LTL Plan Specification for Robotic Tasks Modelled as Finite State Automata

Bruno Lacerda*and Pedro U. Lima
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
{blacerda, pal}@isr.ist.utl.pt

Abstract

We introduce a plan specification method for robotic tasks modelled by finite state automata. Each state of a system composed of (multiple) robot(s) situated in an environment is described by a set of propositional symbols. Events associated to transitions drive the state dynamics and represent actions issued by the robot controller or uncontrollable events (e.g., a failure or an action performed by another robot). The plan specifications are written in linear-time temporal logic, enabling a close-to-natural-language description of specifications while creating a sound method of enforcing them, by designing a supervisor, based on discrete event systems theory, that restricts the uncontrolled behaviour of the robot(s) to the subset of acceptable/desirable behaviours.

1 Introduction

A discrete event system (DES) [1] is a discrete state space dynamic system that evolves in accordance with the instantaneous occurrence, at possibly unknown times, of physical events. DES are a suitable framework for the modelling of robotic tasks and, at a logical level of abstraction, supervisory control (SC) has been introduced as a way to restrict a system's open-loop behaviour in order to fulfil a set of performance objectives. SC of a DES modelling robotic tasks can be seen as a formal planning method, where the supervisor enforces specifications by restricting the original set of system (robot+environment) trajectories to a subset of those which meet such specifications. Formal models help designing practical systems because they provide a systematic approach to analysis and design. In robotic systems, though one can design fairly complex behaviours these days, by using graphical tools or high-level modular (e.g., behaviour-based) languages, little or no formal

*The work of this author was supported by the portuguese Fundação para a Ciência e Tecnologia through grant SFRH/BD/45046/2008.

verification is usually possible when using such development systems. DES provide not only a fairly natural way of modelling robot tasks, but also qualitative analysis tools, including formal verification of deadlocks, livelocks, conservation, among others. Furthermore, quantitative analysis of the robot(s) behaviour, e.g., concerning plan reliability or sensitivity to the reliability of some of its composing actions, is also possible. DES theory also provides systematic tools and methods to design supervisors that guide the open loop system behaviour so as to avoid undesirable states and/or event sequences. All these features enable designing and analysing large-size robot systems models which are not tractable using "manual" methods, thus providing guarantees and meeting specifications. DES theory has been used successfully for years to model and analyse manufacturing systems. Our goal is to extend those results, which concern more structured systems, to more unstructured classes of robotic systems, such as home robots and outdoor robots, including cooperating teams. One problem with DES supervision is that it is usually hard to express in a natural way the desired specifications. LTL is a step forward towards using natural languages to specify the requirements for robotic systems. In [7], we presented a method to synthesize a supervisor for a DES modelled by a finite state automaton (FSA) [4], based on linear-time temporal logic (LTL) [2] specifications over the FSA's event set. That method allowed the supervision of more complex systems, since the translation of our performance objectives to LTL is, in general, more immediate than the standard method of directly building an FSA that represents the specifications. However, the fact that the LTL formulas are only written over the FSA's event set forces us to write some of the LTL formulas in a somewhat artificial way, namely the specifications that refer to a state of the system that must not be reached. The main purpose of this paper is to extend that method in such a way that these kind of specifications can be easily translated into an LTL formula. To achieve this we associate a propositional description of the environment to each state of the FSA and write our LTL specifications over those propositional descriptions, in addition to the FSA event set. This will allow us, for example, to refer to a state by the propositional symbols that describe it. Significant work has been done in applying temporal logic to different robotics fields, e.g., motion planning [6], state-based supervision [5] and planning for temporally extended goals [9]. The difference to our work is that our goal is to coordinate the actions available to a team of robots so that they effectively cooperate among each other. This implies a tighter coordination of the robot's actions, which we handle by writing specifications not only over propositions describing the system but also directly over the actions available at each time.

2 Preliminaries

We start by giving a brief overview propositional logic and LTL [2]. Let Π be a set of propositional symbols. Regarding propositional logic, we just note that we will evaluate the propositional formulas over valuations $v \subseteq \Pi$. LTL is an extension of propositional logic which allows reasoning over infinite sequences of states $\sigma \in 2^{\Pi^\omega}$. LTL formulas extend propositional formulas by adding an X operator, which is read "next" and requires the formula it precedes to be true in the next state and an U operator which is read "until" and requires its first argument to be true until a state where the second is true. As usual, we define the "eventually" and "always" operators, F and G, by abbreviation. We will be interested in the finite sequences that are prefixes of at least an infinite sequence that satisfies φ, which we call consistent sequences with φ. An FSA is a six-tuple $G = \langle X, E, f, \Gamma, x_0, X_m \rangle$ where X is the set of states, E is the set of events, f is the transition function, Γ is the active event function, x_0 is the initial state and X_m is the set of marked states, as defined in [7]. We assume that our FSA have a propositional description over a set of propositional symbols $\mathcal{P}$. This is simply a function $V : X \to 2^{\mathcal{P}}$ such that, for $x \in X$, $V(x)$ is the set of propositional symbols that are true in x. One should notice that for the parallel composition $G_1 \parallel G_2$, the propositional description of a state (x_1, x_2) - which represents that G_1 is in state x_1 and G_2 is in state G_2 - is the union of the propositional descriptions of both of these states. Büchi automata have the same structure as FSA, but generate and mark ω-languages, i.e., languages of infinite strings. It is a known fact that, given an LTL formula φ, there exists a Büchi automaton B_φ such that $\sigma \Vdash \varphi$ if and only if $\sigma \in L_m(B_\varphi)$. In our implementation, we use the method described in [3]. We will assume that the transitions of this Büchi automaton are labelled with propositional formulas. Further details can be found in [3].

3 Supervisor Synthesis

In our method, we start with a DES modelled as an FSA equipped with a propositional description, consisting of all the possible behaviours of a set of robots in their environment. We will model the system as an FSA $G = \langle X, E, f, \Gamma, x_0, X_m \rangle$ with a propositional description V over $\mathcal{P}$ and will specify the intended controlled behaviour as a set of n LTL formulas $\Phi = \{\varphi_1, ..., \varphi_n\}$ built over the set of propositional symbols $\Pi = E \cup \mathcal{P}$. This means that we will be able to refer both the state propositional descriptions and the sequence of event firings of the system. For each φ_i, we build the equivalent Büchi automaton B_{φ_i} and, using the DNF description of the events, the supervisor S_Φ is implemented by the FSA given by $(G \parallel_{spec} B_{\varphi_1} \parallel_{spec} B_{\varphi_2} \parallel_{spec} ... \parallel_{spec} B_{\varphi_n})^{\uparrow A}$. ${S_\Phi}^{\uparrow A}$ is the automaton that generates the supremal controllable language with respect to $L(G)$ and $L(S_\Phi)$, guaranteeing that our supervisor is admissible, as described in [1]. Given an FSA G with a propositional description V over $\mathcal{P}$ modelling the system and the Büchi automaton $B_\varphi = \langle X^B, 2^{(E \cup \mathcal{P})}, f^B, \Gamma^B, x_0^B, X_m^B \rangle$ obtainned by an LTL formula φ, the specification parallel composition of G and B_φ is the FSA $G \parallel_{spec} B_\varphi = \langle X \times 2^{X^B}, E, f', \Gamma', (x_0, \{x_0^B\}), X_m' \rangle$, where, for $x \in X$, $\overline{x} \subseteq X^B$ and $e \in E$:

$$f'((x, \overline{x}), e) = \begin{cases} (f(x,e), g(x, \overline{x}, e)) & \text{if } g(x, \overline{x}, e) \neq \emptyset \\ \text{undefined} & \text{if } g(x, \overline{x}, e) = \emptyset \end{cases}$$

with $g : X \times 2^{X^B} \times E \to 2^{X^B}$ defined as:

$$g(x, \overline{x}, e) = \bigcup_{x^B \in \overline{x}} \; \bigcup_{e^B \in \Gamma^B(x^B)} \{ f^B(x^B, e^B) \mid V(f(x,e)) \cup \{e\} \Vdash e^B \}$$

$G \parallel_{spec} B_\varphi$ restricts G's behaviour to the one specified by φ, i.e., for all $e_0 e_1 ... e_n \in L(G \parallel_{spec} B_\varphi)$, $(V(f(x_0, e_0) \cup \{e_0\})(V(f(x_0, e_0 e_1)) \cup \{e_1\})...(V(f(x_0, e_0 e_1 ... e_n)) \cup \{e_n\})$ is consistent with φ.

Example. Consider a soccer team of n robots. The objective is to reach a situation

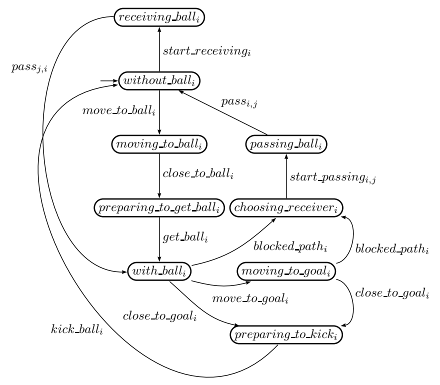

Figure 1: FSA model for robot R_i

in which one of the robots is close enough to the goal to shoot and score. When a robot does not have the ball in its possession, it can move to the ball until it is close enough to take its possession or get ready to receive a pass from a teammate. When it has the possession of the ball, it can shoot the ball, take the ball to the goal if there is no opponent blocking its path or choose a teammate to pass the ball and, when it is ready to receive, pass it. For simplicity, we assume that, when a robot shoots the ball, the team loses its possession and that the opponents are only able to block paths. Figure 1 depicts a possible model for robot R_i situated in a soccer game, with a propositional description over $\mathcal{P}_i = \{moving2ball_i, hasball_i\}$. We associate states $without\_ball_i$ and $receiving\_ball_i$ with $\emptyset$, states $moving\_to\_ball$ and $preparing\_to\_get\_ball_i$ with $\{moving2ball_i\}$ and states $with\_ball$, $preparing\_to\_kick_i$,$moving\_to\_goal_i$ and $passing\_ball_i$ with $\{hasball_i\}$. An FSA model for the whole team is given by $T = R_1 \parallel R_2 \parallel ... \parallel R_n$. The events $close\_to\_ball_i$, $close\_to\_goal_i$ and $blocked\_path_i$ are caused by changes in the environment, hence uncontrollable. The remainning events are controllable events and correspond to the actions available to each robot. One may define the following specifications, which are useful to improve the team's performance, for each robot R_i. First a robot will move to the ball if and only if the ball is not in the team's possession and no other teammate is moving it:

$$\varphi = (G(\bigvee_{i=1}^{n} moving2ball_i \vee \bigvee_{i=1}^{n} hasball_i) \Rightarrow (X(\neg(\bigvee_{i=1}^{n} move\_to\_ball_i))))$$

Second, robot R_i will not get ready to receive a pass, unless one of its teammates decides to pass the ball and, in this case, it will be ready to receive the pass as soon as possible:

$$\psi_i = ((\neg start\_receiving_i) \wedge (G((\bigvee_{\substack{j=1 \\ j \neq i}}^{n} start\_passing_{j,i}) \Leftrightarrow (X start\_receiving_i))))$$

Hence, the controlled system will be given by $(T \parallel_{spec} B_{\varphi} \parallel_{spec} B_{\psi_1} \parallel_{spec} ... \parallel_{spec} B_{\psi_n})^{A\uparrow}$.

4 Discussion and Conclusion

In this paper, we presented an extension of our method to perform LTL based SC of DES. This extension improves the previous method by allowing us, in addition to the previous possibility of reasoning over the sequence of events, to reason directly over the system's states, through a propositional description of each state, hence improving the method's expressiveness. We provided an example to show how our method can be used to coordinate a team with an arbitrary number of robots in order for it to efficiently perform a given task. Analysing this example, one can conclude that, in spite of providing a more natural way to perform SC, the designer still has to have some sensibility about how to define a useful state propositional description, and that the writing of the LTL formulas requires a good understanding of the structure of the system. Some familiarity with LTL is also required but, in our opinion, this is something that can be easily achieved. This method presents a few limitations that arise from the modelling formalism: FSA are not the most suitable formalism to model multi-robot tasks and, since we model the system from a logical point of view, we cannot analyse quantitative properties of the system. There are also some specifications that cannot be implemented by our method, in particular any specification that involves "counting" cannot be enforced in this framework, e.g., we cannot enforce that state x cannot be visited by the system more times than state y. Though most of the LTL formulas relevant to perform SC over these kind of systems yield a relatively small Büchi automaton, we noticed that a particular kind of specifications that can be useful in our case leads us to a state explosion of the Büchi automaton: formulas of the form $(G(p \Rightarrow X_n q))$, where X_n is the application of the "next" operator n times, yield a Büchi automata with 2^n states. The next step to our work will be an improvement to the modelling framework, from FSA to Petri nets, adopting a more suitable formalism for multi-robot task modelling. We will be concerned in defining a supervision method for Petri nets based in temporal logic specifications, possibly adapting methods for Petri nets model-checking with LTL and existing supervisory control theory for Petri nets.

References

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

[2] E. A. 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.

[3] P. Gastin and D. 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.

[4] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 2006.

[5] S. Jiang and R. Kumar. Supervisory control of discrete event systems with CTL* temporal logic specifications. *SIAM Journal on Control and Optimization*, 44(6):2079–2103, 2006.

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

[7] B. Lacerda and P. U. Lima. Linear-time temporal logic control of discrete event models of cooperative robots. *Journal of Physical Agents*, 2(1):53–61, 2008.

[8] M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In *Proc. of the 17th Int. Joint Conf. On Artificial Intelligence (IJCAI'01)*, Seattle, WA, USA, 2001.