Role-Based Management and Matchmaking in Data-Mining Multi-Agent Systems

Ondřej Kazík[1] and Roman Neruda[2]

[1] Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, Prague, Czech Republic
kazik.ondrej@gmail.com
[2] Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 2, Prague, Czech Republic
roman@cs.cas.cz

Abstract. We present an application of concepts of agent, role and group to the hybrid intelligence data-mining tasks. The computational MAS model is formalized in axioms of description logic. Two key functionalities — matchmaking and correctness verification in the MAS — are provided by the role model together with reasoning techniques which are embodied in specific ontology agent. Apart from a simple computational MAS scenario, other configurations such as pre-processing, meta-learning, or ensemble methods are dealt with.

Keywords: MAS, role-based models, data-mining, computational intelligence, description logic, matchmaking, closed-world assumption.

1 Introduction

An agent is a computer system situated in an environment that is capable of autonomous action in this environment in order to meet its design objectives [23]. Its important features are adaptivity to changes in the environment and collaboration with other agents. Interacting agents join in more complex societies, *multi-agent systems* (MAS).

The effort to reuse MAS patterns brings the need of separation of the interaction logic from the inner algorithmic logic of an agent. There are several approaches providing such separation and modeling a MAS from the organizational perspective, such as the *tuple-spaces*, *group computation*, *activity theory* or *roles* [6]. The Gaia methodology [25] fully exploits roles only in the analysis phase and leaves them during the design phase of development. The BRAIN framework [7]describes roles by means of XML-files and offers also the implementation support in JAVA language. The ALAADIN framework [10] is a organization-centered generic meta-model of multi-agent systems. It defines a general conceptual structure which is utilized in the MAS development. The framework describes MAS from an organizational perspective, instead of using terms of agents' mental states (agent-centered). This model (also called AGR) focuses on three basic concepts: agent, group and role.

L. Cao et al.: ADMI 2012, LNAI 7607, pp. 22–35, 2013.

Generally speaking, a role is an abstract representation of stereotypical behavior common to different classes of agents. Moreover, it serves as an interface, through which agents perceive their execution environment and affect this environment. Such a representation contains a set of actions, *capabilities*, which an associated agent may utilize to achieve its goals. On the other hand, the role defines constraints, which a requesting agent has to satisfy to obtain the role, as well as *responsibilities* for which the agent playing this role holds accountable. The role also serves as a mean of definition of *protocols*, common interactions between agents. An agent may handle more roles, and a role can be embodied by different classes of agents. Moreover, agents can change their roles dynamically. A group is a set of agents and its structure is defined by means of roles and protocols allowed in the group.

The role-based solutions may be independent of a particular situation in a system. This allows designing an overall organization of multi-agent systems, represented by roles and their interactions, separately from the algorithmic issues of agents, and to reuse the solutions from different application contexts. The coordination of agents is based on local conditions, namely the positions of an agent playing the role, thus even a large MAS can be built out of simple organizational structures in a modular way.

In order to automatize the composition of MAS, its formal model in description logic (DL) was introduced [16]. We are employing the concepts of role and group and transform the role model in axioms of DL [17]. In this paper, the necessity of axiom definition both under open- and closed-world assumption is highlighted and the model is extended by integrity constraints.

The main contribution of this paper is the unified formal model both for analysis and run-time support of the MAS which utilizes methods of automated reasoning. This formal description allows dynamic finding of suitable agents and groups (matchmaking), verification of correctness of MAS (system checking) or automated creation of MAS according to the task.

The *computational multi-agent systems*, i.e. application of agent technologies in the field of hybrid intelligence, showed to be promising by its configuration flexibility and capability of parallel computation, e.g. in [9]. We present a role-based model of complex data-mining scenarios, such as meta-learning, parallel computing, ensemble methods or pre-processing of data. The no-free-lunch theorem [24] shows that there is no best approach for every task. In practice the user does not know which method to use and how to set its parameters. Agent-based solution enables automatic assembly of the system. The previous experience can be stored for later experiments on similar data.

In the next section, we present a computational intelligence scenario, perform its analysis, and elaborate the role-based model of a computational MAS. In section 3, the model is formalized by means of description logic axioms. In section 4, the implementation of ontology agent managing the dynamic role-based of MAS, atomic actions and matchmaking queries, improving agents' sociability, are described. Section 5 concludes the paper and shows future work.

2 Role Model of Computational MAS Scenario

Hybrid models including combinations of artificial intelligence methods, such as neural networks, genetic algorithms, and fuzzy logic controllers, can be seen as complex systems with a large number of components and computational methods, and with potentially unpredictable interactions between these parts. These approaches have demonstrated better performance over individual methods in many real-world tasks [5]. The disadvantages are their bigger complexity and the need to manually set them up and tune various parameters.

There are various software packages that provide collection of individual computational methods, e.g. Matlab [12] or R Project [22] with focus on unified environment and computational efficiency. However, these frameworks are closed, they are inflexible in replacement of methods and they do not allow automated construction of data-mining processes. Moreover, it impossible to utilize their methods in various contexts such as meta-learning or recommending methods and settings based on previous experience. Other systems, such as the KNIME environment [4] focus on smooth user-friendly interface to complex hybrid models by offering visual assembly of data-mining processes.

Multi-agent systems seem to be a suitable solution to manage the complexity and dynamics of hybrid systems. In our approach, a computational MAS contains one or more computational agents, i.e. highly encapsulated objects embodying a particular computational intelligence method and collaborating with other autonomous agents to fulfill its goals. Several models of development of hybrid intelligent systems by means of MAS have been proposed, e.g. [15], [1] and [8].

In order to analyze computational scenarios and to construct a model of general computational MAS, we are exploiting the conceptual framework of the AGR model [10]. The simplest possible configuration is a task manager which controls model creation and data processing by the computational method — e.g. neural network — whose data are provided by the data source agent. The computational methods corresponds to physical implementation of agents employing the JADE agent platform and Weka data mining library [16]. This elementary computational scenario, where the computation consists of single method, is not sufficient for most cases we are interested in throughout this paper, but it is always contained in different contexts. Let us consider the following data-mining tasks:

- *Decentralized processing:* more computational agents controlled by single task manager, e.g. various ensemble methods or distributed execution of computational methods.
- *Supplementary learning:* separate optimization algorithms in the search space of computational methods' inner variables, e.g. optimization of neural network's weights by means of evolutionary algorithm.
- *Meta-learning scenario:* optimization in search space of method options [21].
- *Pre-processing and post-processing:* e.g. feature extraction, missing values and outlier filtering, or resampling etc. [11].

All these scenarios contain the coordination structures of computational agent which is controlled by a task manager. Also, the data are provided in all scenarios from a mediator, a pre-processing agent or final data source. In general, the machine learning of data-mining methods (e.g. a back-propagation trained perceptron network) can be seen as a search problem, either implicit — incorporated in the computational agent — or as an external search agent. The meta-learning scenario solves the optimization search of the method's options, thus the simple learning case is implicitly included there as an iterated subtask.

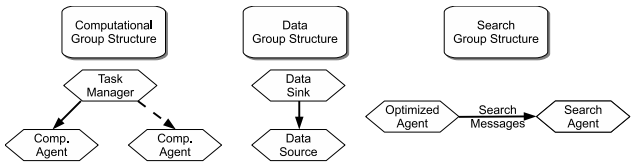

Fig. 1. The organizational structure diagram of the computational MAS

Three general subproblems are considered here: control of computational methods, data provision, and optimization search. Thus the decomposition results — according to the AGR model briefly described in section 1 — in a role *organizational structure* shown at Figure 1. It consists of possible groups, their structures, described by means of admissible roles and interactions between them. This organizational structure contains the following group structures:

- *Computational Group Structure*. It contains two roles: a Task Manager and Computational Agent implementing a computational method. The agent playing the Task Manager role can control more Computational Agents.
- *Search Group Structure* consisting of two roles: a Search Agent and Optimized Agent. It is representing search problem in a general search space.
- *Data Group Structure* contains a Data Sink and Data Source which provides it with data, e.g. training and testing data for computational agent.

The role model allows simplifying the construction of more complex computational multi-agent systems by its decomposition to the simple group structures and roles, which the agents are assigned to. Moreover, the position of an agent in a MAS in every moment of the run-time is defined by its roles without need to take account of its internal architecture or concrete methods it implements. It also reduces a space of possible responding agent when interactions are established, and the model will be used for matchmaking.

3 Description Logic Model of Computational MAS

The family of *Description Logic* (DL), fragment of first-order logic, is nowadays a de facto standard for ontology description language for formal reasoning [2]. In DL, a knowledge base is divided into a T-Box (terminological box), which

contains expressions describing concept hierarchy, and an A-Box (assertional box) containing ground sentences.

Web Ontology Language (OWL), an expressive knowledge representation language, is based on description logic [20]. Semantics of OWL is designed for scenarios where the complete information cannot be assumed, thus it adopts the *Open World Assumption* (OWA). According to the OWA, a statement cannot be inferred to be false only on the basis of a failure to prove it. If the complete knowledge is assumed, the T-Box axioms cannot be used as *Integrity Constraints* (ICs) which would test validity of the knowledge base under OWA. In order to check integrity constraints, the *Closed World Assumption* (CWA) is necessary. There are several approaches simulating the CWA by different formalisms, e.g. rules or queries [20].

We continue in the effort to describe the computational MAS in the description logic model [16]. Our model would incorporate the concepts of group and role. In paper [17], we have elaborated basic role-based model of computational MAS in description logic under OWA. Limitation of standard OWL interpretation under OWA often leads to extension of description logic by other formalism (such as SWRL-rules [14]). We chose the solution where CWA axioms as integrity constraints are expressed in the same OWL language [20] since it preserves the model homogeneity. The authors presented an IC validation solution reducing the IC validation problem to SPARQL query [18] answering. Moreover, they introduced a prototype IC validator extending Pellet [19], the OWL reasoner. For example, the constraint that every service is provided an agent:

$$Service \sqsubseteq \exists isProvidedBy.Agent$$

would not be violated if there is defined a service without agent in an A-Box. The SPARQL representation of this IC would be the following query:

```
ASK WHERE {
  ?x rdf:type Service.
  OPTIONAL {
    ?x isProvidedBy ?y.
    ?y rdf:type Agent.
  }
  FILTER(!BOUND(?y))
}
```

Thus we divided the T-Box of the proposed model into two parts. The first part contains axioms describing mainly the concept hierarchy and the necessary relations between their instances. This schema is interpreted under the OWA and defines the facts the reasoner will infer from the given A-Box. In the second part, there are constraints which define the integrity conditions of the system related mainly to the capabilities of agents. These are interpreted under the CWA. Axioms of T-Box are distinguished in the following text by a subscript of the inclusion axiom symbol. A standard schema axiom interpreted under the OWA is in the form $C \sqsubseteq_O E_1$. An integrity constraint under the CWA has the

form $C \sqsubseteq_C E_2$. The time-dependent information, the current state of the system, is in an A-Box of the ontology.

In the following, we describe a definition of the generic AGR concepts in DL. As we have already mentioned, a role is defined as a set of capabilities, i.e. actions (interactions) an agent assuming this role can use, and a set of responsibilities or events the agent should handle. A group is then described by a set of roles the group contains. A hierarchy of concepts should respect this. In the model, the running agents, groups and initiators are represented as individuals in A-Box. Their roles (i.e. sets of agents with common interface) and group-types are classes defined in T-Box statically.

The T-Box contains the following superior concepts (see Figure 2 left):

- *Responder* is a responsibility of a role. It stands for a type of interaction protocol the agent can handle.
- *Initiator* represents an action from a capability set, and it is coupled to a particular *Responder*. The functional role $isInitiatorOf$ relates to the agent which the action uses. The role $sendsTo$ contains the responding agents to which the action is connected.
- *RequestInit* is a subclass of the previous concept which defines only those initiators that send messages to one agent (unlike e.g. the contract net protocol).
- *Agent* is a class of all running agents and it is a superclass of all agents' roles. The role-agent assignment is achieved simply by a concept assertion of the agent individual in A-Box. The inverse functional roles $hasInitiator$ (inverse of $isInitiatorOf$) and *hasResponder* couple an agent with particular actions and responsibilities. While the *hasResponder* relation is a fixed property, the $hasInitiator$ occurs only when a corresponding connection is established. Finally, the functional role $isMemberOf$ indicates belonging to a group.
- *Group* concept represents a group in a MAS. It has only an inverse of the $memberOf$ role, called *hasAgent*. This relation has two subroles: functional *hasOwner* indicating the member agent which created the group and $rankAndFile$ of all other member agents.

Now we begin to implement our domain specific groups and roles for our domain — computational MAS. The *computational group structure* contains agents with assigned roles of a task manager and one or more computational agent. Between the task manager and each computational agent a control interaction exists.

The sending of control messages between the task manager ($TaskManager$), which initiates this connection, and the computational agent ($CompAgent$) which performs the computational method is modeled by two concepts, an initiator ($ControlMsgInit$) and a responder ($ControlMsgResp$). The initiator of this connection is an instance of $ControlMsgInit$ which is a subclass of the $Initiator$ class. It sends messages only to an agent with a running responder handling these messages, and it is coupled with a Task Manager role as a capability. The schema file of the ontology contains axioms of the initiator concept hierarchy:

$$ControlMsgInit \sqsubseteq_O Initiator$$

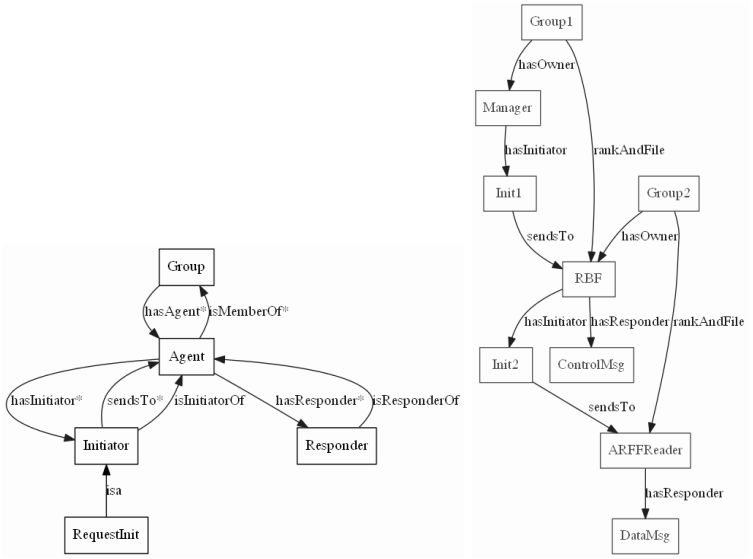

Fig. 2. *Left:* Superior concepts and their relations in T-Box of the role model. *Right:* A-Box state of the simple computational MAS configuration with computational and data group.

The control message responder is a simple descendant of the *Responder* concept and this class contains the instance $ControlMsg$. The schema axioms follow:

$$ControlMsgResp \sqsubseteq_O Responder$$
$$ControlMsgResp(ControlMsg)$$

The following integrity constraints for this concept check the roles of initiating and responding agents:

$$ControlMsgInit \sqsubseteq_C \forall sendsTo.\exists hasResponder.ControlMsgResp$$
$$\sqcap \ \forall isInitiatorOf.TaskManager$$

Role definitions are descendants of the *Agent* concept and have to contain their responsibilities, i.e. responders (capabilities are defined on the initiator side). The responsibility of the computational agent (*CompAgent*) is to respond on the control connections. These are axioms inserted in the schema set:

$$CompAgent \sqsubseteq_O Agent \sqcap \ni hasResponder.ControlMsg$$

The task manager ($TaskManager$) role only sends messages in a group:

$$TaskManager \sqsubseteq_O Agent$$

Finally, the computational group (*CompGroup*) contains only the agents which have asserted that they have the computational agent, task manager or data source role. The subclass-axiom is important for open world reasoning:

$$CompGroup \sqsubseteq_O Group$$

On the other hand, admission (as an owner or general member) of the agent with a wrong role has to be checked by the following closed world constraint:

$$\begin{aligned} CompGroup \sqsubseteq_C & \forall hasAgent.(CompAgent \sqcup TaskManager) \\ \sqcap & \ \forall hasOwner.TaskManager \\ \sqcap & \ \forall rankAndFile.CompAgent \end{aligned}$$

The *data group structure* consists of two roles: the data source owning the group, and the data sink. Between the agents with these two roles is interaction providing data. The data sink requests for certain data the data source which sends them back. The following axioms for these concepts are similar to those for the computational group:

$$\begin{aligned} DataMsgInit \sqsubseteq_O & \ RequestInit \\ DataMsgInit \sqsubseteq_C & \ \forall sendsTo.\exists hasResponder.DataMsgResp \\ \sqcap & \ \forall isInitiatorOf.DataSink \\ DataMsgResp \sqsubseteq_O & \ Responder \\ & \ DataMsgResp(DataMsg) \\ DataSource \sqsubseteq_O & \ Agent \sqcap \ni hasResponder.DataMsg \\ DataSink \sqsubseteq_O & \ Agent \\ DataGroup \sqsubseteq_C & \ \forall hasAgent.(DataSource \sqcup DataSink) \\ \sqcap & \ \forall hasOwner.DataSource \\ \sqcap & \ \forall rankAndFile.DataSink \end{aligned}$$

The elementary computational scenario fits these two group structures. At Figure 2 right, there is A-Box state of the DL model. It contains two group individuals, *Group*1 which is instance of *CompGroup* and *Group*2 with type *DataGroup*. The depicted MAS consists of the following three agents: the Task Manager *Manager*, Data Source *ARFFReader*, and *RBF* agent implementing RBF Neural Network which has two roles: Computational Agent and Data Sink. The individuals *Init*1 and *Init*2 are two initiators realising the control and data connections.

The task manager is also able to control more computational agents in parallel and to collect their results. This configuration is shown at Figure 3 left in simplified cheeseboard diagram [10].

The *pre-processing agent*, i.e. encapsulation of a pre-processing method, obtains data from a data source and provides pre-processed data to other agents. The options of the pre-processing method and source-file have to be set by a

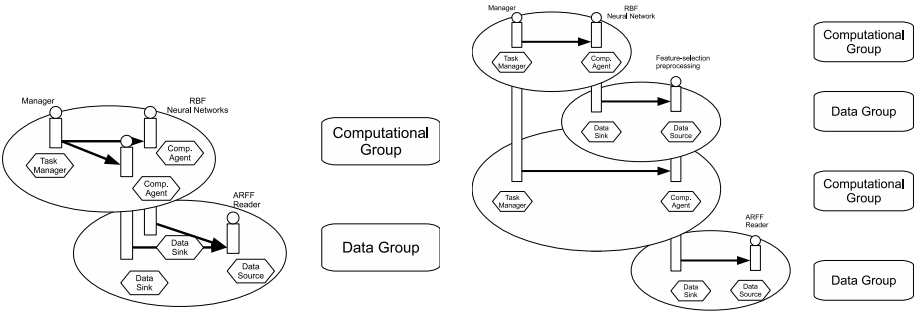

Fig. 3. *Left:* Example of computational MAS configuration with two computational agents controlled by single manager, which process the same data in parallel. *Right:* Example of computational MAS configuration with a pre-processing agent.

task manager who controls the computation. The pre-processing agent gains properties of both the data source (it provides data) and computational agent (it receives data from another source and waits for control messages). Thus the role of *PreprocessingAgent* is defined as an intersection of *DataSource* and *CompAgent*:

$$PreprocessingAgent \sqsubseteq_O CompAgent \sqcap DataSource$$

According to this definition, the pre-processing agent with this role is able to be controlled by a task manager in its own computational group and provide the processed data to another a data sink (e.g. computational agent) as a data source in data group. It also includes the possibility of creation of chain of agents, where on the one end is an agent providing original data table and on the other is a data mining computational method. Diagram of such a configuration with a pre-processing agent is at Figure 3 right.

The *search group structure* is defined in a similar way by the following schema and integrity rules:

$$\begin{aligned}
SearchMsgResp &\sqsubseteq_O Responder \\
&\quad SearchMsgResp(SearchMsg) \\
SearchMsgInit &\sqsubseteq_O RequestInit \\
SearchMsgInit &\sqsubseteq_C \forall sendsTo.\exists hasResponder.SearchMsgResp \\
&\quad \sqcap \ \forall isInitiatorOf.OptimizedAgent \\
OptimizedAgent &\sqsubseteq_O Agent \\
SearchAgent &\sqsubseteq_O Agent \sqcap \ni hasResponder.SearchMsg \\
SearchGroup &\sqsubseteq_O Group \\
SearchGroup &\sqsubseteq_C \forall hasAgent.(OptimizedAgent \sqcup SearchAgent) \\
&\quad \sqcap \ \forall hasOwner.OptimizedAgent \\
&\quad \sqcap \ \forall rankAndFile.SearchAgent
\end{aligned}$$

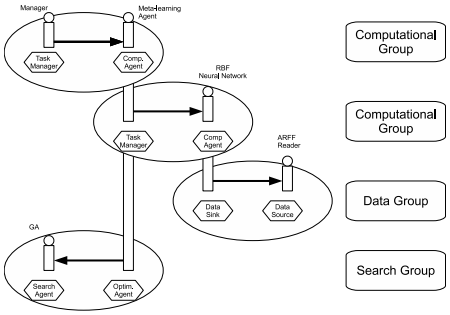

Fig. 4. The configuration of parameter-space search

This structure has various ways how it can be utilized. It can be used for optimization of computational agent's inner variables but also for meta-learning, i.e. finding of optimal options of certain computational method. In this case the agent does not control directly computational methods, but a meta-learning agent whose goal is finding the options with lowest resulting error rate. The meta-learning agent is optimized by a search method, and in the same time it controls the CAs representing the computational method. Such a configuration is shown at Figure 4. Here the options of the RBF neural network on certain data are optimized with Genetic Algorithm.

4 Management of the Model

To coordinate the run-time role organization of a MAS built according to the schemas and constraints of T-Box, it is necessary to have a central authority, separate agent in which the DL model is represented. Other agents will change the state of the model and query it by interaction with this agent.

The model is implemented as an *ontology agent* (OA) in JADE, Java-based framework for a MAS [3]. The goals of the OA are:

- Keeping track of the current state of MAS. Agents present in the MAS register themselves in the OA, state changes of their roles, create and destroy groups and their membership in them, and establish communication channels. The OA infers the necessary facts by means of OWA reasoning. The atomic actions are presented in subsection 4.1.
- Verification of correctness of MAS. The OA controls all changes of the system and does not allow activities which would violate the integrity constraints. The integrity constraints are handled by OA in CWA mode, as discussed in section 3.
- Matchmaking of agents and groups. When exploiting the concept hierarchy, it is possible to search groups of certain types or agents which have a certain role, which are members of certain group or that can handle certain types of messages etc. For the matchmaking queries, see 4.2.

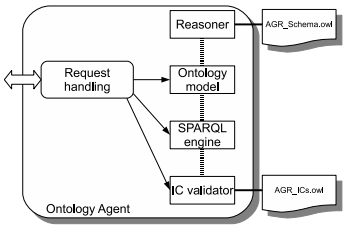

Fig. 5. Architecture of the ontology agent

The ontology agent (shown in Figure 5) consists of the *request handling* module which is responsible for processing of incoming requests and replying. It employs the ontology functions provided by the Pellet OWL-DL reasoner [19] and its extensions. The *ontology model* contains an assertional box of the ontology and describes the current state of the system. The open-world *reasoner* infers new facts from axioms in the OWL schema file and content of the A-Box. The integrity constraints saved in a separate OWL file are converted into SPARQL queries and run by the *SPARQL engine* on the ontology model. The SPARQL engine is also used to answer matchmaking queries.

The *communication ontology* for contents of OA messages has been created. This ontology consists of three types of concepts: *actions* changing state of the model, *matchmaking queries* and concepts informing about *results* of these requests.

4.1 Actions in the Model

During their life-time, the agents in the system will change their roles as well as position in the system. In order to affect the state of the role-based model, it uses several atomic actions which are sent to the ontology model. These actions result in changing of assertions in the A-Box of the model and are validated by the integrity constraints. If any of the ICs is violated, the change is not performed. In this case the action ends by failure.

The following types of actions are allowed in the MAS model:

- `RegisterAgent` — creates a new individual of type *Agent* and returns its name.
- `DeregisterAgent` — removes the individual which corresponds to the agent from the model.
- `SetRole(role)` — adds declaration of agent's role in the A-Box. E.g. if the agent a sets its role to computational agent, the OA adds in the A-Box the assertion $CompAgent(a)$.
- `RemoveRole(role)` — removes the role declaration from the A-Box.
- `CreateGroup(grpType)` — creates new individual defining group and declare it to type `grpType`. The relation *hasOwner* is created between the creating agent and the new group. Successful action returns name of the new group individual.

- `DestroyGroup(grpName)` — removes the individual `grpName` and all its assertions (including agents' membership).
- `EnterGroup(grpName)` — adds the relation *rankAndFile* between group `grpName` and the requesting agent. This change of state can be in contradiction with allowed roles in the group.
- `LeaveGroup(groupName)` — removes all relations *hasAgent*, *hasOwner* and *rankAndFile* between the agent and the group.
- `CreateInitiator(initType)` — creates an individual defining communication initiator. The concept assertion of this individual with class `initType` and role assertion *hasInitiator* between the agent and initiator individuals are added. The successful action returns name of the initiator. The action can contradict the assumptions about associations of roles and initiators' types.
- `DestroyInitiator(initName)` — removes the individual `initName` with all its assertions.
- `CreateConnections(initName, receivers)` — has as a result addition of sequence of *sendsTo* role assertions between initiator `initName` and the agents in the `receivers`. The action can be refused if the initiator type does not match with responders of any agent.
- `DestroyConnections(initName, receivers)` — removes all such assertions.

4.2 Matchmaking of Agents and Groups

The query engine of the model containing current MAS state is employed as a service for agents, which want to find suitable partners in the system for collaboration. The agents are also able to relate to the system by means of social concepts, e.g. groups or roles. The matchmaking requests of agents or groups are transformed by the OA into SPARQL queries [20] and executed on the inferred model.

The concept `getAgent(grpName, responderTypes, role)` specifies properties of demanded agent, i.e. the group it should be member of, list of communication types it should handle, and its role. For example, the matchmaking problem of finding of computing agent (i.e. has a type of *CompAgent*), which is member of group `g`, is converted to the following SPARQL query over the inferred A-Box of the model:

```
SELECT ?Agent WHERE {
  ?Agent rdf:type CompAgent.
  ?Agent isMemberOf g.
}
```

The ontology agent sends back in the inform message the list of agents matching with these properties.

Likewise, the request `getGroup(grpType)` returns list of groups in the MAS with specified type `grpType`.

5 Conclusions

In order to support the real-world data-mining processes, the models of computational group, data group, search group, and pre-processing agent have been included. The computational agent can stand for whole computational group and realize various ensemble methods. The meta-learning scenario as well as external learning can be described and implemented in this model by means of general search schema. The proposed model of pre-processing also allows defining chains of pre-processing agents gradually solving the input data inconsistencies.

The ontology agent representing the model of current MAS state has been implemented. The ontology agent allows general management, correctness verification and matchmaking of the MAS with concepts of agents, roles and groups. For this purpose, reasoning and querying of the DL model is employed.

Both the deduction axioms and integrity constraints are defined in the same formalism of OWL-DL with distinction of open-world and closed-world assumptions. So far, some of these ideas have been incorporated into an existing agent-oriented data-mining system [13] with promising results.

In comparison with other role based models, the proposed model supports analysis, design and implementation phase of development. Moreover it defines it in a formal model of the OWL-DL standard. This allows including of automated reasoning methods, and utilizing them in run-time management.

Further research will be put in ontology classification of computational methods, their parameters and input data. This model will broaden the possibilities of the model to express the computational MAS dynamics. This unified model will support the construction of computational MAS according to a demanded task, finding suitable methods and agents in the system, and possibly its visualization.

Acknowledgments. Ondřej Kazík has been supported by the Charles University Grant Agency project no. 629612 and by the SVV project no. 265314. Roman Neruda has been supported by the Czech Science Foundation project no. P202/11/1368.

References

1. Albashiri, K.A., Coenen, F.: Agent-Enriched Data Mining Using an Extendable Framework. In: Cao, L., Gorodetsky, V., Liu, J., Weiss, G., Yu, P.S. (eds.) ADMI 2009. LNCS, vol. 5680, pp. 53–68. Springer, Heidelberg (2009)
2. Baader, F., et al.: The description logic handbook: Theory, implementation, and applications. Cambridge University Press (2003)
3. Bellifemine, F., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. John Wiley and Sons (2007)
4. Berthold, M.R., et al.: KNIME: The konstanz information miner. In: Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 319–326. Springer (2008)
5. Bonissone, P.: Soft computing: the convergence of emerging reasoning technologies. Soft Computing - A Fusion of Foundations, Methodologies and Applications, pp. 6–18 (1997)

6. Cabri, G., Ferrari, L., Leonardi, L.: Agent role-based collaboration and coordination: a survey about existing approaches. In: Proc. of the Man and Cybernetics Conf. (2004)
7. Cabri, G., Ferrari, L., Leonardi, L.: Supporting the Development of Multi-agent Interactions Via Roles. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, pp. 154–166. Springer, Heidelberg (2006)
8. Cao, L.: Data Mining and Multi-agent Integration. Springer (2009)
9. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent mining: The synergy of agents and data mining. IEEE Intelligent Systems 24, 64–72 (2009)
10. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: An Organizational View of Multi-agent Systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
11. Gibert, K., et al.: On the role of pre and post-processing in environmental data mining. In: International Congress on Environmental Modelling and Software – 4th Biennial Meeting, pp. 1937–1958 (2008)
12. Gilat, A.: MATLAB: An Introduction with Applications, 2nd edn. John Wiley and Sons (2004)
13. Kazík, O., Pešková, K., Pilát, M., Neruda, R.: Implementation of parameter space search for meta learning in a data-mining multi-agent system. In: ICMLA, vol. 2, pp. 366–369. IEEE Computer Society (2011)
14. Martin, D., Paolucci, M., McIlraith, S.A., Burstein, M., McDermott, D., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing Semantics to Web Services: The OWL-S Approach. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005)
15. Neruda, R.: Emerging Hybrid Computational Models. In: Huang, D.-S., Li, K., Irwin, G.W. (eds.) ICIC 2006. LNCS (LNAI), vol. 4114, pp. 379–389. Springer, Heidelberg (2006)
16. Neruda, R., Beuster, G.: Toward dynamic generation of computational agents by means of logical descriptions. International Transactions on Systems Science and Applications, 139–144 (2008)
17. Neruda, R., Kazík, O.: Role-based design of computational intelligence multi-agent system. In: MEDES 2010, pp. 95–101 (2010)
18. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Tech. rep., W3C (2006)
19. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)
20. Sirin, E., Tao, J.: Towards integrity constraints in OWL. In: OWLED. CEUR Workshop Proceedings, vol. 529 (2009)
21. Soares, C., Brazdil, P.B.: Zoomed Ranking: Selection of Classification Algorithms Based on Relevant Performance Information. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 126–135. Springer, Heidelberg (2000)
22. Teetor, P.: R Cookbook. O'Reilly (2011)
23. Weiss, G. (ed.): Multiagent Systems. MIT Press (1999)
24. Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. Tech. rep., Santa Fe Institute (1995)
25. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000)