Teamwork Design Based on Petri Net Plans

Pier Francesco Palamara[1], Vittorio A. Ziparo[1], Luca Iocchi[1], Daniele Nardi[1], and Pedro Lima[2,⋆]

[1] Dipartimento di Informatica e Sistemistica – "Sapienza" University of Rome, Italy
{ziparo,iocchi,nardi}@dis.uniroma1.it
[2] Institute for Systems and Robotics - ISR/IST, Lisbon, Portugal
pal@isr.ist.utl.pt

Abstract. This paper presents a design of cooperative behaviors through Petri Net Plans, based on the principles provided by Cohen and Levesque's Joint Commitments Theory. Petri Net Plans are a formal tool that has proved very effective for the representation of multi-robot plans, providing all the means necessary for the design of cooperation. The Joint Commitment theory is used as a guideline to present a general multi-robot Petri Net Plan for teamwork, that can be used to model a wide range of cooperative behaviors. As an example we describe the implementation of a robotic-soccer passing task, performed by Sony AIBO robots.

1 Introduction

The design of complex robotic behaviors in dynamic, partially observable and unpredictable environments is a crucial task for the development of effective robotic applications. The annual RoboCup soccer competitions provide an ideal testbed for the development of robotic behavior control techniques, as the design of behaviors in the robotic-soccer environment requires the definition of expressive plans for the performance of complex tasks.

Petri Nets [5] are an appealing modeling tool for Discrete Events Systems, that has been used in several works for the modeling of robotic behaviors. [2] provides an interesting formal approach for the modeling and analysis of single-robot tasks, and in [7] it is shown how Petri Nets can be used to model a multi-robot coordination algorithm for environment exploration. In [11] Petri Nets' semantics is used for the definition of a formal language for the representation of generic behaviors: Petri Net Plans (PNPs), which are adopted in this work. The graphical approach of this representation framework allows for an intuitive design of complex plans, and multi-robot PNPs are able to represent multi-robot interactions.

Cooperation in multi-robot systems plays an important role, as teamwork can lead to consistent performance improvements. Several RoboCup teams achieve cooperation facilitating interaction through the assignment of individual behaviors, as for instance through the tactical placements of the team members in the soccer field. Some works have studied the possibility of a structured approach to the design of cooperation, for which coordination and synchronization is required. In [9], synchronization through

⋆ This work was partially supported by Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the POS_Conhecimento Program that includes FEDER funds.

L. Iocchi et al. (Eds.): RoboCup 2008, LNAI 5399, pp. 200–211, 2009.

explicit communication is used to attain cooperation on real robots. Implicit communication is used in [6] for the performance of a pass behavior among the members of a team in the RoboCup Simulation League, while in [4] (also in the RoboCup Simulation League) a neural network is employed to learn the conditions associated to the performance of a pass. The RoboCup Virtual RescueRobots League represents another testbed for teamwork, and some interesting applications can be found. In these works, the engagement of cooperation is not usually explicitly modeled, and it is difficult to handle situations, such as action failures, in which the robots have to withdraw the cooperative execution. In [8], the Joint Commitment theory [1] has been used to guide the implementation of cooperative passes through finite state automata. In our work the principles for cooperation outlined by the theory are modeled through Petri Nets, which are provably more expressive than finite state automata.

The Joint Commitment theory provides a detailed formal specification for the design of generic cooperative behaviors. Its prescriptive approach is easily expressed using PNPs, which provide the required level of expressiveness and intuitiveness, maintaining the desired generality to allow the design of a wide range of cooperative tasks. Building upon these characteristics of Petri Net Plans and the Joint Commitment theory, in this paper we present a general model for the representation of multi-robot cooperative behaviors. A robotic-soccer passing task is detailed, as an example in the RoboCup Four Legged League.

not related to the RoboCup competitions.

explicit communication. The guidelines for the design of cooperation are provided by the Joint Commitment theory, which is easily integrated in Petri Net Plans for the implementation of effective and consistent cooperation.

The paper is organized as follows.

Section 2 briefly describes the key elements and operators of Petri Net Plans. Section 3 introduces the Join Commitment theory, showing how it is used as a guideline for the design of multi-robot Petri Net Plans for teamwork. Section 4 describes how this approach has been used in the domain of the technical challenges in the RoboCup Four Legged League.

2 Petri Net Plans

Petri Net Plans [11] are a behavior representation framework that allows the design of highly expressive plans in dynamic, partially observable and unpredictable environments. Note that PNPs do not follow a generative approach, but are a tool for graphical representation of plans. PNPs are based on Petri Nets ([5]), a graphical modeling language for dynamic systems, which is used to represent the many features that are required for behavior modeling, such as non-instantaneous actions, sensing, loops, concurrency, action failures, and action synchronization in a multi-agent context.

The basic structures of a PNP are non-instantaneous ordinary and non-instantaneous sensing actions, shown in Figures 1 and 2.

In an ordinary action two transitions and three places are employed: p_i, p_e and p_o are, respectively, the initial, execution and termination places. A token in p_e represents the execution phase of the non-instantaneous action. The firing of the transitions t_s

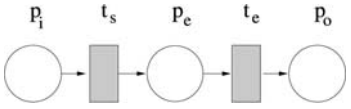

Fig. 1. A non-instantaneous ordinary action

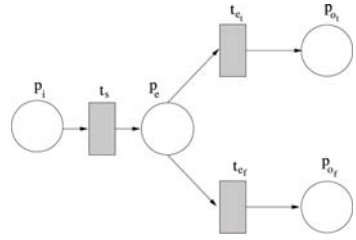

Fig. 2. A non-instantaneous sensing action

and t_e represents, respectively, the starting and the ending of the action. Transitions may be labelled with conditions (typically expressed through a propositional formula) that control their firing. In a sensing action (Figure 2), the ordinary action structure is enriched through an additional transition and a place. Depending on the value of the sensed condition, the corresponding transition is fired (t_{et} is fired if the sensed condition is $true$, t_{ef} is fired otherwise). Ordinary and sensing actions can also be modeled as instantaneous. In this case a single transition is used to represent the start, execution and ending of actions (in the case of a sending action two transitions are used according to the value of the sensed condition). An additional structure, called *no-action*, can be used to connect the structures during the design of a plan. This structure is represented by a single place with no transitions.

In a PNP these elementary structures are combined, through a set of operators, to achieve action sequences, loops, interruption, conditional and parallel execution. These operators are detailed in [11]. t_{interr} becomes $true$ during the execution of the highlighted ordinary action, the token is removed from the execution place p_e, causing the action to be interrupted.

structure consents to concurrently start the execution of multiple actions, while the join operator allows to wait for the termination of multiple actions.

2.1 Sub-plans

In the design of a PNP, sub-plans can be used for a higher modularity and readability. A sub-plan is represented as an ordinary action but refers to a PNP rather than to a primitive behavior.

A plan execution module, running on the robot, takes care of dynamically loading sub-plans in case a super-plan invokes its execution. In particular, whenever a start transition of a subplan is fired, the marking of the subplan is set to the initial one. The subplan will then be executed, possibly concurrently with other primitive behaviors or subplans, until it reaches its goal marking or a condition labeling its ending transition is met. Moreover, subplans allow a more powerful use of interrupts which can be used to inhibit an entire behavior at once. This is a very important feature which will be used, as described in the following, to provide a generic implementation of teamwork through PNPs.

Defined over the end transition in the super-plan are met), and can be interrupted as an ordinary action.

2.2 Multi Robot Plans

Petri Net Plans are also able to represent multi-robot plans [10], through the union of n single robot PNPs enriched with synchronization constraints among the action of different robots. The model we present allows for the design of plans for small teams of robots, such as the ones used in Robocup, and may also be scaled up to medium size teams with an appropriate use of sub-plans. Multi-robot Petri Net Plans are produced in a centralized manner, and then automatically divided, implementing the *centralized planning for distributed plans approach* [3].

Each action of a multi-robot PNP is labeled with the unique ID of the robot that performs it. At execution time each robot divides the multi-robot plan into a single agent plan, for its individual execution. Two operators are used to attain synchronization: the $softsync$ operator and the $hardsync$ operator. Figure 3 shows the structure of the hard sync operator, used to synchronize the execution of two actions.

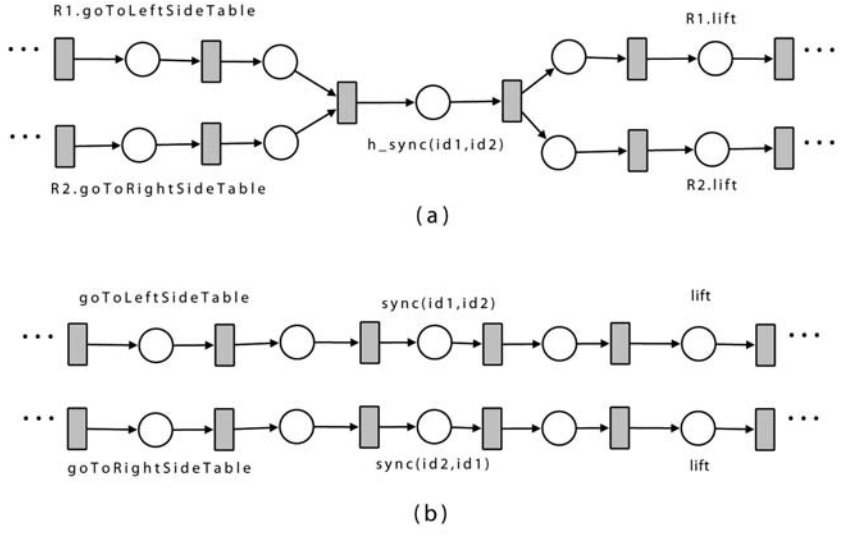

Fig. 3. Hard synchronization operator: (a) multi-robot plan (b) single-robot plans

The hard sync operator relies on the single-robot $sync$ primitive, used to establish a communication link between the two robots to exchange information and synchronize the execution. In the example shown in Figure 3, one robot moves to a side of a table to lift it, while the other robot reaches the other side. The hard sync operator ensures that the table will be lifted only after the robots have successfully terminated their preparation phase.

The soft sync operator provides the possibility to establish a precedence relation among the actions of the individual robots in the multi-robot plan. This structure can be used to allow a robot to asynchronously notify the ending of an action to its partners, starting the execution of another action without waiting for a synchronization with other robots (see [10] for further details).

The case of a multi-robot action interruption is shown in Figure 4.

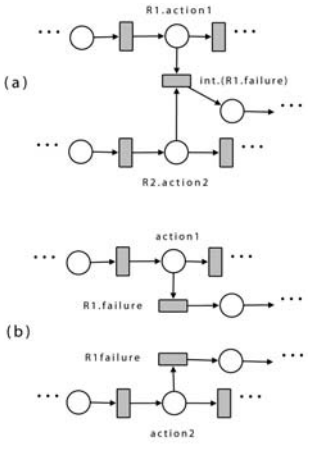

Fig. 4. Multi-robot interrupt operator: (a) multi-robot plan (b) single-robot plans

Single agent communication primitives are again used to communicate the need for an action interruption among different robots. In Figure 4, if the robot $R1$ becomes aware of a $failure$ condition during the execution of $action1$, it notifies the robot $R2$, and the execution of both $action1$ and $action2$ is interrupted.

Petri Net Plans have been used for the implementation of a number of robotic applications. Various videos and complete plans can be found at `http://www.dis.uniroma1.it/~ziparo/pnp`.

3 The Joint Commitment Theory through Petri Net Plans

In [1] P. Cohen and H. Levesque present a formal insight into teamwork, describing the properties that a design of cooperative behavior should satisfy. This section presents a brief overview of these properties, showing how they can be embodied in a PNP and used to implement cooperation.

The Joint Commitment theory isolates a set of basic characteristics that all the cooperating members of a team should share. Too strong and too weak specification of these characteristics are avoided, in order not to set unnecessary constraints on the design, and at the same time to maintain the possibility of a consistent design of cooperative behaviors, given the potential divergence on the mental states of the team members. The theory is rooted in the concept of *commitment*, that is established among the team members that decide to perform teamwork. To summarize, a set of team members that are committed to the execution of a cooperative behavior will continue their individual action execution until one of the following conditions hold:

1. The behavior was concluded successfully
2. The behavior will never be concluded successfully (it is impossible)
3. The behavior became irrelevant

The prescriptive approach of the Joint Commitment (JC) theory can be used to provide a systematic design of cooperative behaviors in a multi-robot team.

3.1 Petri Net Plans for Teamwork

Given the intuitive and expressive behavior programming approach provided by the Petri Net Plans framework, it is easy to embody the specifications provided by the JC theory in the design of multi-robot plans for cooperative tasks. The multi-robot interrupt operator shown in the previous section is used to consistently interrupt the action execution among the different robots that are engaged in a cooperation (being $committed$), in case the behavior becomes irrelevant or fails. The successful conclusion of the individual actions is implemented in the multi-robot plan through a hard-sync operator.

Figure 5 shows a multi-robot Petri Net Plan for the performance of a cooperative behavior, according to the specifications provided by the JC theory.

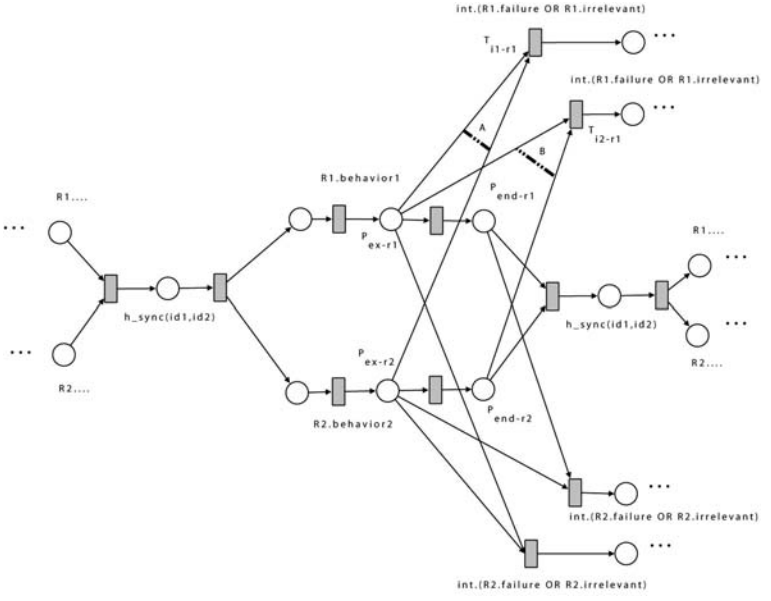

Fig. 5. A Petri Net Plan for a cooperative behavior

After a first synchronization (during which the commitment is established), the two robots start the cooperation, executing their individual behaviors (i.e. $behavior1$ and $behavior2$) which are represented as sub-plans. Following the guideline provided by the JC theory, the commitment is broken if one of the above listed conditions holds. In case one of the engaged robots senses that his behavior became irrelevant or that it has failed, the multi-agent interrupts ensure the event is communicated to the partner, and the execution of the individual actions is interrupted. In the case of successful termination of both $behavior1$ and $behavior2$, a hard sync is performed to successfully end the commitment. It may be possible that one of the two robots successfully terminates the execution of the cooperative behavior while the other is still performing some actions. The possible occurrence of this situation is reflected in the plan shown in Figure 5. If

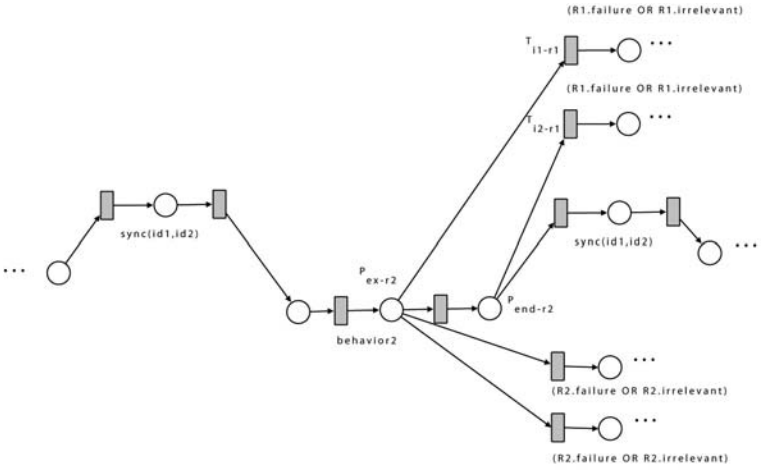

Fig. 6. Single agent plan for the cooperative behavior

the robot $R1$ becomes aware the commitment should be broken while the robot $R2$ has terminated the execution of $behavior2$, the places P_{ex-r1} and P_{end-r2} are marked with a token. Once $Robot1$ has successfully communicated its will to break the commitment, the condition over T_{i1-r1} will be true, and the transition will fire (the edges leading to the transition are highlighted in the figure through the segment A, which is not part of the Petri Net). The transition T_{i2-r1} is instead enabled in the case $R1$ senses the interrupt condition to be true while $R2$ is still executing $behavior2$. In this case the places P_{ex-r1} and P_{ex-r2} are marked with a token, and once $R1$ has successfully notified the interruption to $R2$, the condition over T_{i2-r1} will be true.

Executing $behavior2$, the transition T_{i1-r1} will be $enabled$ (the places P_{ex-r1} and P_{ex-r2} are marked with a token and the condition over T_{i1-r1} is $true$). In the case that $Robot2$ has already finished the execution of $behavior2$, and is currently waiting for a synchronization from $Robot1$, to interrupt the commitment, the transition T_{i2-r1} will be $enabled$ (the places P_{ex-r1} and P_{end-r2} are marked with a token and the condition over T_{i2-r1} is $true$). The arcs involved in the firing of the interrupt transitions in the figure have been linked through the segment A (for the transition T_{i1-r1}) and B (transition T_{i2-r1}).

The structure of this Petri Net Plan prevents a deadlock to occur in the described situation. In Figure 6 the single agent plan for the robot $R2$ is shown.

Only one of the two interrupt transitions (T_{i1-r1} and T_{i2-r1}) connected to the execution of the behavior $behavior2$ can fire during the execution, as the other robot will only handle one of the two possible multi-robot interrupt communications. The communication required to evaluate the interrupt conditions in this Petri Net is implemented through the underlying communication layer.

3.2 Applications

Teamwork is very beneficial, if not unavoidable, in many robotic applications. The structure shown in the PNP of Figure 5 can be used as a model for a wide range of

cooperative tasks that require the establishment of an explicit $commitment$ among robots.

As an example, in the RoboCup Rescue domain, consider a mini UGV and a mini UAV proceeding in formation during the exploration of a terrain. The two vehicles are $committed$ to the cooperative exploration. While committed, the mini UAV and the mini UGV perform coordinated individual behaviors for the exploration. The formation is in this case a necessary condition for the success of the cooperation. In case, for some reason, the formation is broken (e.g. the mini UGV looses visual contact with the mini UAV), the commitment is broken (through a communication action), and the cooperative exploration is interrupted. This interruption leads to the execution of individual behaviors that will allow the reestablishment of the formation (e.g. the mini UAV performs a behavior to facilitate its detection, while the mini UGV seeks its partner). The described behaviors can be easily represented in the PNP framework, making use of the structure of Figure 5 to handle the commitment of the two vehicles.

Explicit cooperation for the execution of complex tasks may be required in the RoboCup Soccer scenario as well. Consider the example of a pass between two robotic-soccer players. If the conditions for a pass hold, a commitment is established. The robots will need to agree on the allocation of the required tasks (pass and receive the ball). Suppose the passer robot looses the ball (e.g., an adversary robot intercepts it before the pass can take place): the failure of the pass needs to be communicated to the receiving robot, which is meantime preparing to receive the pass, and the execution of the individual cooperative behaviors needs to be interrupted. This example has been implemented in the RoboCup Four Legged League scenario, and will be detailed in the next section.

4 An Example in the Robotic-Soccer Environment

In the past editions of the RoboCup competitions the development of cooperative behaviors has been encouraged. The Passing Challenge, proposed in 2006 (Bremen, Germany) and 2007 (Atlanta, USA) in the Four Legged League, directly addresses the problem of cooperation. In this technical challenge the robots are placed in three spots on the soccer field with the task of passing a ball. Passing the ball to a robot which was not engaged in the last pass has a higher score reward, and a pass is considered valid if the robot intercepts the ball within a certain distance from its assigned position. In this paper, we describe a passing task with a more specific focus on cooperation, exogenous events and dynamic assignment of tasks, ignoring the aspects related to the localization of the robots in the field.

The implementation of this task requires (besides the development of basic functions such as vision, locomotion and primitive behviors) synchronization and coordination. The implemented system has been presented at the seventh international conference on Autonomous Agents and Multi-Agent Systems (AAMAS08), and received the “Best Robotic Demo Award”. The multi-robot PNPs written for this task, as shown below, reflect the principles of the JC theory.

The assignment of the roles for the pass behavior is performed in the multi-robot PNP at the first stage of the task execution: two of the three robots select the roles of $Passer$ and $Receiver$, according to the position of the ball in the field and the previously performed passes (a robot that recently passed the ball has a lower probability of

Fig. 7. Two robots passing the ball during the passing task

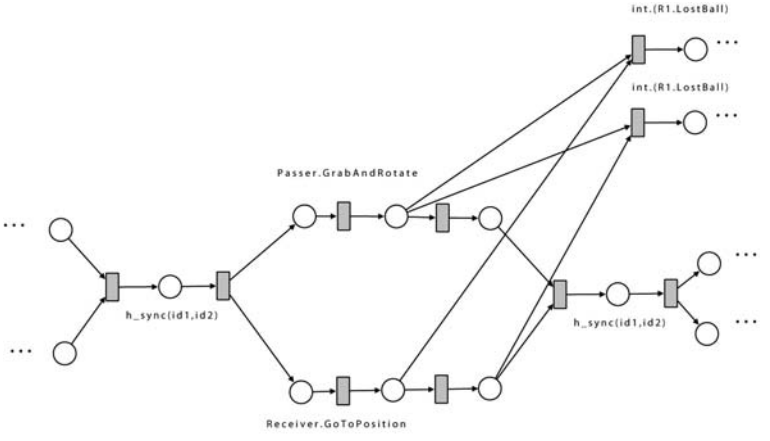

Fig. 8. Preparation phase of the pass behavior

being assigned with the role of $Receiver$). To allow role assignment the robots initially exchange their local information on the distance from the ball, and synchronize for a consistent allocation of roles. For more details on the assignment of the cooperative roles see [10].

The synchronized execution of actions required by the passing task is implemented through Petri Net Plans, which embody, as shown in the previous sections, the guidelines provided by the Joint Commitment Theory. A first synchronization is used to commit the robots to the execution of the pass. The hard synchronization operator is used for this purpose. The robot that has been assigned with the $Passer$ role reaches the ball, grabs it and rotates towards its partner. Meanwhile the $Receiver$ robot reaches the desired position and prepares to intercept the passed ball, rotating towards the $Passer$. At the end of this phase, the robots renew their commitment through another synchronization. The hard sync operator is again used to ensure both the robots have completed their task before they can proceed with the pass. This preparation phase is prone to action failures, due to the difficulty of implementing reliable grab and rotation primitives with AIBO robots, and due to possible occurrence of exogenous events (e.g. collisions with other robots) that may interfere with the predicted performance of the primitives. Reflecting the principles of the JC theory, the robots break their commitment if and

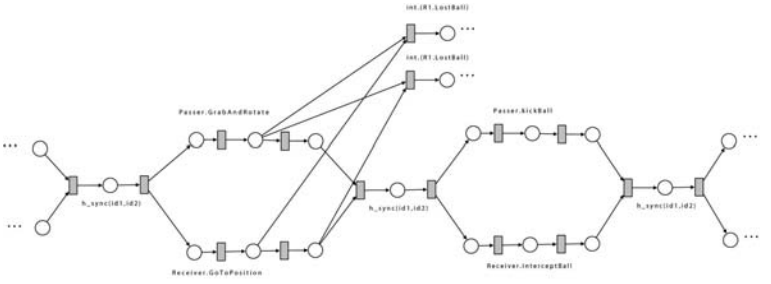

Fig. 9. Multi-robot Petri Net Plan for the pass behavior

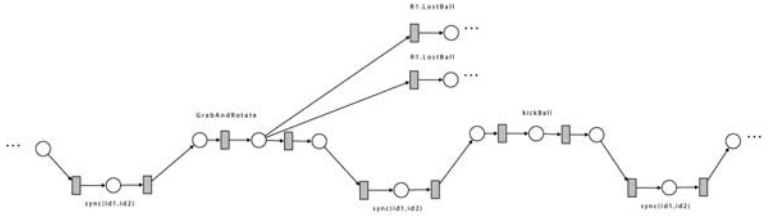

Fig. 10. Single-robot Petri Net Plan for the pass behavior: Passer

when a failure occurs during this phase (in this particular task the cooperative behavior is never considered irrelevant, as the robots have the unique task of passing the ball).

Figure 8 shows the Petri Net Plan for the execution of this first part of the task. The $LostBall$ condition becomes $true$ in case the $Passer$ robot realizes that the ball has been lost during the grab or the rotation phases. The ball may in fact roll away from the robot, causing the need for a new task assignment procedure. If control of the ball is lost by the $Passer$ robot, the $Receiver$ robot needs to be notified, in order to break its commitment to the current execution of the pass. A multi-robot interrupt operator is used to consistently interrupt the execution of the actions of both the $Passer$ and the $Receiver$.

If the commitment is successfully maintained the pass can take place. The $Passer$ robot kicks the ball towards the receiver, which in the meantime performs an intercept behavior. This phase does not require particular attention for action interruption, as the kick and the intercept behaviors are atomically performed and the pass behavior is concluded both in the case of success and in the case of failure of the pass. A further synchronization (through a hard sync operator) is performed to exchange information about the outcome of the behavior, and the commitment is broken. The final multi-robot plan for the pass behavior is shown in Figure 9, while Figure 10 shows the single agent plan executed by the $Passer$ robot.

Necessary high level programming. Using the modularity that Petri Net Plans offer through the possibility of using sub-plans, it is possible to implement the multi-robot behavior for the pass, that requires synchronization among the robots, and represents

the core of the behavior definition for this task. The plan for the two robots performing the pass was shown in Figure 3.

5 Conclusions

The use of Petri Net Plans for the representation and execution of robotic behaviors has proven very effective. Besides the formal characteristics of the framework, and its intuitive graphical interface, an appealing characteristic of PNPs is the systematic approach that has been provided for the implementation of single and multi-robot behaviors. In this work we have introduced a general model for the design of cooperation through PNPs, building upon the multi-robot synchronization operators, aided by the specifications provided by the Joint Commitment Theory. To illustrate the effectiveness of the proposed model we have detailed the design of a robotic-soccer task, but the same approach may be applied to a wide range of cooperative behaviors.

To achieve teamwork, communication is required. In the presented work we have assumed the existence of a reliable communication channel. However, the appropriate use of behavior interruptions, not shown for simplicity in the presented PNPs, allows the handling of noisy communications as well.

As a future work towards a structured definition of cooperation in the RoboCup domain, we are working on the integration of cooperative behaviors in the soccer competitions. To this extent, we are currently developing a system for the establishment of commitments among the team members during the soccer games, using a task assignment algorithm based on utility functions.

References

1. Cohen, P.R., Levesque, H.J.: Teamwork. Special Issue on Cognitive Science and Artificial Intelligence 25, 486–512 (1991)
2. Costelha, H., Lima, P.: Modelling, analysis and execution of robotic tasks using petri nets. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007, October 29 - November 2, vol. 2, pp. 1449–1454 (2007)
3. Durfee, E.H.: Distributed Problem Solving and Planning. In: Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (2000)
4. Matsubara, H., Noda, I., Hiraki, K.: Learning of cooperative actions in multiagent systems: A case study of pass play in soccer. In: Sen, S. (ed.) Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems, Stanford University, CA, pp. 63–67 (1996)
5. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
6. Pagello, E., D'Angelo, A., Montesello, F., Garelli, F., Ferrari, C.: Cooperative behaviors in multi-robot systems through implicit communication. Robotics and Autonomous Systems 29(1), 65–77 (1999)
7. Sheng, W., Yang, Q.: Peer-to-peer multi-robot coordination algorithms: petri net based analysis and design. In: Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, July 24-28, pp. 1407–1412 (2005)

8. van der Vecht, B., Lima, P.U.: Formulation and implementation of relational behaviours for multi-robot cooperative systems. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS, vol. 3276, pp. 516–523. Springer, Heidelberg (2005)
9. Yokota, K., Ozaki, K., Watanabe, N., Matsumoto, A., Koyama, D., Ishikawa, T., Kawabata, K., Kaetsu, H., Asama, H.: Uttori united: Cooperative team play based on communication. In: Asada, M., Kitano, H. (eds.) RoboCup 1998. LNCS, vol. 1604, pp. 479–484. Springer, Heidelberg (1999)
10. Ziparo, V., Iocchi, L., Nardi, D., Palamara, P., Costelha, H.: Pnp: A formal model for representation and execution of multi-robot plans. In: Padgham, M., Parkes, Parsons (eds.) Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, pp. 79–86. IFAAMAS Press (May 2008)
11. Ziparo, V.A., Iocchi, L.: Petri net plans. In: Proceedings of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA), Turku, Finland, Bericht 272, FBI-HH-B-272/06, pp. 267–290 (2006)