Individual and Cooperative Behaviors Representation Based on Petri Nets
(December 2010)

Nuno Rodrigues and Pedro Lima

Abstract—This work focuses on developing, testing and analyzing individual and relational behaviors between autonomous robots, in a robotic soccer setting. A new methodology was developed to model relational behaviors, that can be used to perform a dynamic pass between two robots but also has the potential of being used for any other type of relational behavior between several agents. Two different passes were modeled, short and long, that are used in different game situations. Every behavior developed can be composed with models of the actions and the environment in order to obtain a single model in which a quantitative analysis can be done by using Markov Chains. Environment model uses not only controllable events by our agents but also uncontrollable events that represent the world physics and/or the impact of other agents in the world. This is an abstraction of the surrounding world but by analyzing the performance important knowledge can be obtained a priori.

Keywords — Petri Nets, Robotic Soccer, Individual Behavior, Relational Behaviors, Quantitative Analysis, Commitment Establishment

I. INTRODUCTION

This work follows the direction of [2], [3] and [4], nevertheless, there was a need to create new behaviors in order for the robots to perform accordingly to each game situation. Some weren’t yet implemented, such as, having a good defensive behavior when the game is in a foul situation, be it for us or the other team, having a behavior for dropped ball and searching for the ball during a foul situation or during the game play. Other individual behaviors needed to be optimized and changed to allow a bigger rate of success, when performed by the real robots, and to keep up with the changes in the rules.

In regards to cooperative behaviors, the method used to perform a dynamic pass needed to be completely redone in order to be more effective and have the potential to be used in more situations than the ones described in this work. Also the old conditions used to perform a dynamic pass were very restrictive and as a result the pass was rarely performed and when it was, the success rate would be low. This was even more of a problem because when a dropped ball situation occurred the robot had to perform a pass before scoring a goal. This called for the need of modeling a simple pass that in a dropped ball situation could be performed with the maximum security, by taking in account the teams limitation, and another pass that could be more flashy and be performed when a more restrictive set of conditions presented themselves.

Another goal was to be able, using the framework in [8], to perform an a priori analysis of the behaviors by composing it with environment and action models. Two types of analysis were performed, one with the objective of finding in a behavior which action has the biggest impacts on the overall goal of the behavior and a second one which has the objective of ascertain which pass has the higher probability of being successful.

II. PETRI NETS

A. Ordinary Petri Nets

In [5] a formal definition of Petri Nets can be found. A Petri Net is a 5-tuple, \( PN = (P, T, A, w, M_0) \), where:

\( P = (p_1, p_2, \ldots, p_n) \) - is a finite set of places;
\( T = (t_1, t_2, \ldots, t_m) \) - is a finite set of transitions;
\( A \subset (P \times T) \cup (T \times P) \) - is a set of arcs (flow relation);
\( w : A \rightarrow N \) - is a weight function;
\( M_0 \) - is the initial marking.

It is also explained the firing rule [8] which allows the Petri Net to evolve. This rule is composed of the following three consecutive steps:

1. A transition is said to be enabled if each input place is marked with at least \( w(p, t) \) tokens, where \( w(p, t) \) is the weight of the arc from the place to the transition. When \( w(p, t) \) isn’t specified, its value is 1.
2. An enabled transition may or may not fire (depending on whether or not the event actually takes place).
3. The firing of an enabled transition removes \( w(p, t) \) tokens from each input place, and adds \( w(t, p) \) tokens to each output place, where \( w(t, p) \) is the weight of the arc from the transition to the place. When \( w(t, p) \) isn’t specified, its value is 1.

B. Generalized Stochastic Petri Nets

In [6] the concept of Generalized Stochastic Petri Nets is introduced. The main difference is the introduction of stochastic (exponential) transitions that, once enabled, only
fire after an exponentially distributed time $d_j$ has elapsed.

A GSPT is an eight-tuple $(P, T, A, w, M_0, R, S)$, where:

- $P = \{p_1, p_2, \ldots, p_n\}$ - represents a finite set of places;
- $T = (T_E, T_I)$ - represents a finite set of transitions, that can be immediate transition ($T_I$) or exponential transitions ($T_E$);
- $A \subseteq (P \times T) \cup (T \times P)$ - represents the set of arc connections from place to transition and transition to place respectively;
- $w: A \rightarrow N$ - weight function associated with each arc;
- $M_0$ - initial marking of the PN places;
- $R$ - function from the set of transitions $T_E$ to the set of real numbers, $R(t_{ij}) = \mu_j$, where $\mu_j$ is called the firing rate of $t_{ij}$;
- $S$ - set of Random Switches which associates probability distributions to subsets of conflicting immediate transitions. These random switches can be static (invariant to the marking of the net) or dynamic (dependent on the marking of the net).

The GSPT marking is a Markov process with a discrete state space given by the reachability graph of the net for an initial marking. This fact allows the performance of a qualitative analysis of the behavior developed.

III. PETRI NET MODELS OF ROBOTIC TASKS

In [7] it is introduced a way to model a robotic task, using Petri Nets, by dividing the task model in layers with different degrees of abstraction. In the SocRob project there are three layers of abstraction:

- Team Organizer – Selects the team’s tactic by deciding which role the robot should execute, role Attacker, Supporter or Defender. Highest level of abstraction;
- Behavior Coordinator – Organizational decisions are performed by choosing which behavior is selected depending on the game situation;
- Behavior Executor - Where are executed the behaviors. Every behavior is a Petri Net of primitive actions, that are the most basic element of the robotic task. Lowest level of abstraction.

In the Petri Net framework a place has different meanings depending on the prefix used, such as:

- Action - When a place with this prefix is marked, it means that a role, behavior or primitive action is being executed, depending on the layer. For example, if the place is in TeamOrganizer level, a role is being executed, if the place is in the BehaviorCoordinator level, a behavior is being executed and finally if the place is in the BehaviorExecutor level, an action is being executed;
- Predicates – Performs an evaluation on the world and, if it is true, the place will have a token but, if it is false, the negation of that predicate will be marked (e.g. if the robot sees the ball predicate SeeBall will have a token but if the robot doesn’t see the ball predicate NOT SeeBall will have a token). Since these places are marked only through an evaluation, all outputs transitions are also input transitions;
- Macros – They are used to reduce the complexity and increase the modularity of the Petri Nets, since they represent a Petri Net inside of another. Every macro can end in a successful or unsuccessful status;
- No prefix – Internal memories of the nets.

Each transition fires, as well, according to its prefix:

- $T\#$ - This transition fires the moment it is enabled;
- Event - This transition fires only if the event associated occurs. Events are mainly sent by the robots;
- GOALReached - this transition fires if the associated macro reached a successful status;
- GOALNotReached - this transition fires if the associated macro reached an unsuccessful status.

IV. ENVIRONMENT AND ACTION MODELS

In [8] a framework, to model the environment and the actions, is given.

The environment models, represent, for instance, state changes that might occur due to actions performed by other robots/agents or even physics. A discrete set of relevant states is abstracted from the actual environment and as a result, these models will not fully model the environment, but an abstraction of it.

The environment abstraction is achieved by discretising the world using logic predicates. As such, environment models consist of GSPNs with predicate places.

An action is mainly described by the effects it causes on the environment and the conditions that need to be met for the effects to take place. In logical terms, the action properties can be partitioned in the following sets:

- Running-conditions: Conditions that need to be met for the action to be able to produce changes in the world;
- Effects: Composed of Success Effects and Failure Effects, reflects the impact an action has on the world;
- Success-effects: These are the effects associated with the success of the action. These include the desired-effects of the action plus additionally intermediate effects that might occur in order to achieve success;
- Failure-effects: These effects are the undesired ones, which might happen as a direct result of running the given action.

V. COMPOSING

A major advantage of Petri nets is their modularity. The various models that constitute the entire model can be adjusted separately and combined only when needed, for example, when trying to perform the analysis of the full model. In [8] a in-depth explanation can be found on the method of composing a behavior that runs in the robot with the environment and action models in order to obtain a single Petri Net in which an analysis can be performed, be it qualitative or
quantitative, allowing for an a priori knowledge of how the behavior will behave.

VI. COMMITMENT

A relational behavior occurs when two or more robots act in cooperation during the execution of a behavior, in order, to achieve a common goal. A robot knows if the other is still executing the expected behavior, if the other keeps on sending messages that he is committed.

In this work a commitment will be established in a request/accept method. A robot sends a request to the others in order to find a partner to perform a relational behavior with, and if he receives answers, he will choose which one of them is the best candidate to perform a commitment. Note that a commitment should be maintained while the goal isn’t reached by any of the robots, and only if it at some point doesn’t become unreachable.

The moment the goal is reached, or the moment it is concluded that the goal can’t be reached the robots must end the commitment. When a robot loses communicating with the others, they must assume the commitment was broken.

VII. DEVELOPED BEHAVIORS

The number of behaviors modeled in this work doesn’t allow for an in-depth description of each one but they can be found in [9]. Therefore only the relational behaviors will be described bellow.

The two passes created were the short pass and the long pass.

The short pass is initialized by the attacker and is performed when a dropped ball situation occurs in the game since the rules stipulate that after a dropped ball it is mandatory to perform a pass. In this pass, the receiver moves until it is right in front of the taker in which time the taker gets out of the way, leaving the ball behind for the receiver. Since the ball is stationary the pass has a bigger probability of success.

The long pass performs a spontaneous dynamic pass during the game when the conditions for it are met (attacker has the ball and has obstacles in front of the goal and supporter is in a good position to receive the pass). The taker turns towards the receiver and passes the ball. Since the ball is moving, the probability of success is low.

A. BehaviorCoordinator Level

In the coordinator level the block represented in Figure 1 can be added in any role but since the objective is to perform a dynamic pass, this block will be added in the role attacker and the role defender.

From Figure 1 we can ascertain that while the robot is executing the individual behavior, BehaviorBaseAttacker for the attacker and BehaviorBaseSupport for the supporter, the procedure needed to perform a dynamic pass is running in parallel. The memory place DynamicPassControl guaranties that the attempt to perform a dynamic pass is only made after a previous attempt ended.

In order for the robot to attempt to perform a commitment it will have to receive a request from another robot, predicate DynamicPassRequest TRUE, or the conditions for the robot to initialize a pass with another robot are met, predicate DynamicPassConditionsOK TRUE. Note that the condition for a given pass changes depending on which one we want to perform.

If the commitment is not a success, the memory place DynamicPassControl gets a token again and the cycle begins anew. On the other hand, if the commitment is a success, the robot stops performing the individual behavior and starts executing the relational behavior, which will only stop when the pass was a success, predicate DynamicPassEnded TRUE, or if the pass had to be interrupted, StopDynamicPass TRUE (e.g. game was stopped, the time allotted for the pass expired or if the conditions to perform the pass are no longer met).

Finally, macro CommitmentBreakingBC is executed in order to break the commitment.

The macro used for establishing a commitment has the procedure to request a commitment and to accept a commitment because each robot can find himself in a situation where they can be the one requesting a commitment or the one accepting a commitment. Note that when the commitment is established an action will be executed in order to inform all the robots which pass is being performed because the PNs executed by the taker and the receiver are the same and what will determinate what actions will be performed are the predicate DoingShortPass and predicate DoingLongPass.

B. BehaviorExecutor Level

BehaviorDynamicPassReceiver

When this behavior is selected, the robot knows that it will receive a dynamic pass, therefore, the first action executed is Move2DynamicPassReceive. This position is different if the robot is performing a short pass or a long pass, nevertheless, as soon as this position is achieved, action SendReady2Receive is performed in order for the partner to know that he can pass the ball. First point of synchronism.
When the predicate PassDone is TRUE, second point of synchronism, the robot moves towards the ball with action Move2Ball and when it is close to it, predicate CloseToBall TRUE, action CatchBall is executed. Finally, when the robot catches the ball action SendDynamicPassEnded is performed in order to inform all the robots that the pass ended, third point of synchronism.

BehaviorDynamicPassTaker

When this behavior is selected the robot knows that it will perform a pass so the first action executed is TurnAroundBallToPartner since it is the job of the receiver to go to the receiving position. If the robot is turned to the partner, it executes action Stop and waits for confirmation that the partner is ready to receive, predicate PartnerReady2Receive TRUE, however, while waiting the robot might lose the ball so actions Move2Ball and Catch ball are also included in the plan. When predicate PartnerReady2Receive is TRUE, first synchronism point, the robot kicks the ball and then sends pass done if it is a long pass or it will execute MoveAwayFromBall and then send pass done if it is a short pass.

VIII. ANALYSIS

A. Analyzing which action has more impact in the plan, Case 1

In this analysis, the goal is to have a way to determinate which primitive action has a bigger impact in the overall goal of one behavior. For that, the behavior of Figure 2 was created with the overall objective is to score a goal.

An initial transient analysis is performed to determinate the probability of a goal being scored. Then, each action, InterceptBall and Dribble2Goal, will be improved and two different analyses, to ascertain which one has a bigger impact, will be performed. The stochastic transitions are arbitrary but not unrealistic.

All models used in the three analysis are the same, only some of the stochastic transitions will have their value changed in order to model the improvement of each primitive action.

In order to perform a transient analysis, the behavior has to be composed with the environment models and with the action models. Figure 3 thru 6 are environment models and Figure 7 thru 10 are action models.

Since the objective in only to ascertain the impact of InterceptBall and Dribble2Goal the environment models don’t have to be very complex, only needing to take in account the aspects that represent the success of each primitive action in order for them to be improved.

BallPosition: The field is divided in five positions and the ball can travel between them because of something other than the robot, but only when he doesn’t have the ball. Note that when the ball reaches BallOppGoal or BallOwnBall a goal was scored and the ball can’t exit this position.

Robot position: It is consider that the robot always knows its position so no stochastic transitions are needed.

HasBall: At any given time the robot can loose the ball and this is what this model represents. Note that when improving the Dribble2Goal the rate of T1 should be decreased.

ShootOpportunity: After action Dribble2Score finds a good spot to shoot the ball, this spot can be lost before the robot has the chance to kick.
performed by each robot during the dynamic pass. In order to create an environment, in which the short pass and dynamic pass could be tested, the field was divided in six different positions. This led to a huge increase of complexity in both the environment and action models. They can be consulted entirely in [9].

Here, several considerations were taken in account:

- Only the relational behaviors were used in the analysis. The individual behaviors that the robots are executing before the relational behavior were not modeled;
- The part of the commitment is not important for this analysis so after the macros there are stochastic transitions that represent the time that the macros spent running;
- Only the ball related models and communications models are the same for both robots. All the other have to be modeled for each robot, e.g. R1HasBall and R2HasBall;
- Communications used for the synchronism between the receiver and the taker are done by a pair action/predicate. The only thing that the actions do is change the value of each predicate pair associated to TRUE. On the other hand, the only thing that predicate models do is to change the predicates values back to False, after a small amount of time has elapsed;
- Since the same models are used for both types of pass, the rate of the transitions can be arbitrary as long as they aren’t unrealistic;
- All the other action models are modeled in order for the desired effect to occur.

In order to test the short pass, the composed PN will have the following predicates with a token:

\[\text{BallX2Y1, R1X2Y2, R2X1Y3, DoingShortPass, DynamicConditionsMet, GameIsStarted, HasBall, R1HasBall, R1CloseToBall, SeeBall, BallStopped.}\]

In order to test the long pass, the composed PN will have the following predicates with a token:

\[\text{BallX1Y1, R1X1Y1, R2X1Y3, DoingLongPass, DynamicConditionsMet, GameIsStarted, HasBall, R1HasBall, R1CloseToBall, SeeBall, BallStopped.}\]

Note that if the predicate was not mention in the lists above, the negation of the predicate is the one that has a token.

\[\text{IX. Results}\]

In Case 1 the goal was to see if by improving one primate action at a time it would be possible for us to identify the one that contributes more to the overall goal of the behavior. This is a transient analysis and what is measured is the probability of having a token in predicate BallOppGoal (goal for us).

In Figure 11 and 12 the results for the three situations are shown. In blue we see the probability of scoring a goal with the base values for the transitions. In green we see the results for when the InterceptBall is improved by around 50% and finally in magenta we see the results of when the Dribble to
goal is improved 50%.

Figure 11: Result for Case 1 when a big amount of time passes

Figure 12: Result for Case 1 when a small amount of time passes.

From Figure 11 we can see that improving the primitive InterceptBall, the probability of scoring a goal in the opponent goal is bigger than when action Dribble2Goal is improved. With Figure 12 we see that this probability increases much faster when the InterceptBall is improved.

In Case 2 the goal was to perform a direct comparison between the short pass and long pass. For that a transient analysis was done to determinate the probability of having a token in predicate R2HasBall which indicates that the pass was a success.

Figure 13: probability for the receiver to catch the ball in a short and long pass.

From Figure 13 one can easily see that the short pass has a bigger probably of success than the long pass.

X. CONCLUSION

Using Petri Nets as a modeling tool proved to be a good choice because of their modularity. Also, Markov Chains can be used to analyze quantitatively the behaviors by composing them with the environment and action models. The theoretical analyses performed on the Petri nets allowed for important information to be obtained. With Case 1 we see that improving the action InterceptBall improves the behavior a bit more than when Dribble2Goal is improved. An evaluation like this can be performed for any behavior with the objective of detecting the primitive action that is more critical for the overall objective.

With Case 2 we can conclude that the short pass is safer that the long pass therefore this should be the pass used when the dynamic pass is mandatory.

All the behaviors were tested in the simulator Webots and the real robots. Videos can be found in http://socrob.isr.ist.utl.pt/videos/behaviors/behaviors2011.html

ACKNOWLEDGMENT

The behaviors were developed and tested in the Middle Size League soccer robot team of the SocRob project (Society of Robots or Soccer Robots), in ISR/IST (Instituto de Sistemas e Robótica / Instituto Superior Técnico).

REFERENCES