A Formalization of the Coach Problem

Citation for published version:
Schropp, GYR, Meyer, JC & Ramamoorthy, S 2015, A Formalization of the Coach Problem. in RoboCup 2014: Robot World Cup XVIII. Lecture Notes in Computer Science, vol. 8992, Springer International Publishing, pp. 345-357. DOI: 10.1007/978-3-319-18615-3_28

Digital Object Identifier (DOI):
10.1007/978-3-319-18615-3_28

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
RoboCup 2014: Robot World Cup XVIII

A Formalization of the Coach Problem

G.Y.R. Schropp[1], J-J. Ch. Meyer[1], and S. Ramamoorthy[2]

[1] Utrecht University (UU), The Netherlands
gwendolijn.schropp@phil.uu.nl
[2] University of Edinburgh (UoE), United Kingdom

Abstract. Coordination is an important aspect of multi-agent teamwork. In the context of robot soccer in the RoboCup Standard Platform League, our focus is on the *coach* as an external observer of the team, aiming to provide his teammates with effective tactical advice during matches. The coach problem can be approached from different angles: in order to adapt the behaviour of his teammates, he should at first be able to perform *plan recognition* on their observable actions. Furthermore, in providing them with appropriate advice, he should still adhere to the norms and regulations of the match to prevent penalties for his team. Also, when teammates' profiles and attributes are unknown or the system is only partially observable, coordination should be more 'ad hoc' to ensure robustness of the Multi-Agent System (MAS). In this work, we present a formalization of the problem of designing a coach in robot soccer, employing a temporal deontic logical framework. The framework is based on *agent organizations*[14], in which social coordination and norms play an important part.

Keywords: agent organization, multi-agent system, teamwork, coordination, logic, plan recognition

1 Introduction

RoboCup's Standard Platform League (SPL) now allows for a coach robot, whose main role is to provide tactical advice to his team. Besides that, he could help the team in disambiguating signals (e.g. in localization) and give them high level suggestions to improve their play. As the coach is a novel addition to the SPL, there is currently little guidance on what a good coach should do. In contrast, there is a lot of work on for example path planning and localization. As such, the need for a formal understanding of the coach's role arises. This formalization is the goal of this paper. In the simulation and middle size leagues however, coaching has been the topic of research since 2001 [45].

For the formalization of the robot soccer system we use the concept of agent organizations: a set of entities or agents that are regulated by social order in achieving common goals. A logic-based framework enables a precise and formal but abstract specification, which could guide many different kinds of detailed future implementations via agent programming languages [14, 12]. Because agents are left unspecified in our framework, they can be developed according to the needs and wishes of the designer, yielding a flexible and adaptable system [14]. The contribution of this paper is is to present a formalization using deontic temporal logic, capturing the specification of the coach within the RoboCup SPL, and describing the rules and temporal aspects of such a system. Once a type of agents have been chosen, this framework is ready to be implemented. Suggestions for implementation are also given in this work.

2 Related Research

There are few formal MAS for robot soccer based on agent organizations. However, aspects of our framework are related to ideas that have appeared previously in the literature on multiagent systems and robotics.

Agent organizations and developing methods have been presented in [14, 16, 33, 47, 17]. Esteva et al. [16] and Dignum [14] introduce the notion of *norms* in combination with the agent *roles* already used by Odell et al. [33], Ferber and Gutknecht [17] and Wooldridge [47]. Roles can be used to decompose the tasks to be performed by the MAS into sub-objectives to increase efficiency [44, 18, 42]. The role-based approach to ad hoc teamwork by Genter et al. [18] determines role selection on the team's utility. In robot soccer, roles have been based on absolute position in the field [32], position relative to the ball [2, 3, 31] and robot trajectories, sometimes also considering positions of other players [48].Other work on team coordination through roles and dynamic positioning can be found for example in [28, 27, 13]. Desheng et al. achieve collaboration between simulated agents via roles in situation calculus. One of the characteristics of the human organizations framework used in this work is communication or negotiation. However, coordination without negotiation [22] is more appropriate if an ad hoc approach is required. Tracking multiple soccer players' trajectories can for example be done using the camera feed, in combination with an analyzing system that segments motions into classified ball actions [2]. Extracting tactic events from human soccer video feeds has also been done using spatio-temporal interaction among players and the ball [48].

In order to handle the team's knowledge representation, it is important that the robots share the same definitions of concepts in their environment. Moreover, these concepts should be *grounded* in order to link them to the robot's percepts of the real world, for example via *ontologies* [20, 35] and pattern recognition [24]. Also, a communication language for the coach should be defined, for example the COACH UNILANG language, enabling both high-level and low-level coaching through coach instructions [37].

As a first step towards integrating the framework in a coach robot, we pay special attention to the problem of *plan recognition*. There's related work on the use of *roles* to infer a robot's plans or intentions [4, 5, 7, 18]. 'Plans' can be interpreted in various ways, ranging from a sequence of actions currently performed according to some behaviour [7, 40] to a robot's entire internal state (e.g. belief base, goals). The latter approach is sometimes called 'opponent modelling' and used to learn an optimal strategy against the modelled type of agent [1, 8, 38, 39]. Although plan recognition was first introduced in the field of logical inference by [25], the current state of the art is mainly based on probabilistic methods like Bayesian Networks [6, 7, 9] and Markov Models [6, 4, 21, 10, 46, 26]. For example in [39], simulated robot soccer players adapt their positions strategically in adaptive response to the opponent's behaviour throughout the game. As for logic-based approaches, the work by Sindlar et al. [43] seems to fit our framework best: the highly structured and regulated character of robot soccer seems suitable for their abductive reasoning approach to intention recognition (more in section 5). Besides abductive reasoning, intention recognition has also been modelled formally, in a language based on situation calculus [29, 19]. In this approach, observed behaviour is incrementally matched to an annotated library of plans, which makes it more restricted than Sindlar's approach.

3 Formal Framework

For the formalization of the coach problem in the context of RoboCup SPL, we used Dignum's OperA framework development methodology [14], as it is formal and elaborate enough to precisely describe the system while still being flexible, reusable and adaptable to specific agent designs and future innovations.

In OperA, structures are described in a formal logic called 'logic for contract representation' (LCR), which is a combination of CTL* (computation tree logic, a temporal logic), STIT ('sees to it that') and Deontic expressions [14]. Deontic logic is the logic of norms (obligations, prohibitions, permissions, violations and sanctions), allowing reasoning about ideal states versus actual states of behaviour [23]. STIT logic is used to determine which agent should 'see to it that' a certain goal is achieved [14]. Since robot soccer is a highly regulated game with both time specific and role specific tasks and events, LCR is an appropriate language to describe it. In our domain we don't need the full specification of LCR because, for example, communication is limited in comparison to that in a human organization [41].

OperA's methodology contains several layers of design: the Organizational Model is by far the most elaborate layer, in which the characteristics of the domain are given in *social, interaction, normative* and *communicative structures*. The Social and Interaction Models are meant to instantiate specific agents and interactions for actual implementation, which is largely outside the scope of this work. The OperA development methodology yields a formal model with descriptions of the roles, rules and interactions of the robot soccer system. This framework allows for extensions and adaptations of existing interactions.

3.1 Organizational Model

Roles and Dependencies Roles are an important part of the framework. Two kinds of roles should be distinguished: facilitation roles and operational roles. Moreover, as our domain contains both humans and robots, a difference is made between roles that can be enacted by human or robot agents. Specification of domain concepts and entities is given in a domain ontology and in terms of *identifiers* respectively. The ontology is, to certain extent, developed in Protégé[3], similar to Opfer's approach for the Middle-Sized League [35]. It includes formulas describing (parts of) the field (e.g. $\forall x.isPartOf(x, OppArea) \leftrightarrow isPartOf(x, OppHalf)$: all areas within the opponent's area are also within the opponent's half of the field). Identifiers are used as names for the sets of agents (both human and robot). Opponent robots are not included in the framework at this stage but can be added in future work.

The human roles of our domain are {head-referee, assistant-referee, GameController-operator, human-teammember}[11], whereas the robot roles are {goalkeeper, defender, attacker, coach}. The amounts of robots playing each role depends on specific team formation (except that there is always only one goalkeeper and one coach). At least one human should enact the human-teammember role, meaning that he/she is able to request for pick ups and time outs.
Roles in the robot soccer domain are defined as tuples $role(r, Obj, Sbj, Rgt, Nor, tp)$ where $r \in Roles$ is the identifier of a role, $Obj \subseteq Act$ is the set of objectives of the role, $Sbj \subseteq Act$ is the set of sub-objectives sets of the role, $Rgt \subseteq Deon$ are the rights of the role and $Nor \subseteq Deon$ the norms of the role. $tp \in \{operational, institutional\}$ is the type of the role [14, 41]. Institutional roles (e.g. referees) are typically enacted by impartial agents, ensuring global activity, while actors of operational roles aim to achieve their part of the society goals.

Based on these roles, agents have certain *(sub-)objectives* to achieve and *norms* to adhere to. For example, a coach should aim to send tactic messages to the players but is not allowed to leave its seated position beside the field. These norms and objectives are formalized in LCR to facilitate future implementation. Norms are based on the official RoboCup SPL rules and identified via a Norm Analysis method, yielding the responsible roles and triggers for each norm [14, 41]. The coach role is given as an example in table 1. The sub-objectives as defined in this table are merely suggestions for how to handle the plan recognition module in combination with a decisionmaking module yet to be developed. In section 5 this will be discussed in more detail.

In order to achieve their objectives, enactors of roles depend on each other. These role dependencies determine the interactions that occur in the system. Role dependencies depend on the *power relations* between roles, for example, players can *request* things from one another while the coach's advice should perhaps have a higher priority. The robot soccer roles and their dependencies per objective are depicted in graph 1. For example for the dependencies written in red: the coach depends on the GameController-operator to send his messages to the players, and the head-referee depends on the assistant-referee(s) to apply his requests.

Interaction Scenes and Landmarks The interactions, determined by role dependencies, are described as 'interaction scene scripts' and can be seen as

[3] http://protege.stanford.edu

Role: Coach	
Role id	coach

Objectives	o1 := messaged-tactics
	o2 := followed-rules

Sub-objectives	Πo1 = ({∀p∈ Players: executed-plan-rec-module(p, role(p), t), got-plan(p, plan)), got-tactic-list(plan, formation, Tactics), decided-tactic(Tactics, tactic),got-msg(tactic, msg), message-sent(coach, GC-op, msg), wait(10s)} Πo1' =({∀p∈ Players: executed-plan-rec-module(p,t), got-role-map(plan(p), role(p))), got-formation-map(role(p), Formations), got-team-tactics(formation, TeamTactics), decided-tactic(TeamTactics, tactic), got-msg(tactic, msg), message-sent(coach, GC-op, msg), wait(10s)}
Rights	message-via-GC-op, decide-tactic(coach, (Team)Tactic)

Norms	PROHIBITED(coach, move(¬(head∧arms)))
	PROHIBITED(coach, communicate(coach, Robots, direct))
	PERMITTED(coach, have-clothes(anyColor, anyPattern))
	OBLIGED(coach, meet-msg-requirements(Msg, [Msg-Requirements]))

Type	operational

Table 1. Role definition for the coach; t = window of observation, msg = message.

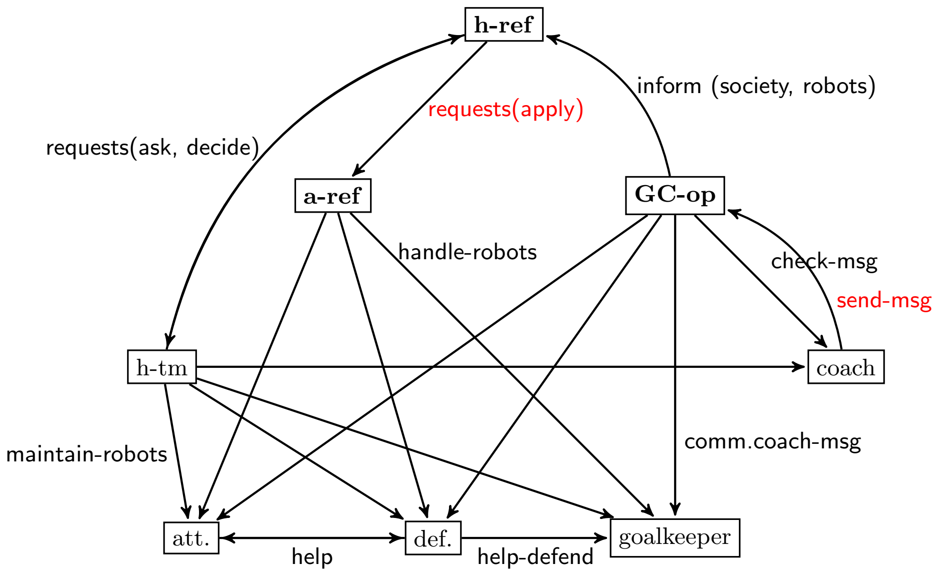

Fig. 1. Role dependency graph

the coordination of interactions among several roles. Scene scripts are tuples **scene(s,Rls,Res,Ptn,Nor)** where **s** is the identifier of the scene, **Rls** is the set of identifiers of the roles enacting the scene, **Res** are the results of the scene in terms of *achievement expressions*, **Ptn** the interaction patterns (sub-achievements) and **Nor** the relevant norms of the agents in the scene. Achievement expressions are statements that describe the situation of a state after a certain goal is achieved (e.g. 'penalty-applied', 'goal-scored', 'ball-passed'). The notion of *landmarks* is used to represent such states. How exactly these landmarks have been reached is not defined at this level since it depends on the specifics of the participating agents.

In the Organizational Model, all roles, dependencies, norms and interactions of the 'robot soccer society' are formally defined. Since the entire model is quite elaborate, please see [41] for the complete framework.

3.2 Social and Interaction Models

Where the Organizational Model consists of the formal model of the society, the Social Model continues with the explicit representation of how an agent will enact such a role. At this level, the requirements and conditions of actual agents can be taken into account. When a specific agent is assigned a role, he becomes a *role-enacting agent (rea)*. The roles as described in the Organizational Model can be adjusted to the wishes of that specific agent about to enact it. For example, if an agent can only enact 'assistant-referee' for half a match, this can be adjusted in his *social contract*. That contract describes the role-enacting agent: his expected behaviour when he enacts that role. That is, a role as described in the Organizational Model can be enacted in various ways by different agents, based on their personal objectives and functionality. This is described in the social contract. However, as this is fully dependent on the specific agents yet to be implemented, we cannot describe the link between those unknown agents and the general definitions in the Organizational Model.

The same holds for the Interaction Model, where the specific role-enacting agents of the Social Model can be combined with interaction scenes from the Organizational Model to 'instantiate' the scenes. This happens in a similar way as described above, via instatiation of *interaction contracts* in which the wishes of the participants in the scene are reflected. The method to go about the instantiation of the Social and Interaction Models is given in [41, 14].

3.3 Validation

Since our framework currently is abstract and intended mainly to formalise the specifications, we can not as yet experiment with it to verify it works as intended. OperA frameworks can however be validated and verified in multiple ways. The main requirements as presented in [14] are meant to check the model for inconsistencies and contradictions. Besides formally verifying its structures, the framework should also be checked to represent the objectives of the society. As we only developed the Organizational Model in detail, this verification step can only occur at the organizational level. On this level the roles, objectives and dependencies are confirmed to represent society purposes. Furthermore, we haven't found conflicts within or between the descriptions of roles, results, objectives, norms and scenes. Clearly, this has already been kept in mind in the development phase. When instantiating the Social and Interaction Models, one should make sure its contracts do not contradict the descriptions of the Organizational Model.

Besides a formal verification, a note on the robot soccer specific validation is in order. As this framework is based firmly on the official RoboCup SPL regulations and developed in collaboration with Edinferno's team (inspiration has been drawn from earlier implementations together with advice and ideas of current members), it is not only formally verified but also validated on content.

4 Plan Recognition

A necessary first step in implementing this framework is a plan recognition module for the coach. The idea here is that the coach should be able to identify a player's behaviour from observed actions only. We assume that behaviour in this sense is a set of goals and plans to achieve those goals, leading to characteristic *trajectories* on the field. The trajectories can be tracked by means of the player's self-localization module and collected in the form of a vector containing relative distances and angles to the possible goals. We choose to test on two behaviours representing the intention to go either towards the center of the 'own' goal or the 'opponent' goal. The problem is modelled as a parameterized Markov Chain $M : (S, A, T)$, where S is a finite set of states, A is a finite set of actions and T is the transition function: for any state and action $\mathbf{s} \in S$, $\mathbf{a} \in A$, the probability of each possible next state s' is $\mathrm{T}(\mathbf{s'} \mid \mathbf{s}, \mathbf{a})$. The state vector contains the distances and angles from the player, the action vector the distances travelled between the current state and a next state. For both behaviours, sets of states and actions are collected as training data, based on which probabilities are calculated: given a behaviour and a current state, what is the most likely next state for the player to be in. A Gaussian distribution is fitted to the training data to cover the entire soccer field for both behaviours.

In the test phase, for each observed state of the player, the most likely transition is calculated using the multivariate normal density function. Using these likelihoods, maximum log-likelihoods for entire observed trajectories are found. Behaviours are classified using Bayes' rule (eq. 1): the behaviour (b) with the highest log-likelihood given the observations (O) is chosen as the correct one.

$$argmax_b \; p(b|O) = argmax \; p(O|b) \; p(b)$$

$$\text{Decide } \; b_1 \; \text{ if } \; p(b_1|O) > p(b_2|O); \; \text{ otherwise, decide } \; b_2. \tag{1}$$

For the 'go to opponent goal' behaviour we collected 60 trajectories, for the 'go to own goal' behaviour 46. These were divided in a 80/20 ratio into training and validation sets, where the training sets yielded the probability distribution on the field given a certain behaviour. For 'go to opponent goal' we measured

91.6% precision, for ‘go to own goal’ only 44.4%. These numbers are the result of an initial implementation within the framework. More sophisticated models could have been used, for example Bayesian network representations [30]. Despite smoothing, some trajectories still did not represent their actual paths. That is, the logs showed jumps in positions or otherwise incorrect coordinations, mostly due to mirrorring of the field. Also, the amount of training data for the own goal behaviour is considerably less. A lot of the collected trajectories could not be used due to flaws with the player’s self-localization. However, it should be noted that the differences between the maximum log-likelihoods that determined the classification decision were in most cases extremely small. A more sophisticated classifier appears necessary for stronger results in this direction.

5 Application

There are several roads to follow from here in implementing the coach. Two suggestions will be presented in this section. The first suggestion is to continue with the current plan recognition module and integrate it with a decision making and tactic adaptation module, for example in one of the following ways:

1. Tactics as independent sub-objectives
2. Tactics as dependent sub-objectives
3. Team tactics

For all these options, we assume that the coach has a library of plans and corresponding messages that he can send. The decision making module serves to compute the optimal plan given the observed current plan per single player (1 and 2) or for the entire team (3). In the first suggestion, we take the coach’s advice to be obligatory for the player to perform and independent of the role of that player: the coach merely observes the game situation and decides tactic plans based on player position without including their role specific abilities. This could be a naive first test of such a decision module. The second option does consider the player’s role: the coach decides optimal tactics like before, but sends messages to the role-enacting agent with the most suitable role and position to perform that tactic. The third option is to infer current roles and plans of the entire team before deciding an optimal team tactic. The players should re-arrange and divide the coach’s tactic plans between themselves in a similar way as the current method of role switching (e.g. based on position relative to the ball). For example, if the coach sends a plan involving passing a ball, then the robot that is currently closest to the ball is in the best position to execute that plan.

However, should one want to connect the logical framework to a logical method of plan recognition, we suggest to continue along the lines of the work done by Sindlar et al. [43]. They propose a method for intention recognition via *mental state abduction* (MSA), were agents are assumed to be BDI agents (Belief/Desires/Intentions [36]). Based on observations combined with knowledge of the rules and roles, it can be inferred why an agent performs a certain action.

MSA uses ‘answer set programming’ (ASP) for nonmonotonic, abductive reasoning in the agent programming language (2)APL [12]. In this language, a goal achievement rule of the form $n : \gamma \leftarrow \beta \mid \pi$, where n is the identifier of the rule, γ the goal to be achieved, β the beliefs that should be true in order to be allowed or able to perform plan π, where π generates observable action sequences leading to achieving that goal γ. That is, a plan can generate multiple observable sequences and also multiple *computation sequences*, meaning that various different routes can lead to the same goal. There is a subtle difference between *observable* and *seen* actions: an observable action is a possible observation (something that is possible according to the theory), while seen actions have actually been observed. Intuitively, if an action is seen, it should also be observable [43].

From these APL rules, a translation step is made to a logical theory and subsequently to a logical program in ASP style, making it directly implementable. We will give a short example for our domain to explain how MSA works. Consider the rule $R = \{1:$ `hold-ball` $\leftarrow$ `in-ownPA(g) and in-ownPA(b)` $\mid$ `move(g,b); if B(opponent-near) then pickup(g,b) else skip`$\}$, which would lead to the answer set program P_R:

```
2{g(hold-ball,0), b(conj(in-ownPA(g), in-ownPA(b)),0)}2 :- r(1,1)
2{g(hold-ball,0), b(conj(in-ownPA(g), in-ownPA(b)),0)}2 :- r(1,2)
                       2{o(move(g,b),1), o(pickup(g,b),2)}2 :- r(1,1)
                                          1{o(move(g,b),1)}1 :- r(1,2)
                                                   1{r(1,1), r(1,2)}1.
                                                 :- s(A,T), not o(A,T),
```

where the last statement says that candidate answer sets that were seen, but not deemed observable at step `T` should be discarded. The reason that `r(1,1)` and `r(1,2)` are the same, is because the plan of this rule has two possible computation sequences, represented by the two possible observations given (depending on the belief of ‘opponent-near’). The belief whether or not the opponent is near is a different kind of belief than β since it is a test case in the plan (π) part of rule R: B(opponent-near) does not need to be satisfied to execute rule R, while beliefs β (`in-ownPA(g) and in-ownPA(b)`) should.
Let us say we have seen the goalkeeper moving towards the ball:
$P' = P \cup \{$`s(move(g,b),1)`$\}$. This can be explained by both `r(1,1)` and `r(1,2)`. Next, we see him picking up the ball: $P'' = P' \cup \{$`s(pickup(g,b), 2)`$\}$; this can only be explained by `r(1,1)`. We can now infer
$P'' \models$ `goal(hold-ball)`$\wedge$`bel(conj(in-ownPA(g), in-ownPA(b)))` $\wedge$`bel(opponent-near)`, revealing the mental state of our goalkeeper based on observed actions and known rules.

6 Discussion

The organizational MAS development method ‘OperA’ yielded a grounded formalization of the robot soccer society and a detailed description of its roles and coordination. However, a drawback of this method is that guidelines for actual implementation are not provided. Recent extensions like OMNI [15] and OperettA [34] explore possible means of implementation for such frameworks. The fact that agent designs are left out of the framework is presented as an advantage, as it gives the designer freedom to adjust the agents to his needs while they can still be used in the general yet formal framework [14]. This way, there are multiple options for future work, like the modular approach we started on or the BDI/abduction approach we suggested above.

This formalization of the coach role has been largely based on RoboCup's regulations [11]. Requirements for the coach's messages have been mentioned in [41], but the content of these messages has not been defined yet. This depends on the decisions made in the actual development of the coach: what would be most valuable for him to say in order to help the team. The format for the messages as described in the regulations has been included in the framework. All other specifics as mentioned in the regulations are included in this formalization [41]. Examples and additions to these specifics are inspired by (former) RoboCup members.

The plan recognition module as presented above is merely a naive first pass implementation that could be improved in several ways. For example, our initial implementation assumed that the coach has access to the player's self-localization logs, while ideally he should be able to use actual observations. Furthermore, we only tested on static goals, while ball behaviour would be more informative to decide tactics. However, from this first step we gained some important insights into the difficulties of the coach problem.

7 Conclusion and Further Research

In general, the OperA framework has shown to be very suitable to describe robot soccer in terms of roles and interaction structures. The coach role, as described by RoboCup's regulations, has been formalized within this framework. As it leaves agent designs undefined, the next step is to choose for example between the modular approach introduced in 4 and the logical approach based on BDI agents suggested in 5. As robot soccer is a highly regulated game, and all robots should have a knowledge base of the rules in any case, it seems intuitive to continue along the lines of mental state abduction. So far we only spoke of full observable sequences, but work on sequences with gaps (partially observable) already exists [42]. Besides implementation of a new and improved plan recognition module, the tactic decision making problem is also subject for further research.

References

1. N. Bard and M. Bowling. Particle filtering for dynamic agent modelling in simplified poker. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22(1), 2007.

2. M. Beetz, J. Bandouch, and S. Gedikli. Camera-based observation of football games for analyzing multi-agent activities. In *Proceedings of AAMAS '06*, 2006.
3. S. Behnke, J. Müller, and M. Schreiber. Playing soccer with robosapien. *In: A. Bredenfeld et al. (editors): RoboCup 2005, LNAI 4020, Springer*, pages 36–48, 2006.
4. C. Boutilier. Sequential optimality and coordination in multiagent systems. *International Joint Conferences on AI*, 99:478–485, 1999.
5. M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *Proceedings of AAAI'05*, pages 53–58, 2005.
6. H.H. Bui. A general model for online probabilistic plan recognition. *International Joint Conferences on AI*, 3:1309–1315, 2003.
7. S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48, 2001.
8. D. Carmel and S. Markovitch. Model-based learning of interaction strategies in multi-agent systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):309–332, 1998.
9. E. Charniak and R.P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.
10. C. Claus and C. Boutilier. Reinforcement learning in cooperative multiagent systems. In *Proceedings of AAAI-98*, pages 746–752, 1998.
11. RoboCup Technical Committee. Robocup standard platform league (nao) rule book, 2013.
12. M.M. Dastani. 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.
13. X. Desheng and X. Keijan. Role assignment, non-communicative multi-agent coordination in dynamic environments based on the situation calculus. In *Proceedings of the WRI Global Congress on Intelligent Systems*, volume 1, pages 89–93, 2009.
14. V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. PhD thesis, Utrecht University, 2004.
15. V. Dignum, J. Vázquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. *In: R.H. Bordini et al.(editors): PROMAS 2004, LNAI 3346*, pages 181–198, 2005.
16. M. Esteva, J.A. Rodríguez-Aguilar, C. Sierra, P. Garcia, and J.L. Arcos. On the formal specification of electronic institutions. *In: F. Dignum, C. Sierra (editors): Agent-Mediated Electronic Commerce (The European AgentLink Perspective), LNAI 1991, Springer*, pages 126–147, 2001.
17. J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, 1998.
18. K. Genter, N. Agmon, and P. Stone. Role-based ad hoc teamwork. In *Proceedings of PAIR-11 (workshop at AAAI)*, 2011.
19. A. Goultiaeva and Y. Lespérance. Incremental plan recognition in an agent programming framework. In *Proceedings of Plan, Activity and Intent Recognition (PAIR)*, 2007.
20. M. Grüninger and M.S. Fox. Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, 1995.
21. K. Han and M. Veloso. Automated robot behavior recognition. *Robotics Research - International Symposium*, 9:249–256, 2000.

22. M. Isik, F. Stulp, and H. Utz. Coordination without negotiation in teams of heterogeneous robots. *In: G. Lakemeyer, E. Sklar, T. Sorrenti (eds): RoboCup 2006: Robot Soccer World Cup X, LNAI 4434. Springer.*, pages 355–362, 2007.
23. F. P. M. Dignum J.-J. Ch. Meyer, R. J. Wieringa. The role of deontic logic in the specification of information systems. *In: J. Chomicki, G. Saake (editors): Logics for Databases and Information Systems, Kluwer Academics Publishers*, pages 71–115, 1996.
24. B. Johnston, F. Yang, R. Mendoza, X. Chen, and M. Williams. Ontology based object categorization for robots. *In: T. Yamaguchi (editor): PAKM 2008, LNAI 5345. Springer.*, pages 219–231, 2008.
25. H.A. Kautz and J.F. Allen. Generalized plan recognition. In *Proceedings of AAAI-86*, volume 86, pages 32–37, 1986.
26. A. Kleiner, M. Dietl, and B. Nebel. Towards a life-long learning soccer agent. *In: G.A. Kaminka, P.U. Lima, R. Rojas (editors): RoboCup 2002, LNAI 2752. Springer*, pages 126–134, 2003.
27. J.R. Kok, M.T.J. Spaan, and N. Vlassis. Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50(2-3):99–114.
28. N. Lau, L.S. Lopes, G. Corrente, and N. Filipe. Multi-robot team coordination through roles, positioning and coordinated procedures. In *Proceedings of IROS 2009*, 2009.
29. H. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Computer and Information Science*, 3(18), 1998.
30. L. Liao, D.J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171, 5-6:311–331, 2007.
31. P. MacAlpine, F. Barrerra, and P. Stone. Positioning to win: A dynamic role assignment and formation positioning system. In *Proceedings of the RoboCup International Symposium 2012*, 2012.
32. M. Mohr, P. Krustrup, and J. Bangsbo. Match performance of high-standard soccer players with special reference to development of fatigue. *Journal of Sports Sciences*, 21:7:519–528, 2011.
33. J.J. Odell, H. Van Dyke Parunak, and M. Fleischer. The role of roles in designing effective agent organizations. *in A. Garcia et al. (editors): SELMAS 2002, Lecture notes in computer science 2603*, pages 27–38, 2003.
34. D. Okouya and V. Dignum. Operetta: A prototype tool for the design, analysis and development of multi-agent organizations (demo paper). In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 2008.
35. Stephan Opfer. Towards Description Logic Reasoning Support for ALICA. Master's thesis, Universität Kassel, 2012.
36. A.S. Rao and M.P. Georgeff. Modeling rational agents within a bdi-architecture. *KR*, 91:473–484, 1991.
37. L.P. Reis and N. Lau. Coach unilang: a standard language for coaching a (robo)soccer team. In *In: A. Birk, S. Coradeschi, S. Tadokoro (eds): RoboCup 2001: Robot Soccer World Cup V, LNAI 2377. Springer*, pages 183–192, 2002.
38. P. Riley and M. Veloso. Coaching a simulated soccer team by opponent model recognition. *AGENTS'01*, 2001.
39. P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. *RoboCup 2001: Robot Soccer World Cup V. Springer*, pages 453–458, 2002.
40. S. Saria and S. Mahadevan. Probabilistic plan recognition in multiagent systems. In *Proceedings of ICAPS-04, AAAI*, pages 287–296, 2004.
41. G.Y.R. Schropp. Agent organization framework for coordinated multi-robot soccer. Master's thesis, Utrecht University, 2014 (in review).

42. M.P. Sindlar, M.M. Dastani, F. Dignum, and J-J.Ch. Meyer. Mental state abduction of bdi-based agents. *In: M. Baldoni et al. (editors): DALT 2008, LNAI 5397. Springer.*, pages 161–178, 2008.
43. M.P. Sindlar, M.M. Dastani, and J-J.Ch. Meyer. Programming mental state abduction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 301–308, 2011.
44. P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.
45. U. Visser, C. Drucker, S. Hubner, E. Schmidt, and H. Weland. Recognizing formations in opponent teams. In *In: P. Stone, T. Balch, G. Kraetzschmar (eds): RoboCup 2000, Robot Soccer World Cup 4, LNAI 2019. Springer.*, pages 391–396, 2001.
46. T. Weigel, K. Rechert, and B. Nebel. Behavior recognition and opponent modeling for adaptive table soccer playing. *In: U. Furbach (editor): KI 2005, LNAI 3698. Springer.*, pages 335–350, 2005.
47. M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.
48. G. Zhu, C. Xu, Q. Huang, and W. Gao. Automatic multi-player detection and tracking in broadcast sports video using support vector machine and particle filter. In *Proceedings of the International Conference on Multimedia and Expo*, pages 1629–1632, 2006.