Designing Petri Net Supervisors for Multi-Agent Systems from LTL Specifications

(Extended Abstract)

Bruno Lacerda*
Institute for Systems and Robotics
Instituto Superior Técnico
Lisboa, Portugal
blacerda@isr.ist.utl.pt

Pedro U. Lima
Institute for Systems and Robotics
Instituto Superior Técnico
Lisboa, Portugal
pal@isr.ist.utl.pt

ABSTRACT

In this paper, we use LTL to specify acceptable/desirable behaviours for a system modelled as a Petri net, and create a Petri net realization of a supervisor that is guaranteed to enforce them, by appropriately restricting the uncontrolled behaviour of the system.We illustrate the method with an application to the specification of coordination requirements between the members of a team of simulated soccer robots.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Discrete Event*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Temporal Logic*

General Terms

Design, Theory

Keywords

Petri Nets, Supervisory Control, Linear Temporal Logic

1. INTRODUCTION

When designing multi-agent systems (MAS), concepts such as concurrency, parallelism, synchronisation or decision making are of central importance. In order to be able do deal with these notions as the systems become more complex, one needs a formal approach to modelling, analysis and controller synthesis. In this paper, we use Petri nets (PN) to model and analyse MAS, due to to the fact that PNs are particularly well suited to model distributed systems and handle all the above concepts. Given a PN model of a MAS and a natural language specification for it to fulfil, we will be interested in synthesising a PN realization of a supervisor based in discrete event system (DES) theory that restricts the behaviour of the system such that the specification is satisfied. The construction of this supervisor is done by translating the natural language specification into a linear temporal logic (LTL) formula and then composing its equivalent Büchi automaton (BA) with the the PN model in such a way that the composition complies with the LTL specification. There has been a considerable amount of work on the control of PNs. For example, in [3] a method where the specifications are written as linear constraints on the reachable markings of the system and the number of firings of each transition is defined and in [2] a study on the advantages and limitations of using PNs as a tool to realize supervisors is provided. There have been several approaches to the use of temporal logic as a tool to specify and synthesize goal behaviours. The work presented in [6] introduces a planning algorithm over a domain given as an non-deterministic finite state automaton (FSA) where the states correspond to sets of propositional symbols and the goal is given as a temporal logic formula over those symbols. In both of this work, the temporal logic formulas are written only over the state space of the system, thus direct reasoning about sequences of events is not allowed. In [4], a motion planning method where the goals are defined as LTL formulas is presented. The work in [5] also deals with motion planning with temporal logic goals but allowing the robot to also react to sensor readings and perform actions other than moving. This approach, using DES models, reduces the involved complexity in comparison with hybrid systems models, by only taking the (discrete) sequences of actions into account.

2. CONSTRUCTING THE LTL BASED PN SUPERVISOR

We will explain the method through an example. Consider a soccer team of n robots. The goal is to reach a situation 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 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. In Figure 1, we present the PN N_i for one of the robots. We depict both events labels, associated to transitions, and state description symbols, associated to places, as $\langle . \rangle$. The LTL formulas will be written

*Work partially funded by FCT (ISR/IST pluriannual funding) through the PIDDAC Program funds and FCT grant FRH/BD/45046/2008.

Cite as: Designing Petri Net Supervisors for Multi-Agent Systems from LTL Specifications (Extended Abstract), Bruno Lacerda, Pedro U. Lima, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1253-1254.

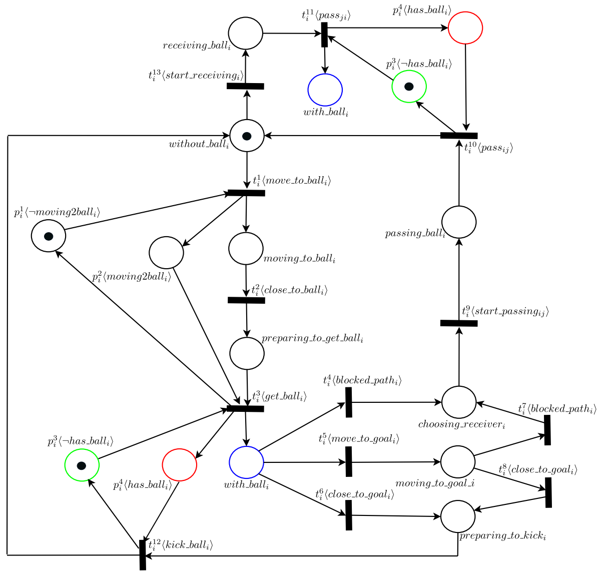
$t_i^{11}\langle pass_{ji}\rangle$
$p_i^4\langle has\_ball_i\rangle$
$receiving\_ball_i$
$t_i^{13}\langle start\_receiving_i\rangle$
$p_i^3\langle \neg has\_ball_i\rangle$
$with\_ball_i$
$without\_ball_i$
$t_i^{10}\langle pass_{ij}\rangle$
$t_i^1\langle move\_to\_ball_i\rangle$
$p_i^1\langle \neg moving2ball_i\rangle$
$passing\_ball_i$
$moving\_to\_ball_i$
$p_i^2\langle moving2ball_i\rangle$
$t_i^2\langle close\_to\_ball_i\rangle$
$t_i^9\langle start\_passing_{ij}\rangle$
$preparing\_to\_get\_ball_i$
$t_i^4\langle blocked\_path_i\rangle$
$t_i^3\langle get\_ball_i\rangle$
$choosing\_receiver_i$
$t_i^7\langle blocked\_path_i\rangle$
$p_i^3\langle \neg has\_ball_i\rangle$
$t_i^5\langle move\_to\_goal_i\rangle$
$t_i^8\langle close\_to\_goal_i\rangle$
$p_i^4\langle has\_ball_i\rangle$
$with\_ball_i$
$moving\_to\_goal\_i$
$t_i^6\langle close\_to\_goal_i\rangle$
$preparing\_to\_kick_i$
$t_i^{12}\langle kick\_ball_i\rangle$

Figure 1: PN model for robot i. Places depicted with the same color represent the same place, we separated them to improve readability.

over the union of these two sets. A PN model for the whole team is given by the parallel compositions of the PN models for each robot, synchronizing transitions with common event labels and keeping the state description. The events $close\_to\_ball_i$, $close\_to\_goal_i$ and $blocked\_path_i$ are caused by changes in the environment, hence uncontrollable. The remaining events are controllable events. For each N_i, we define the set E_c^i as the set of controllable events of N_i. This set is used to guarantee the supervisor admissibility: instead of writing that a controllable event $e \in E_c^i$ must occur, we write that all other controllable events in E_c^i cannot occur until the occurrence of e. One may define the following specifications: For the whole team, 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 towards it:

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

For each robot N_i, it will not get ready to receive a pass if none of its teammates wants to pass it the ball:

$$\psi_i = G((\bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg start\_passing_{j,i}) \Rightarrow (X\neg start\_receiving\_i))$$

For each robot N_i, when one of the teammates decides to pass it the ball, it will be ready to receive the pass as soon as possible:

$$\gamma_i = G((\bigvee_{\substack{j=1 \\ j\neq i}}^{n} start\_passing_{j,i}) \Rightarrow$$
$$(X((\bigwedge_{e'\in E_c^i\setminus\{start\_receiving_i\}} \neg e')U start\_receiving_i)))$$

The supervisors are built by appropriately composing the BA obtained for each LTL specification[1] above with the PN model of the system. This composition yields a PN that simulates a run in parallel of the BA and the PN modelling the system, only allowing the occurrence of the PN transitions that lead the system to a sequence of events plus state description symbols that satisfies the LTL formula. To build such a PN, we compare each transition of the PN model of the system with the labels of the BA transitions (encoded as propositional logic formulas) and add reflexive-arcs[2] between the PN transitions and the places representing a truth value of a state description symbol that is needed for the BA transition label to be satisfied. Hence, we only allow the firing of the PN transition when it leads us to a marking for which the set of true state description symbols, in conjunction with the fired event, satisfies the BA transition. If it is not possible to satisfy the BA transition, no transition is added to the PN supervisor. We were able to build the supervisor to a team of up to 10 robots. Even though the supervisors are large, we were able to build them in a decent amount of time and for an already large number of robots[3]. We argue that without a formal method for the construction of the supervisors that automatically guarantees that the specifications are met, the construction of supervisors for this number of robots would not be feasible.

3. CONCLUSION AND FURTHER WORK

We presented a method to build PN supervisors that are guaranteed to fulfil LTL specifications. This method allows the designer to specify intricate behaviours, e.g., coordination rules, in a close-to-natural-language formalism, as illustrated in an application example. Our main goal for future work is to add uncertainty to the models in order to provide a method that is robust both to failures in performing actions and errors in sensor readings.

4. REFERENCES

[1] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *CAV '01: Proc. of 13th Int. Conf. on Computer Aided Verification*, pages 53–65, London, UK, 2001.

[2] A. Giua and F. DiCesare. Blocking and controllability of Petri nets in supervisory control. *IEEE Transactions on Automatic Control*, 39(4):818–823, 1994.

[3] M. V. Iordache and P. J. Antsaklis. Supervision based on place invariants: A survey. *Discrete Event Dynamic Systems*, 16(4):451–492, 2006.

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

[5] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[6] M. Pistore and P. Traverso. Planning as model checking extended goals in non-deterministic domains. In *Proc. of the 17th Int. Joint Conf. On Artificial Intelligence (IJCAI'01)*, pages 479–484, Seattle, WA, USA, 2001.

[1] The BA are obtained using the LTL2BA algorithm [1].

[2] A reflexive arc between t and p is a pair of arcs, one from p to t and the other from t to p.

[3] For 10 robots, the supervisors were built in around 2h30m, using an Intel(R) Core(TM) i5 CPU 750 @ 2.67GHz.