A state $\sigma(i)$ satisfies $X\varphi$ when state $\sigma(i+1)$ satisfies φ;
- A state $\sigma(i)$ satisfies $G\varphi$ if all states $\sigma(j)$ with $j \geq i$ satisfy φ;
- A state $\sigma(i)$ satisfies $\varphi U \psi$ if φ is satisfied in all states, until a state appears where ψ is satisfied.

With these operators, we can specify a wide array of different rules for Maggie to fulfil. We will illustrate this by providing the LTL specifications for three different target behaviours. The first behaviour is a simple sequence triggered by touching the head of the robot. The sequence can be stated in natural language as "*When the head is touched raise the left arm, then raise and lower the right arm. When the head stops being touched, lower the left arm*". This sequence can be translated into the following simple rules:

- The left arm must only be raised when the head is being touched:

$$G(touching\_head \Leftrightarrow (X(\neg left\_arm\_down))) \quad (1)$$

- The right arm must only be raised after the left arm is raised:

$$G(raise\_left\_arm \Leftrightarrow (Xraise\_right\_arm)) \quad (2)$$

- After raising the right arm, it must be immediately lowered again:

$$G(raise\_right\_arm \Rightarrow (Xlower\_right\_arm)) \quad (3)$$

The second behaviour shows how to specify different reactions to different sensor readings. We can state it has "*If the left shoulder is touched, wait until the right shoulder is touched and then raise your right arm. If the right shoulder is touched, wait until the left shoulder is touched and raise your left arm. After raising an arm, lower it again and return to the initial state*". To keep track of which shoulder is touched, we use the state of the eyelids. If both eyelids are down, then no shoulder has been touched yet. If one of the eyelids is up, then the corresponding shoulder was touched and Maggie is waiting for the other shoulder to be touched. Hence, the behaviour can be implemented by the following rules:

- If the left shoulder is touched and both eyelids are down (i.e., no shoulder was touched yet), raise the left eyelid:

$$G((sense\_left\_shoulder \wedge right\_eyelid\_down \wedge left\_eyelid\_down) \Leftrightarrow (Xraise\_left\_eyelid)) \quad (4)$$

- If the right shoulder is touched and the left eyelid is up (i.e., the left shoulder was previously touched), raise the right arm:

$$G((sense\_right\_shoulder \wedge \neg left\_eyelid\_down) \Leftrightarrow (Xraise\_right\_arm)) \quad (5)$$

- After raising the right arm, return to the initial state by lowering the left eyelid and the right arm:

$$G(raise\_right\_arm \Leftrightarrow (Xlower\_left\_eyelid)) \quad (6)$$

$$G(lower\_left\_eyelid \Leftrightarrow (Xlower\_right\_arm)) \quad (7)$$

- If the right shoulder is touched and both eyelids are down (i.e., no shoulder was touched yet), raise the right

eyelid:

$$G((sense\_right\_shoulder \wedge right\_eyelid\_down \wedge left\_eyelid\_down) \Leftrightarrow (X raise\_right\_eyelid)) \quad (8)$$

- If the left shoulder is touched and the right eyelid is up (i.e., the right shoulder was previously touched), raise the left arm:

$$G((sense\_left\_shoulder \wedge \neg right\_eyelid\_down) \Leftrightarrow (X raise\_left\_arm)) \quad (9)$$

- After raising the left arm, return to the initial state by lowering the right eyelid and the left arm:

$$G(raise\_left\_arm \Leftrightarrow (X lower\_right\_eyelid)) \quad (10)$$

$$G(lower\_right\_eyelid \Leftrightarrow (X lower\_left\_arm)) \quad (11)$$

Note that in this behaviour, touching the same shoulder more than one time in a row does influence the arm to be raised. After touching one of the shoulders, Maggie ignores all other sensor readings until the other shoulder is touched. When that happens, she raises the corresponding arm.

The third behaviour is also triggered by touching the head, but we allow the robot to perform random actions during the behaviour. It can be stated as "*Move arms randomly. When the head is touched, start spinning until both arms are up*". This can be translated into the following rules:

- Only start spinning when the head is touched, you are not spinning yet and both arms are not already up:

$$G((sense\_head \wedge (\neg spinning) \wedge \neg(\neg left\_arm\_down \wedge \neg right\_arm\_down)) \Leftrightarrow (X start\_spinning)) \quad (12)$$

- After starting to spin, continue spinning until both arms are up:

$$G(start\_spinning \Rightarrow (X(spinning U(\neg left\_arm\_down \wedge \neg right\_arm\_down)))) \quad (13)$$

- When both arms are up, stop spinning (or continue stopped if you are not spinning):

$$G(\neg left\_arm\_down \wedge \neg right\_arm\_down) \Rightarrow (X \neg spinning)) \quad (14)$$

All of these examples also include an additional formula which avoids the execution of the actions that are not referred to in the specification, i.e., in each example, a formula of the form $G\left(\bigwedge_{e \in E_o} \neg e\right)$, where E_o is the set of actions that is not mentioned in that example is also added.

V. Constructing the Supervisor

To build the supervisor, we need to define a way to compose the LTL formulas with the PN model. This is done by translating the formula φ to a (non-deterministic) Büchi automaton (BA) B_φ that accepts exactly the infinite sequences that satisfy φ. There are several methods for the construction of such automaton. In the implementation of the method we present here, we use one of the most efficient translation algorithms, LTL2BA, described in [8]. In Figure 3, we show the BA obtained for formula (14).

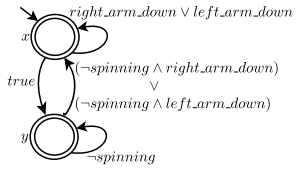

Fig. 3. The Büchi automaton for LTL formula (14)

The alphabet set of this automaton is $2^{E \cup D}$, but propositional logic formulas in the disjunctive normal form (DNF) are used to describe the transition labels in a more compact way. A formula in the DNF is the disjunction ($\vee$) of a set of conjunctive clauses. A conjunctive clause is a conjunction ($\wedge$) of literals. A literal is a propositional symbol or its negation. For example, the transition label from sate y to state x means that any element of $2^{E \cup D}$ that does not contain *spinning* and contains *right_arm_down* or that does not contain *spinning* and contains $left\_arm\_down$ is a label for the transition. We also note that the automata outputted by LTL2BA are *trimmed*, i.e., all their states can be reached and there is a path between each state and at least one accepting state.

The PN that restricts Maggie's behaviour to the one specified by an LTL formula is a PN that simulates a run in parallel of the PN model of Maggie and the observer[1] of the BA obtained for that formula, where a transition t can only fire in parallel with a transition of the observer of the BA when we are ensured that the marking to which the PN evolves satisfies the formula labelling the BA transition. This is done by looking at each transition of the PN and checking in which conditions it can fire while satisfying the transition labels of the BA. To illustrate, we will show how to compose the transitions from state x in Figure 3 with transition t_3 of Figure 2. We start by stating the facts that are guaranteed to happen after t_3 fires:

- Action *start_spinning* has just occurred;
- All other actions and sensor readings in E did not just occur;
- State description variable *spinning* is $true$ - this fact is guaranteed because place p_5, the place corresponding to *spinning* being $true$, is an output place of t_3, hence after t_3 fires it will always have a token - and *base_idle* is $false$ - by the same reasoning as with *spinning*.

Hence, we can define a *partial* valuation that represents all the information about events and state description variables that is guaranteed to happen after the firing of t_3:

$$v_{t_3}(\pi) = \begin{cases} 1 & \text{if } \pi \in \{start\_spinning, spinning\} \\ 0 & \text{if } \pi \in (E \setminus \{start\_spinning\}) \cup \{base\_idle\} \\ \downarrow & \text{if } \pi \in D \setminus \{spinning, base\_idle\} \end{cases}$$

[1]The observer of a non-deterministic automaton G is its deterministic version, built using the known power-set construction [4].

The valuation is undefined for all state description variables that are not directly related to the firing of t_3, in the sense that none of the places representing its value receives a token with the firing of t_3. This valuation can be applied to the transition labels of the BA, which we will denote as $[\![.]\!]_{v_{t_3}}$. The result of such evaluation is:

- *true* or *false* if v_{t_3} provides enough information to evaluate the truth value of the label;
- The formula composed of the elements of the label for which v_{t_3} should be defined for one to unambiguously be able to evaluate its truth value.

In the case of the transitions of the BA in Figure 3:

$$[\![right\_arm\_down \vee left\_arm\_down]\!]_{v_{t_3}} = right\_arm\_down \vee left\_arm\_down \tag{15}$$

$$[\![true]\!]_{v_{t_3}} = true \tag{16}$$

$$\left[\!\!\left[\begin{array}{c} (\neg spinning \wedge right\_arm\_down) \\ \vee \\ (\neg spinning \wedge left\_arm\_down) \end{array} \right]\!\!\right]_{v_{t_3}} = \left[\!\!\left[\begin{array}{c} (false \wedge right\_arm\_down) \\ \vee \\ (false \wedge left\_arm\_down) \end{array} \right]\!\!\right]_{v_{t_3}} = false \tag{17}$$

$$[\![\neg spinning]\!]_{v_{t_3}} = false \tag{18}$$

Due to the non-determinism of the BA, we will have to analyse 3 different cases for state x:

- *Can we keep the BA is state x after firing t_3?* To guarantee that we stay in state x, we need to check in which conditions can t_3 fire while satisfying formula (15) and not satisfying formula (16), i.e., satisfying the label of the transition that goes to x and not satisfying the label of the transition that goes to y:

  $$(right\_arm\_down \vee left\_arm\_down) \wedge \neg true = false$$

  This means that this situation can never happen. Thus, we do not add transitions to the supervisor;
- *Can we take the BA from state x to state y after firing t_3?* In this case we need to check in which conditions can t_3 fire while not satisfying formula (15) and satisfying (16), i.e.:

  $$\neg(right\_arm\_down \vee left\_arm\_down) \wedge true = \neg right\_arm\_down \wedge \neg left\_arm\_down$$

  Hence, we add one transition to the supervisor (because there is only one conjunctive clause), with arcs equal to t_3 plus place $\{x\}$ as an input place and place $\{y\}$ as an output place (thus evolving the "observer" of the BA from the state $\{x\}$ to state $\{y\}$) and a reflexive arc[2] to the the places representing the literals in the obtained conjunctive clause, i.e., one reflexive arc to the place representing $\neg right\_arm\_down$ (p_{18}) and one reflexive arc to the place representing $\neg left\_arm\_down$ (p_{16});
- *Can we take the BA from state x to both states x and y after firing t_3?* In this case we need to check in which conditions can t_3 fire while satisfying both formulas (15) and (16) , i.e., satisfying:

  $$(right\_arm\_down \vee left\_arm\_down) \wedge true = right\_arm\_down \vee left\_arm\_down$$

  Hence, we add two transitions to the supervisor (one for each conjunctive clause, i.e., one corresponding to *right_arm_down* and one corresponding to *left_arm_down*) with arcs equal to t_3 plus place $\{x\}$ as an input place and place $\{x,y\}$ as an output place and, for each of the added transitions, a reflexive-arc to the place representing the corresponding literal.

Figure 4 depicts the fragment of the supervisor obtained from the analysis above, showing the three transitions created. The algorithm starts by analysing the transitions from the initial state of the Büchi automaton and adds newly reached states of the observer to a queue to be analysed next. Note that the fact that both formulas (17) and (18) are false means that when we analyse t_3 in the observer BA state $\{y\}$, no transition will be created. This is aligned with what one would expect, since we are building a PN that satisfies the natural language rule "*When both arms are up, stop spinning (or continue stopped if you are not spinning)*", hence the action *start_spinning* must be disabled whenever both arms are up, which, simplifying, is the meaning of BA state y.

The PNs obtained using this composition are then used in the feedback loop of modular supervisory control [4]. Informally, they run in parallel with the PN model of Maggie, executing the same events, and outputting the set of enabled events, given by the intersection of the labels of the active transitions for each PN supervisor. At each step, all events that are not in the set of enabled events cannot be executed by Maggie. This feedback loop was implemented on top of the Petri net executor.

In the video available at `http://bit.ly/dQqVQK`, we show both the uncontrolled behaviour of Maggie and its behaviour when being supervised by the Petri nets obtained from the specifications described here.

VI. Conclusion and Further Work

We described the implementation on Maggie, the social robot, of a method to perform PN-based supervision of robotic tasks, where the admissible behaviours are given in LTL. The similarities between natural language and temporal logic, and the fact that we can write formulas over both the events (actions plus sensor readings) of the system and a set of variables used to describe the state of the robot, allows the writing of rules for a wide array of different goal behaviours. This approach can be the basis for a method based on speech interaction, that allows the user to state rules for the robot to fulfil in natural language. Specifically for Maggie, this can be made into a game where children can see the impact of their rules in the robot's behaviour.

[2]A reflexive arc between a transition t and place p is a pair of arcs, one from p to t and another from t to p. Hence, t only fires when there is a token in p but it does not change the amount of tokens in p.

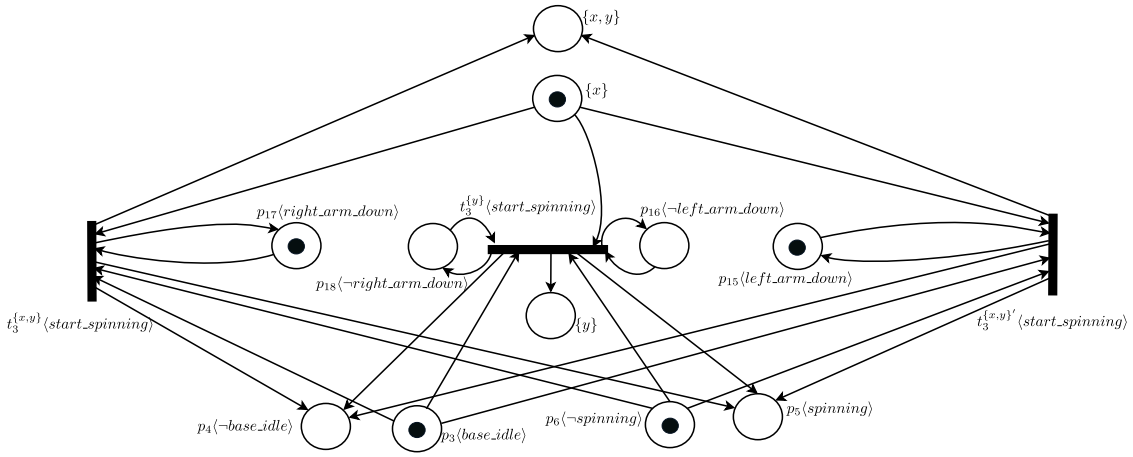

Fig. 4. A fragment of the obtained supervisor

Future work includes creating such a method, where the allowed natural language utterances are translated into LTL and the method presented here is then applied. This can probably be achieved by adapting the structured English defined in [11] to Maggie's domain. Maggie already has a speech recognition skill, hence this appears to be the main issue in the development of the method. Afterwards, we intend to compare this approach with the work presented in [9], where sequences of actions are taught to Maggie by means of speech interaction, where the user exhaustively describes every possible action to be executed. We feel that, by providing a more abstract approach where a set of rules results in a certain behaviour, this approach might be more appealing to older children that already have the capabilities of inferring what might be the consequences of stating a given rule for the robot to fulfil.

REFERENCES

[1] Aris Alissandrakis, Chrystopher L. Nehaniv, Kerstin Dautenhahn, and Joe Saunders. Evaluation of robot imitation attempts: comparison of the system's and the human's perspectives. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 134–141, Salt Lake City, Utah, USA, 2006. ACM.

[2] R. Barber and M.A. Salichs. A new human based architecture for intelligent autonomous robots. In *Proceedings of the The Fourth IFAC Symposium on Intelligent Autonomous Vehicles*, pages 85–90, Sapporo, Japan, 2001.

[3] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, A. Hoffman, C. K. H. Lee, J. Lieberman, and A. Lockered. Working collaboratively with humanoid robots. In *Humanoids '04: Proceedings of the 4th IEEE/RAS International Conference on Humanoid Robots*, pages 253 – 272, Los Angeles, CA, USA, 2004.

[4] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] International Electrotechnical Commission. Preparation of function charts for control systems: Sequential function chart. Technical Committee No. 65: Programmable Controllers - Programming Languages, IEC 61131-3, Ed 2.0, 2003.

[6] Hugo Costelha and Pedro Lima. Modelling, analysis and execution of multi-robot tasks using Petri nets. In *AAMAS '08: Proceedings 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1449–1454, Estoril, Portugal, 2008.

[7] E. Allen Emerson. *Temporal and modal logic. In Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[8] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification*, pages 53–65, London, UK, 2001.

[9] Javi F. Gorostiza and Miguel A. Salichs. Teaching sequences to a social robot by voice interaction. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2009)*, pages 797–802, Toyama, Japan, 2009.

[10] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[11] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. From structured english to robot motion. In *Proceedings of IROS '07: IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717 – 2722, San Diego, California, USA, 2007.

[12] Bruno Lacerda and Pedro Lima. LTL plan specification for robotic tasks modelled as finite state automata. In *Proceedings of Workshop ADAPT - Agent Design: Advancing from Practice to Theory, AAMAS '09: 8th International Conference on Autonomous Agents and Multi-agent Systems*, Budapest, Hungary, 2009.

[13] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[14] M.A. Salichs, R. Barber, A.M. Khamis, M. Malfaz, J.F. Gorostiza, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, and D. Garcia. Maggie: A robotic platform for human-robot social interaction. In *RAM '06: 2006 IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–7, Bangkok, Thailand, 2006.

[15] V. A. Ziparo, L. Iocchi, D. Nardi, P. F. Palamara, , and H. Costelha. Petri net plans: a formal model for representation and execution of multi-robot plans. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 79–86, Estoril, Portugal, 2008.