Modelling, Analysis and Execution of Multi-Robot Tasks using Petri Nets

(Short Paper)

Hugo Costelha
Inst. for Sys. and Robotics, IST, Lisboa, Portugal
Polytech. Inst. of Leiria, ESTG, Leiria, Portugal
hcostelha@isr.ist.utl.pt

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

ABSTRACT

This paper introduces Petri net (PN) based models of cooperative robotic tasks, namely those involving the coordination of two or more robots, thus requiring the exchange of synchronisation messages, either using explicit (e.g., wireless) or implicit (e.g., vision-based observation of teammates) communication. In the models, PN places represent primitive actions, subtasks and predicates set by sensor readings and communicated messages. Events are associated to PN transitions. The PN models can be used for task planning, plan execution and plan analysis. Different PN views enable the analysis of different properties. In this work we focus on plan analysis, namely on properties such as boundedness and liveness, corresponding to checking if resources usage is stable and plans have no deadlocks, as well as on stochastic performance, concerning the plan success probability. One novel feature of our work is that the analysis consists of composing several small action PN models with environment PN models, leading to a closed loop robot team/environment analysis methodology. Examples of application to simulated robotic soccer scenarios are presented.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Petri nets*

General Terms

Design, Performance, Verification

Keywords

Multi-Robot Systems, Petri nets, Modelling, Analysis

1. INTRODUCTION

Formal task design methods for general robotic tasks enable a systematic approach to modelling, analysis and design, scaling up to realistic applications, of formal properties, and design from specifications. Discrete event systems [1] provide an appropriate framework for such goal. Several approaches described in the literature use finite sate automata (FSA) to model robot behaviors. However, frequently they do not take advantage of the formal properties associated to such models, using them only as a helpful design tool.

Discrete event models are especially relevant to model teamwork, as communication signals exchanged by the involved teammates often involve synchronization events and transition between discrete states. This is particularly true for PNs. In PNs, the state information is distributed among a set of places which capture key conditions governing the system, and PNs have increased modularity for model-building. Furthermore, an automaton can always be represented as a PN, but not all PNs can be represented as finite-state automata, therefore the language expressive power is greater for PNs than for FSA, enabling richer behavior models [1]. PNs have been used before to model robotic tasks [3, 4], both for manipulators and mobile robots, taking advantage of the models to explore task analysis topics.

We have introduced a PN based framework for modelling, analysis and execution of PN tasks in [5]. In this paper we extend the proposed framework with multi-robot cooperative task models. Some additional building blocks that enhance and modularize even further the task design process of individual robotic tasks are also described. In the developed framework, sensor information and events are seen, respectively, as resources and transitions. The same can be done regarding communication. Thus, multi-robot cooperative tasks are modeled by adding extra information to the individual task models to account for synchronisation, either by explicit or implicit communication. The closest work to ours can be found in [6], however the authors focus there on the design of the tasks for execution purposes only, and do not perform task analysis.

The paper starts by introducing PN models and the basic building blocks of the framework in Section 2. Section 3 details how individual tasks can be modelled, analysed and executed. Section 4 introduces the extensions needed for multi-robot tasks. Finally, Section 5 shows results obtained in a robotic soccer scenario, followed by the conclusions and future work in Section 6.

2. PETRI NETS

Petri nets are the base formalism used in this work. We use both Marked Ordinary Petri nets (MOPNs) [1], wich are suited for qualitative analysis, and Generalised Stochastic Petri nets (GSPNs) [7], suited for performance analysis.

The GSPN marking is a semi-Markov process with a dis-

Cite as: Modelling, Analysis and Execution of Multi-Robot Tasks using Petri Nets (Short Paper), Hugo Costelha and Pedro Lima, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16.,2008,Estoril,Portugal,pp.1187-1190.

crete state space given by the reachability graph of the net for an initial marking [7]. A Markov chain can be obtained from the marking process, and the transition probability matrix computed by using the firing rates of the exponential timed transitions and the probabilities associated with the random switches. This enables the use of tools already available to analyse Markov chains directly with the GSPN, instead of relying on e.g., Monte Carlo simulation.

We first define some base construction models to assist us on the design of the robotic tasks, namely *Macro Places* and *Predicate Places* [5]. Macro places are special places which allow an hierarchical view and use of PN models. A macro place is actually a place holder for an entire PN model, which can also contain other macro places, thus introducing models with unlimited depth. We propose a definition of macro places based on *connection places*, which can be *Input* places, *Output* places, or both. These connection places define where to connect incoming arcs (inputs) and outgoing arcs (outputs) when replacing a macro place by its corresponding PN model.

A macro place, depicted as a dashed place, is used as a macro-model of a PN model, allowing a PN model to replace, or be replaced by, a single place. This PN model must have at least one input and one output places or, at least one input/output place, in order to be replacable by a macro place. Note that a place can be both a connection place and a macro place. The term *expanding* a macro place refers to replacing the macro place by its corresponding PN model.

3. INDIVIDUAL ROBOTIC TASKS

Under this framework, the robotic task models are organised in layers with different degrees of abstraction, which include an Environment layer (with PN-based models of the robot environment), Action Executor and Coordination layers (with PN-based robot action models) and an Organisation layer (concerning PN-based role assignment and goal selection models). The Environment layer models are used solely for analysis purposes. The other layers are directly related with what runs in the robot, each of them including several (usually small) models. Specification of individual robotic tasks is achieved by designing the various models separately and by building a PN-based task plan which uses those models, as detailed in [5].

Analysis of individual robot tasks is done by building a single PN modelling the entire task execution. The task plan macro places are expanded until there are no macro places and then composed with the Environment models, considering that all predicate places with the same label are the same, and all the other components are different, regardless of their label. The resulting PN is analysed using available PN analysis techniques. These can be logical properties of the task execution, like deadlocks, conservation properties, livelocks, etc., or based on Markov chain analysis techniques for quantitative properties. Given that the marking graph of a GSPN is a semi-Markov process, a Markov chain results from the original PN, which is analysed for quantitative properties, such as mean time to reach a given state, probability of reaching a given state, etc..

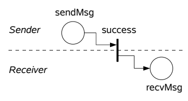
4. MULTI-ROBOT TASKS

The main difference between individual and cooperative multi-robot tasks, is that synchronisation must occur among the robots during task execution. Synchronisation occurs through the use of communication, either explicitly or implicitly. Explicit communication happens when a robot (the sender) sends a message directly to the other robot(s), usually using ethernet or wireless communications. Implicit communication happens when a robot, or robots, (the receivers) perceive some situation regarding the sender robot. As such, in order to model multi-robot tasks with our PN-based framework, we first introduce communication models.

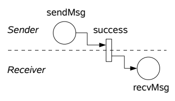
4.1 Communication Models

The major problem when using communication is the time information takes to go from the sender to the receiver, which, theoretically, can go from zero time to infinite time (communication failure). To model communication, we considered three different communication models, which cover this time range. The base concept in these models is that a robot has a predicate place with a given value and wishes to transmit that information to a teammate. The teammate, upon receiving the information, gets its predicate updated to the same value as its teammate.

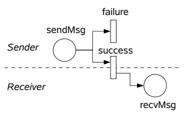
The simplest communication model is presented in Fig. 1a. Here the communication is considered instantaneous and always successful. Increasing the model complexity by adding a probabilistic arrival time for the communication, results in the model depicted in Fig. 1b. In this case, communications are still considered always successful. The full communication model, which adds the failure possibility to the previous model, is presented in Fig. 1c.



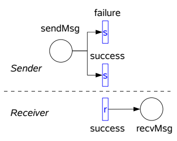
(a) Deterministic communication model without failures.



(b) Communication model with probabilistic time and without failures.



(c) Full communication model with probabilistic time and failures.



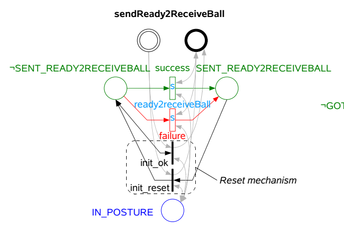
(d) Separate view of the full communication model.

Figure 1: Communication models.

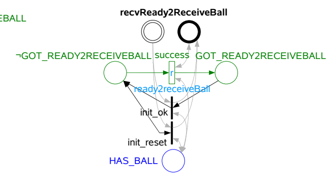
Given the various communication models, we can choose which one to use, according to the context where the model is being applied and the property we wish to analyse. Communication models are distributed as shown in Fig. 1d, thus the communication transitions include a tag to distinguish if the transition belongs to the sender or the receiver.

4.2 Communication Actions

In order to use the communication models to model direct communication between robots during a relational task, we define Communication Actions, which will be used to establish the required synchronisation. For each sending communication model there must be a receiving communication action model. As an example, see the Action Executor level models of actions `sendReady2ReceiveBall` and `recvReady2ReceiveBall`, for a robotic soccer scenario, in Fig. 2a and Fig. 2b respectively.



(a) Action `sendReady2-ReceiveBall`.



(b) Action `recvReady2Recei-veBall`.

Figure 2: Communication actions.

These actions, besides the specifications already defined for ordinary actions, include an output place (represented by a bold circle) in addition to the usual input/output (double circle) place and a reset mechanism. The additional place is required to distinguish the initial action selection (token in the double circled place) from signaling that the action is actually running (token in the bold circled place). This way, the send/receive predicates are reset to their initial values (not sent, not got) immediately upon initialization. Two init transitions are also required to distinguish the cases when the communication actions started with the send/receive predicates already set to their initial values or not.

4.3 Task Plans

With the introduction of the communication models and communication actions, specifying a multi-robot task is similar to the specification of individual robot tasks. The major difference is that we need to use communication actions to ensure that the subtasks running during a multi-robot task execution are synchronised. For now we are assuming that the choice of running a relational task was already done, and focus on the multi-robot task execution analysis.

4.4 Analysis

The analysis of multi-robot tasks in this framework is similar to the individual robot tasks case, adding the introduction of the communication models and actions. The difference relies on the need to prefix all the place labels identifying the robot they belong to, and the expansion of the communication actions need an additional step.

Although the action places are always considered different places, regardless of their label, these are also prefixed with the robot label, since different robots can have different action models. During the expansion of the macro places for analysis, all the communication actions are replaced by their analysis version instead of their original version. If we have more than two robots, then a selection mechanism must be used to select to which robot, or robots, the message is to be sent. The user never needs to see the analysis versions of the actions, since these are used only internally for analysis, and are automatically created from their original versions.

5. APPLICATION TO ROBOTIC SOCCER

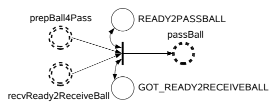
5.1 Setup

Consider a pass example between two robots, the kicker and the receiver. Given two subtasks, *coordinatedKick*, for the kicker, and *coordinatedReceive*, for the receiver, a two-robot Pass task plan corresponds to a single *coordinatedPass* relational task, which consists of running both subtasks in parallel, one in each robot. The key here is to make sure that both subtasks run synchronously. We assume that some higher level took the decision that the robots should commit with the coordinated pass task, and will focus on the task execution analysis, keeping the critical sections synchronised.

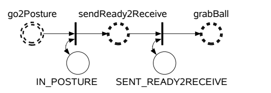
First we define a set of actions, needed for the pass relational task: `standBy`, do nothing action; `prepBall4Pass` - the robot which has the ball, the kicker, gets ready to pass the ball to the receiver; `sendReady2ReceiveBall` - the receiver acknowledges that it is ready to receive the ball(Fig. 2a); `recvReady2ReceiveBall` - waits for a communication from the receiver to know it is ready to receive the ball (Fig. 2b); `passBall` - passes the ball to another robot. In this case we considered a simple version, where passes are only done from near the own goal to near the opponent goal; `go2ReceivingPosture` - the robot moves to a destination posture, which is good for receiving the ball. In this case we also considered a simpler version, where the receiving posture is near the opponent's goal; `grabBall` - the robot grabs the ball.

Subtask *coordinatedKick* is obtained by running actions `prepBall4Pass` and `recvReady2ReceiveBall` in parallel, followed by `passBall` upon getting predicates `READY2PASSBALL` and `GOT_READY2RECEIVEBALL` to true. The *coordinatedReceive* subtask is formed by a sequence of actions, starting with `go2ReceivingPosture`, followed by `sendReady2ReceiveBall` when predicate `IN_POSTURE` gets true, followed by `grabBall` when `SENT_READY2RECEIVEBALL` gets true. The PN models of both subtasks are shown in Fig. 3a and Fig. 3b.

Regarding communication, the relevant actions for the *coordinatedPass* relational task are `recvReady2ReceiveBall` and `sendReady2ReceiveBall`, the two communication actions detailed previously.



(a) *coordinatedKick* subtask model.



(b) *coordinatedReceive* subtask model.

Figure 3: Individual pass substasks.

5.2 Task Plan & Analysis

We consider a scenario where both robots are already set up for the execution of the pass, and analyse its execution. As such, we place the kicker robot near its own goal with the ball and the receiver robot near the opponent goal. Both robots will start immediately in the *coordinatedPass* relational task, thus resulting in the Pass task plan depicted in Fig. 4. Note that this is the team task plan, as the top and bottom places represent, respectively, the kicker and receiver task plans.

As said previously, the analysis of this task plan is performed by expanding its macro places, taking into attention the communication models, obtaining a single PN, which is analysed both for qualitative and quantitative properties. Note that the obtained PNs are supposed not to be seen by the user, given that they are created automatically for analysis purposes only.

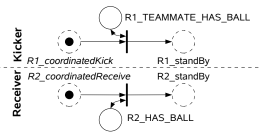


Figure 4: Pass task plan.

Table 1: Plan success probability vs transition rates.

#	Env.	Action success	Comm. success	Comm. failure	Plan success probability
1	1	10	1	1	0.11
2	1	10	1	10	0.04
3	1	10	10	1	0.39
4	1	10	10	10	0.23
5	1	10	100	0.01	0.51
6	1	10	10000	0.0001	0.51
7	1	20	10000	0.0001	0.70

5.3 Results

Given the various models of the actions and the analysis techniques described earlier, we obtained several results concerning the expected plan success probability using the TimeNET tool [8]. We started by testing with a deterministic environment, i.e., an environment without stochastic transitions. Given that the only actions which directly include failures are the communication actions, the plan success probability in this case depends only on the relation between the two communication rates, success and failure. In that case, the plan success probability is given by

$$P_{Plan\ success} = \frac{\lambda_{comm\ success}}{\lambda_{comm\ success} + \lambda_{comm\ failure}}$$

Using a probabilistic environment where it was considered that the ball could move uncontrolled and where losing the ball is also uncontrollable, we have a more realistic model. In this case we experimented with different rates for the Environment stochastic transitions, for the Actions success transitions and, for the communication success and communication failure transitions, obtaining the results in Table 1. In the table we can see the influence of the various rates in the plan success probability. It clearly shows the expected result, i.e, the communication plays an important role in the success of the plan. Furthermore, in such an environment, high success probabilities are always difficult to achieve. Experiment 4, 5 and 6 clearly show the influence of increasing the proportion between the success communication rate and the other rates. Looking at experiment 5 and 6, we see that, at some point, increasing that proportion does not yeld relevant improvements to the plan success probability. In that case the other option is to improve the success of the other actions, as shown by experiment 7.

Given that the plan is purely sequential, in terms of action execution we always end in a deadlock. In the deterministic environment case the resulting PN has two deadlocks, corresponding to the successful termination (both robots in the `standBy` action) and unsuccessful termination (`R1` in the *`coordinatedKick`* subtask and R2 in the `coordinatedReceive` subtask) of the plan. In the probabilistic case, given that the ball keeps moving around the field, the previous situations correspond to livelocks. In that case, R2 will reach `standBy` since the ball will eventually pass near it, if given enough time.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we extended an earlier PN-based framework to model robotic tasks involving multiple robots, by introducing communication models and communication actions, which allowed us to model and analyse the execution of multi-robot tasks.

We obtained results in a robotic soccer scenario which show the analysis potential of the proposed framework regarding multi-robot tasks. In order to analyse more complex multi-robot tasks, with relational subtasks interleaved with individual subtasks and more than two robots, one needs to incorporate selection and commitment maintenance mechanisms. These mechanisms already exist in the literature [2] and we are developing a PN model which will incorporate them in our model. When analysing the complete models we will also be able to extract properties concerning the communication protocols and commitment maintenance.

7. ACKNOWLEDGEMENTS

This work was supported by the Portuguese Fundação para a Ciência e Tecnologia under under grant SFRH/BD/12707/2003 and ISR/IST pluriannual funding through the POS_Conhecimento Program that includes FEDER funds.

8. REFERENCES

[1] C. Cassandras, S. Lafortune. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.

[2] Philip R. Cohen; Hector J. Levesque, *Teamwork*, Nous, vol. 25, no. 4, pp. 487-512, 1991.

[3] F. Wang et al, *A Petri-Net Coordination Model for an Intelligent Mobile Robot*, IEEE Transactions on Robotics and Automation, Vol. 9, No. 3, pp. 257-271, 1993.

[4] L. Montano, F. García, J. Villaroel, *Using the Time Petri Net Formalism for Specification, Validation, and Code Generation in Robot-Control Applications*, The International Journal of Robotics Research, vol. 19, no. 1, pp. 59-76, 2000.

[5] H. Costelha, P. Lima, *Modelling, Analysis and Execution of Robotic Tasks using Petri Nets*, Proc. of IROS 2007 - IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1449-1454, San Diego, CA, USA, 2007.

[6] V. Ziparo, L. Iocchi, *Petri Net Plans*, Proc. of the Fourth International Workshop on Modelling of Objects, Components, and Agents, pp. 267-290, Turku, Finland, 2006.

[7] N. Viswanadham, Y. Narahari, *Performance Modelling of Automated Manufacturing Systems*, Prentice Hall, 1992.

[8] A. Zimmermann, *A Software Tool for the Performability Evaluation with Stochastic Petri Nets*, http://pdv.cs.tu-berlin.de/˜timenet/, 2001.