Original source of publication:
Computer Safety, Reliability, and Security - SAFECOMP 2015 Workshops
F. Koornneef and C. van Gulijk (Eds.)
Lecture Notes in Computer Science, Volume 9338

Reconfiguration Testing for Cooperative Autonomous Agents

Francesca Saglietti, Stefan Winzinger, Raimar Lill

University of Erlangen-Nuremberg
Lehrstuhl für Software Engineering (Informatik 11)
Martensstr. 3, 91058 Erlangen, Germany
{francesca.saglietti, stefan.winzinger, raimar.lill}@fau.de

Abstract. In order to verify reconfiguration of interacting autonomous agents to be exclusively beneficial and never hazardous to cyber-physical systems, this article suggests a systematic approach based on incremental model-based testing and illustrates its application to cooperating mobile robots.

Keywords. Cyber-physical systems, robots, autonomous agents, cooperation, reconfiguration, CPN modelling, incremental testing.

1 Introduction

Cyber-physical systems increasingly tend to rely on the pro-active cooperative behaviour – or at least on the safe co-existence – of autonomous systems, e.g. mobile robotic agents, singularly developed and validated beforehand. The major purpose of systems-of-systems resulting from aggregation and cooperation of individual agents in a common environment is the provision of higher service performance or efficiency than can be expected from the mere union of their parts. In particular, such improvement may involve autonomous decision-making on proper counteractions to be activated upon detection of anomalous operational conditions by means of suitable reconfiguration strategies.

Evidently, the additional behaviour emerging from interaction of cooperating agents must be verified to be exclusively beneficial and never hazardous to the cyber-controlled physical world, especially in case of safety-relevant applications. For this purpose, the present article suggests a systematic approach to incremental testing of reconfiguration behaviour for cooperating mobile robots by defining objective metrics of structural coverage to be successively fulfilled by automatic test case generation.

The article is structured as follows: after this introduction, the stages characterizing individual and cooperative robot activities are presented and analysed in terms of potential fault and failure modes (section 2). Successively, these stages are further considered in the light of reconfiguration strategies (section 3). For the purpose of modelling and verifying increasing levels of reconfiguration, the selection of Coloured Petri Nets is justified in section 4 and its application illustrated in section 5 by means of an example inspired by a hospital logistics system involving cooperative trolleys. Finally, section 6 proposes a novel structural testing concept based on the

incremental generation of test case sets targeted at covering all consecutive state pairs of corresponding CPN models capturing increasing degrees of reconfiguration.

2 Processing Stages

2.1 Individual Behaviour

The typical processing scheme of a robot deployed in a given environment is structured along the following successive stages.

- **Sensing.** Measurement of raw data by means of appropriate (e.g. electric, electromagnetic or optical) sensors [1] concerning contextual information about environmental conditions (e.g. distance to next obstacle, external temperature, brightness, GPS-coordinates) or robot attributes (e.g. energy, internal temperature, speed).
- **Perception**. Interpretation of sensed data for the purpose of gaining insight about properties of the real world surrounding the perceiving robot such as to allow to represent it by means of environmental models providing a solid knowledge base for further decision-making, e.g. by visual pattern recognition algorithms for object identification.
- **Reasoning.** Application of decision-making algorithms based on pre-defined logic rules to the current perception of reality for the purpose of determining how to proceed, i.e. which actions to instantiate next, typically involving trade-off optimization w. r. t. alternative options.
- **Action**. Execution of action(s) previously identified during decision-making, where super-ordinate actions, e.g. *target next charging station*, may involve further sensing, perception and reasoning activities concerning corresponding sub-ordinate actions, e.g. in the above case the sub-actions *find shortest path*, *move*, *recharge* and *resume original mission*.

2.2 Cooperative Behaviour

In case each robot disposes of own local sensors, the initial sensing phase may be usually assumed to rely on exclusively individual behaviour of interacting agents, while all later processing stages (s. Fig. 1) may involve cooperative behaviour:

- **Perception-based Cooperation.** Fusion of data sensed by multiple robots for the purpose of extracting more information than achievable by the sum of its parts, e.g. robots gathering visual information from different view angles such as to obtain consistent stereo-vision.
- **Reasoning-based Cooperation.** Coherent decision-making of robots working in a common environmental context, such that the global behaviour emerging from individual actions is beneficial to cooperation. For example, facing robots aiming at switching their position must avoid symmetrical evasive manoeuvre in order to exclude mutual blockades.

- **Action-based Cooperation.** Efficient and effective coordination of individual robot actions in order to achieve a common task. For example, the cooperative lifting of a heavy load requires synchronicity of movement and balanced application of forces in order to avoid tilting effects.

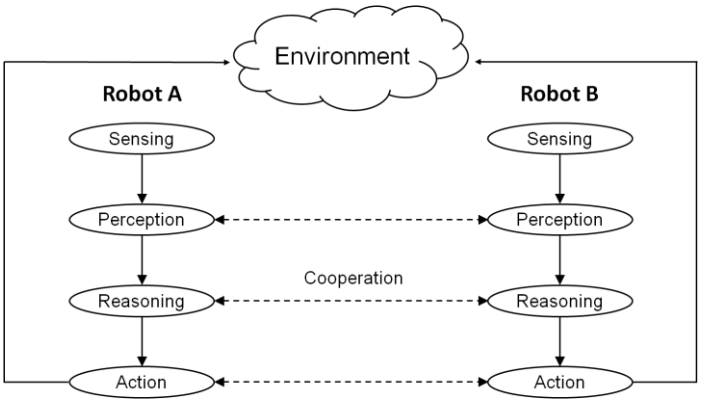

Fig. 1. Individual and cooperative processing stages

Evidently, each of these stages may be affected by specific faults or inaccuracies potentially jeopardizing the performance of the corresponding activity as well as that of later stages relying on it (s. Table 1). As sensing, perception and action represent classical challenges in robot design and construction which may rely on long research and industrial experience, in the following they are assumed to have been trained and verified before deployment with respect to a wide range of target environments.

On the other hand, the real challenges arising after robotic construction are felt to be especially related to the verification of *reasoning*, *cooperation* and *reconfiguration* tasks which depend on plant-specific design concepts and therefore must (and can only) be explicitly addressed in later testing phases.

While systematic testing of cooperative behaviour based on autonomous reasoning has been the subject of investigations on which we reported in the past [2], [3], the present article poses the focus of its considerations on the particular issue of testing *reconfiguration* of autonomous cooperative agents.

3 On-line Fault and Anomaly Handling

3.1 Fault Detection and Fault Tolerance

In case the acceptable system behaviour can be specified in advance in a unique and precise way, classical fault tolerance approaches may be applied to enforce this behaviour (possibly allowing for some degree of degradation) even in the sporadic occurrence of component failures.

This can be achieved by activating during runtime appropriate counter-measures based on different redundancy classes:

- **fault masking**, where a majority is selected by comparison checking (voting) among *structurally redundant* components;
- **error recovery**, where a predefined acceptable state is (re-)established upon detection of an error by a *functionally redundant* component (acceptance test);
- **error correction**, where errors resulting from transmission or storage failures are corrected thanks to the *information redundancy* provided by additional data.

Table 1. Fault classes and failure modes

Stage	Fault Classes / Failure modes	
	Individual behaviour	**Cooperative behaviour**
Sensing	flawed sensor(s) resulting in - delivery of distorted data, - data delivery outside required time slots, - (partial) omission of data delivery.	----
Perception	inaccurate interpretation of sensed data by - inaccuracy of environmental model, - lack of significant sensed data, - inappropriate perception algorithm.	incorrect representation by inconsistent data fusion
Reasoning	incorrect decision-making due to - inadequate specification, - flawed design, - incorrect implementation.	conflicting decisions by non-concerted reasoning
Action	incapability of completing action due to - inappropriate use of instruments, - flawed instruments, - lack of energy resources, - environmental constraints.	inefficient / ineffective / unsafe behaviour by uncoordinated actions

3.2 Anomaly Detection and Reconfiguration

In general, the behavioural multiplicity of robotic applications, induced by varying missions and operational conditions, does not allow for the determination of one single target behaviour. Upon detection of anomalies preventing them from carrying out a standard functionality, robots must rather evaluate alternative procedures and select one of them on the basis of their current perception by reconfiguration, i.e. by adjusting their future physical and/or logical activities. Reconfiguration, therefore, goes beyond fault tolerance by permitting to adapt the behaviour to anomalous operational conditions; it includes the following classes:

- **Adjusted Sensing.** Upon detecting that sensed information does not offer acceptable quality, be it due to environmental conditions or to sensor defects, a robot may replace its current sensing technique by an alternative one revealing as

more suitable under the present circumstances, e.g. by switching to infrared-based sensors if the light conditions are insufficient to rely on daylight camera sensors.

- **Adjusted Perception.** Upon detecting that the perception technique currently applied does not provide for satisfactory quality, a robot may switch to an alternative algorithm, e.g. a different image filtering technique for identifying object patterns.
- **Adjusted Reasoning.** Upon detecting that the targeted action cannot be carried out satisfactorily due to anomalous conditions, a robot may revise its reasoning stage under the constraints just identified, e.g. by adapting its current route planning to allow for an intermediate stop at the closest opportunity if recharging is required.
- **Adjusted Action.** Upon detecting that the targeted action cannot be carried out satisfactorily by means of the techniques currently applied, a robot may select an alternative physical or logical instrument to achieve the same task, e.g. by switching between continuous and discontinuous gait modes for moving depending on the conditions of the terrain and of the robot leg joints [4].
- **Adjusted Autonomy.** Upon detecting that an intended task cannot be carried out at an acceptable level of efficiency, a robot may temporarily transfer part of its autonomy to another entity, be it another robot or a central controller, e.g. by proceeding in a coordinated formation or *platoon* [5], [9].

The fault and anomaly handling strategies just mentioned are summarized in Table 2.

Table 2. On-line fault / anomaly handling strategies

	Technique	**Examples**
Fault Tolerance	fault masking by structural redundancy	N-version programming by majority voting
	error recovery by functional redundancy	recovery block programming by acceptance testing
	error correction by information redundancy	error-correcting codes, cyclic redundancy checks
Reconfiguration	adjustment of sensing instruments	switch between infra-red and daylight camera sensing
	adjustment of perception algorithms	change of filtering algorithms for object patterns identification
	adjustment of decision-making	route replanning for recharging or collision avoidance
	adjustment of action	adaptation of movement mode in case of joint failure
	adjustment of autonomy degree	platooning, delegation

4 Modelling Reconfiguration by Coloured Petri Nets

This section focuses on the representation of reconfiguration behaviour by an appropriate formal notation capable of capturing multiplicity of runtime behaviour including both time-varying, scenario-based operational properties and time-invariant, plant-specific design concepts.

4.1 Multiplicity of Runtime Behaviour

For the purpose of capturing multiplicity of runtime behaviour by appropriate models, the following scenario-dependent attributes must be taken into account:

Robots. Agents are characterized by current information on

- *sensing capabilities* providing in particular continuous feedback on their locations,
- *functional capabilities* reflecting the degree of acceptance of their performance,
- *energy resources* available.

Missions. Tasks are characterized by

- *required functional capabilities* to be provided by entrusted robots,
- *working area(s)* to be accessed / traversed to carry out the mission,
- *current processing status* (available, allocated, completed, degraded).

Environment. Working areas are characterized by

- *sensing and functional complexity*, concerning a. o. visibility or slipperiness,
- *resource consumption*, e. g. depending on path steepness,
- *mobility*, especially concerning the presence of obstacles hindering access.

4.2 CPN Modelling of Behaviour Allowing for Reconfiguration

Based on the positive experiences gained in the past w. r. t. non-reconfigurable robots [2], [3], the notation used in the following to capture both plant-specific reconfiguration concepts and multiple operational conditions is CPN (Coloured Petri Nets [6], [7]), a well-known extension of the more classical Place / Transitions Petri Nets [8]. The main advantage offered by CPN lies in its high compactness and scalability providing appropriate modelling elements to allow to represent both

- permanent, plant-specific reconfiguration concepts by the static CPN part, i.e. by the *net structure* consisting of CPN *places*, *transitions* and *arcs*, and
- time-varying, scenario-dependent information by the dynamic CPN part, i.e. by the *marking* capturing the momentary state by means of flowing CPN *tokens*.

5 Example

The following application is inspired by a robot-based logistics system for hospitals consisting of a number of trolleys moving autonomously along predefined lanes for the purpose of transporting household linen to predetermined places.

5.1 Requirements Concerning Regular Behaviour

Plant Topology. The plant is structured in concentric rings (s. Fig. 2) partitioned into numbered segments, where

- for safety reasons at most one trolley can traverse a segment at any time;
- the two internal circular lanes are used for *clockwise* traffic movement (*inner lane*) and for *anticlockwise* traffic movement (*outer lane*);
- the two external circular lanes (*inner border* and *outer border*) are used as parking lots and partly also as battery-loading areas.

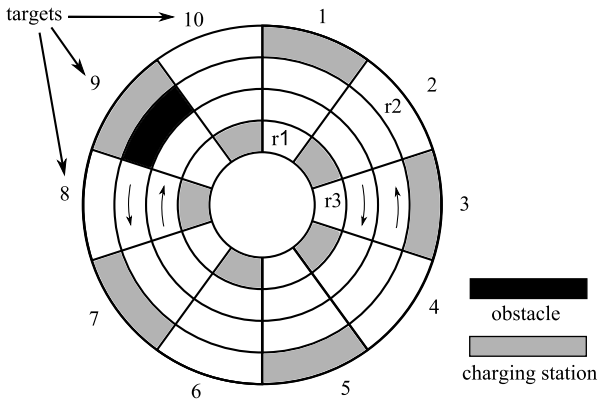

Fig. 2. Plant topology with initial robot positions, obstacle and mission targets

Mission Allocation. Missions are

- characterized by the target segments to be reached;
- allocated to idle trolleys providing the functional capabilities required;
- completed as soon as a border of their target segment is reached.

Robot Movement. Whenever possible, trolleys try to move towards their target:

- once a mission is assigned, the trolley entrusted with it determines its direction (*clockwise* or *anticlockwise*) such as to allow for the shortest path to its target;
- successively, the trolley moves forward by accessing segments as long as they are perceived to be free from obstacles;
- the energy required to traverse a segment amounts to 5 % of a full charge;
- upon reaching its target segment, the trolley moves to the right margin and stops.

5.2 Requirements Concerning Reconfiguration

Upon perception of particular internal anomalies or external anomalous conditions, robots can react by changing their current target behaviour as follows:

Reconfiguration Strategy 1 (R1): Battery Recharging. As soon as their energy level is sensed to be below a predefined threshold, robots will

- target the closest charging station to be encountered in their current direction;
- after conclusion of energy recharging, resume their mission.

Reconfiguration Strategy 2 (R2): Route Replanning. Upon perceiving an obstacle (passive object or human) immediately preceding them on their moving lane, robots

- move to the opposite traffic lane and
- change their moving direction accordingly.

Reconfiguration Strategy 3 (R3): Platooning [5], [9]**.** For reasons of safety, traffic efficiency and energy saving, if disposing of sufficient energy and entrusted with yet incomplete missions, queuing robots build formations by

- temporarily delegating decision-making to the front robot and following their predecessor, hereby reducing energy consumption to 2 % of a full charge per segment traversal;
- as soon as its energy level is under a predefined threshold, a formation-building robot will abandon the platoon to target the closest charging station (s. above), hereby giving rise to a splitting of the platoon;
- as soon as a formation-building robot reaches its target segment, it will abandon its platoon to stop at the margin, hereby giving rise to a splitting of the platoon.

5.3 CPN Models for Increasing Levels of Reconfiguration

The CPN model developed to represent the behaviour illustrated above is shown in Fig. 3. It consists of the following 3 *CPN places* to store state information:

- *MissionPool* captures information on the current mission state,
- *RobotPlatoons* captures information on the current lists of moving formations,
- *Areas* captures information on the current environmental conditions,

as well as of the following 7 *CPN transitions* reflecting generic atomic actions:

- *AssignMission* denotes entrusting a given robot with a given mission,
- *MoveForward* denotes accessing the next segment,
- *ChangeLane* denotes moving to the next closest traffic lane,
- *JoinPlatoon* denotes releasing of autonomy by following the preceding robot,
- *LeavePlatoon* denotes resuming of autonomy w.r.t. movement decisions,
- *Park* denotes moving to the right border to stop there,
- *Charge* denotes recharging the battery.

The CPN net structure G shown in Fig. 3 includes all 3 reconfiguration behaviours R1, R2, R3 mentioned before. By removing the 2 transitions *JoinPlatoon* and *LeavePlatoon* it can be easily degraded to a simplified CPN net structure G' representing the same regular behaviour without allowing for building formations. Assuming the common mission assignment shown in Table 3, these 2 net structures give rise to the following 4 different Coloured Petri Nets:

- CPN0 based on net G', fully charged robots and no obstacles;
- CPN1 based on net G', robots charged for one third (33%) and no obstacles;
- CPN2 based on net G', robots charged for one third (33%) and an obstacle in segment #10 (outer lane);
- CPN3 based on net G, robots charged for one third (33%) and an obstacle in segment #10 (outer lane).

Table 3. Series of CPN models capturing incrementing reconfiguration levels

<table>
<tr><th></th><th>CPN0</th><th>CPN1</th><th>CPN2</th><th>CPN3</th></tr>
<tr><td>reconfiguration</td><td>no</td><td>R1</td><td>R1, R2</td><td>R1, R2, R3</td></tr>
<tr><td>net structure</td><td>G'</td><td>G'</td><td>G'</td><td>G</td></tr>
<tr><td rowspan="3">initial marking</td><td colspan="4">r1: #1(inner parking lane)→#10
r2: #2 (outer parking lane)→#9
r3: #3 (inner parking lane)→#8</td></tr>
<tr><td>a full charge</td><td colspan="3">33% of a full charge</td></tr>
<tr><td colspan="2">no obstacles</td><td colspan="2">obstacle in #10 (outer lane)</td></tr>
</table>

The number of corresponding CPN entities is shown in Table 4, where events denote variable bindings enabling transition firing and state pairs denote pairs of consecutive markings.

Table 4. Size and complexity of CPN models considered

	CPN0	CPN1	CPN2	CPN3
# transitions	5	5	5	7
# events	23	42	70	1966
# states	454	1265	3581	17325
# state pairs	981	2508	7537	30824

Within the set of all state pairs of CPNi ($i \in \{1, 2, 3\}$) we may further distinguish the subset of Ri-state-pairs initiating reconfiguration Ri, more precisely:

- R1-state-pairs traversed by firing transition *Park* for charging purposes, i.e. when the parking segment differs from the mission target;
- R2-state-pairs traversed by firing transition *ChangeLange* upon sensing an obstacle on the next segment to be accessed by the robot considered;
- R3-state-pairs traversed by firing transition *JoinPlatoon*.

The number of the Ri-state-pairs ($i \in \{1, 2, 3\}$) is shown in Table 5.

Table 5. Number of state pairs involving corresponding reconfiguration levels

i	1	2	3
#Ri-state-pairs in CPNi	198	294	1448

6 Incremental Reconfiguration Testing

This section is devoted to the definition of a systematic procedure for the determination of CPN test cases targeted to the verification of the 3 reconfiguration strategies illustrated above. Hereby, a CPN test case is defined as an initial marking followed by a sequence of events occurring during its execution such as to represent behaviour from mission assignment to mission accomplishment. As Ri-state-pairs are defined to initiate reconfiguration, their coverage suffices to test the complete corresponding reconfiguration behaviour.

It is envisaged to reduce testing effort by avoiding to retest behaviour already verified beforehand. The testing procedure proposed for this purpose addresses increasing levels of reconfiguration; at each level, appropriate test cases must traverse all pairs of consecutive states involving corresponding reconfiguration behaviour. Evidently, the advantage of proceeding incrementally is the stepwise inclusion of anomalous operational conditions. In other words, after regular behaviour has been verified by extensive coverage of CPN0, additional reconfiguration testing requires further test cases covering Ri-state-pairs in CPNi for the purpose of addressing reconfiguration strategies Ri ($i \in \{1, 2, 3\}$).

Depending on the state transition graph generated by CPN Tools [7], appropriate test cases may be determined by analytical [10] or heuristic approaches [11], [12]. In the present case, appropriate test cases were analytically generated by *hot-spot prioritization* [10] based on the successive identification of test cases providing the maximum number of state pairs yet uncovered. Their number is shown in Table 6.

Table 6. Number of test cases required by different test coverage criteria

test coverage criterion in CPNi	i = 0	i = 1	i = 2	i = 3
# test cases covering all state pairs	133	270	614	3981
# test cases covering Ri-state-pairs	n.a.	111	239	1011

The incremental testing approach proposed reveals as beneficial in terms of testing effort, as it requires only 1494 test cases (the sum of the shaded entries in Table 6) instead of the 3981 test cases required by non-incremental reconfiguration testing.

7 Conclusion

In order to verify reconfiguration behaviour potentially emerging from interacting autonomous agents to be exclusively beneficial and never hazardous to cyber-physical systems, this article proposed a systematic approach based on incremental model-

based testing and illustrated its application to cooperating mobile robots involving 3 different reconfiguration strategies.

The testing procedure derived is based on the automatic generation of test cases covering a series of CPN models capturing increasing degrees of reconfiguration. An exemplifying application confirmed both the practicality and the cost-effectiveness of the approach developed.

Acknowledgment

The authors gratefully acknowledge that part of the work presented was carried out within the European Research Programme ARTEMIS (Advanced Research and Technology for Embedded Intelligence and Systems), project R5-COP (Reconfigurable ROS-based Resilient Reasoning Robotic Co-operating Systems), agreement number 621447.

References

1. Bajd, T. et al: Robotics. In: Series: Intelligent Systems, Control and Automation: Science and Engineering, Vol. 43, Springer, 2010.
2. Saglietti, F., Lill, R.: A Testing Pattern for Automatic Control Software Addressing Different Degrees of Process Autonomy and Cooperation, In: Proc. 19th IFAC World Congress, International Federation of Automatic Control, 2014.
3. Lill, R., Saglietti, F.: Model-based Testing of Cooperating Robotic Systems using Coloured Petri Nets. In: SAFECOMP 2013 Workshop Proceedings, LAAS-CNRS, 2013.
4. Krishnan, V. L. et al.: Reconfiguration of Four-legged Walking Robot for Actuator Faults. In: Proc. Spring Simulation Multiconference (SpringSim '10), ACM Digital Library, 2010.
5. Klančar, G., Matko, D., Blažič, S.: Wheeled mobile robots control in a linear platoon. In: Journal of Intelligent and Robotic Systems, Vol. 54(5), 2009.
6. Jensen, K., Kristensen, L.M.: Coloured Petri Nets. Springer, 2009.
7. Jensen, K., Kristensen, L. M.; Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. In: International Journal on Software Tools for Technology Transfer (STTT), Vol. 9, No. 3-4, Springer, 2007.
8. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, 77(4), 1989.
9. Bergenhem, C., Shladover, S., Coelingh, E.: Overview of Platooning Systems. In: Proceedings 19th World Congress on Intelligent Transportation Systems (ITS), 2012.
10. Wong, W.E., Lei, Y., Ma, X.: Effective generation of test sequences for structural testing of concurrent programs. In: Proc. 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 2005.
11. Lill, R., Saglietti, F.: Testing the Cooperation of Autonomous Robotic Agents. In: Proc. 9th International Conference on Software Engineering and Applications (ICSOFT-EA), Scitepress Digital Library, 2014.
12. Saglietti, F., Föhrweiser, D., Winzinger, S., Lill, R.: Model-based Design and Testing of Decisional Autonomy and Cooperation in Cyber-physical Systems. To be published in: Proc. 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE Xplore Digital Library, 2015.

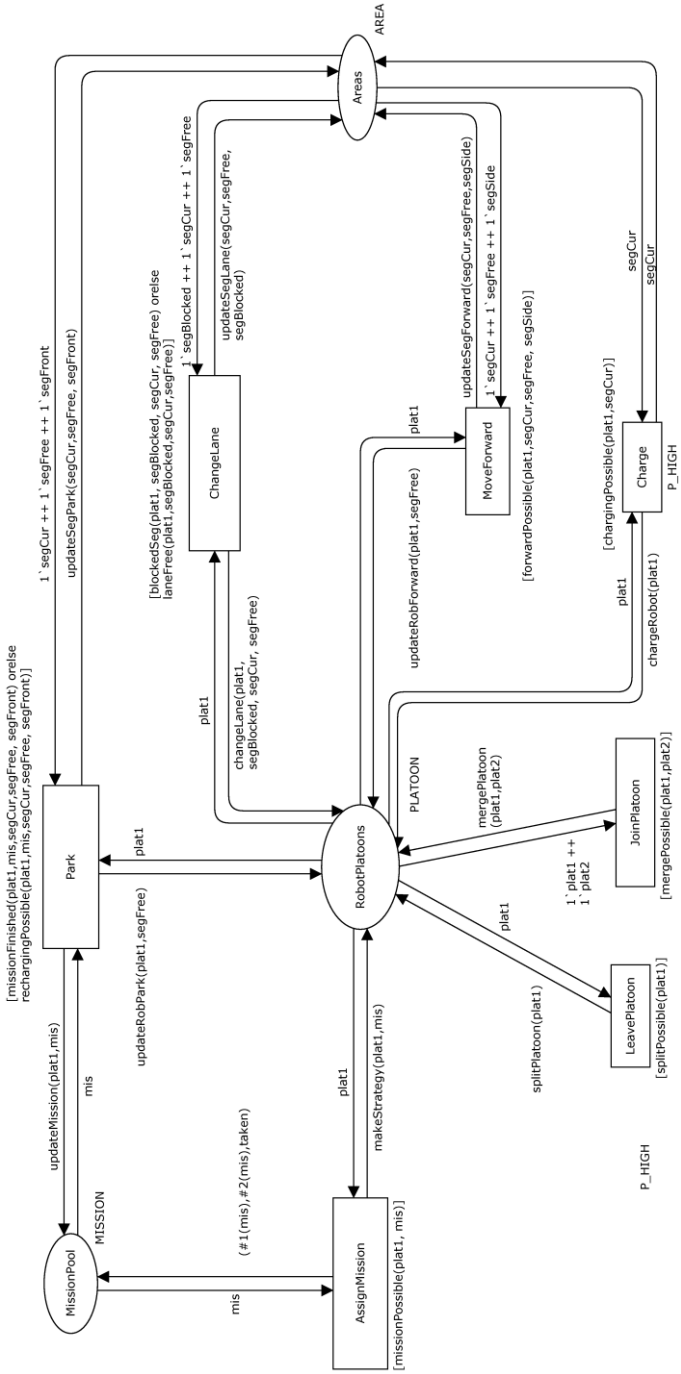

Fig. 3. CPN model of reconfigurable cooperating robots