

Learning Mechanism for Distributed Default Logic Based MAS – Implementation Considerations*

Dominik Ryzko, Henryk Rybinski, and Przemyslaw Wiech

Institute of Computer Science, Warsaw University of Technology

Abstract

The paper presents learning mechanisms for MAS based on Distributed Default Logic (DDL), the formalism for multi-agent knowledge representation and reasoning. In the distributed environment learning processes provide measures to order default rules, which gives the agent better use of its local and external knowledge. Such mechanisms allow the system to work effectively in a changing environment, where basic facts and sources of information are uncertain.

Keywords: Artificial Intelligence; Intelligent Agents; Logic Programming; Distributed Default Logic; Machine Learning

1 Introduction

In real world many tasks require knowledge, which is located, at least partly, outside of the entity assigned to solve them. Such entities must have an efficient way of cooperating in order to reach solutions of the problems presented to them. Multi-agent systems (MAS) bring tools for modeling such situations using the concept of a set of cooperating autonomous, intelligent and proactive agents.

The need for knowledge sharing in a distributed environment is an obvious issue. Recently a new branch of AI called Agent-Mediated Knowledge Management has emerged as a result of research in the area of Knowledge Management in the context of multi-agent systems. The field shows a shift of interest from traditional knowledge management to collaboration of various, often heterogeneous, knowledge sources (Dignum, 2003).

An example of a similar approach is shown in Poggi (2002), where a multi-agent system for Corporate Memory Management (CoMMa) is introduced. The system facilitates one creating and reusing knowledge in an organization by utilizing several AI techniques from agent technology to machine learning. In Mercer (2002) MAS is proposed for knowledge sharing in a system designed for good programming practice advice. The system is composed of agents representing various programmers in the team. This allows each of the nodes to adapt to individual preferences of a particular programmer.

This paper contributes to this area, but we propose using default logic (DL) as a framework for knowledge representation and sharing. The paper is a continuation

*The work has been granted by Polish Ministry of Education (grant No 3T11C 002 29)

of Ryzko (2003), Ryzko (2004), and Ryzko, Rybinski (2006) by considering learning requirements for DL based MAS. We present here the algorithms for knowledge sharing and learning in Distributed Default Logic framework (DDL) (Ryzko, Rybinski, 2006). We also focus on learning mechanisms in the DDL. Additionally, implementation considerations are provided with architecture of environment for building MAS based on DDL, as a knowledge modeling tool.

2 Related Work

Many multi-agent systems adopt logical programming for knowledge representation. Kowalski and Sadri (1999) show an extension of logic programming (LP) for modelling multi-agent systems. They introduced object-oriented benefits to the LP by embedding a standard proof procedure within the agent cycle. The approach is shown to be an extension of BDI and Shoham's Agent0 architectures.

Reasoning in a logical system is always a complex task. The study of efficiency in logical systems has led to the notion of partitioning of logical theories. This partitioning based on some detectable structure can simplify the process of reasoning by performing local inference and passing the results between partitions. Amir and McIlraith (2005) show how a reasoning tree can be decomposed in the case of first-order and prepositional theories. They show a sound and complete forward and backward message passing algorithms for such partitioning.

In Ryzko and Rybinski (2006) our main goal was to provide a framework for distributed default reasoning. The formalism used there is an extension of Reiter's Default Logic (Reiter, 1980). In particular we have defined a notion of Distributed Default Theory (DDT). We also adopted the notion of the stratification of default theories, which has been studied by Cholewinski (1993) for semi normal theories, and analyzed partitioning of DDTs in MAS.

The fundamental problem in using default logic is that there are possibly many extensions that can be generated from a particular default theory. Several papers have been published to deal with this problem, leading to modifications of the original formalism, for example justified extensions (Lukaszewicz, 1988) or stationary default extensions (Przymusinska, 1994). In Ryzko and Rybinski (2006) we have proposed a novel approach, which uses priorities based on a learned confidence function. This allows agents to adapt to a changing environment, as the reliability of the information sources can improve or deteriorate in time.

A concept of calculating priorities based on trust has been recently also proposed in Katz (2006). Trust values are derived there from web-based social networks. In our approach priority measure is calculated dynamically during the reasoning process and is influenced by other measurable properties of the agents. It is integrated with the distributed reasoning in the DDL framework. This allows generating appropriate extensions for particular requests and controlling constraints imposed by environment or user, like time of response and/or cost.

We focus in this paper on learning capabilities of agents in MAS. We presume that in the changing environment agents can learn priorities during operation. We analyze a combination of learning methods suitable for this case. Some preliminary experimental results are presented along with architecture of the DDL based MAS.

3 Distributed Default Logic (DLL)

We define MAS as a set of agents in the following way:

Definition 3.1: A *Multi-Agent System (MAS)* is a collection of agents A_1 through A_n operating in the same environment:

$$MAS = \{A_1, A_2, \dots, A_n\}$$

Agents can interact with the environment and with each other using interfaces, according to predefined communication protocols and languages. It is assumed the system is asynchronous which means that agents can perform their actions at any time as long as they are valid in the current MAS state.

A default theory is described as a pair (D, W) , where D represents a set of default rules while W is a set of first-order formulas.

Definition 3.2: A *default rule* d is a rule of the form

$$\frac{\alpha(x) \mid \beta_1(x), \dots, \beta_m(x)}{\omega(x)}$$

where $\alpha(x), \beta_1(x), \dots, \beta_m(x)$ and $\omega(x)$ are all classical logical forms. $\alpha(x)$ is called prerequisite (denoted by $p(d)$); $\beta_1(x), \dots, \beta_m(x)$ is justification (denoted by $j(d)$); and $\omega(x)$ is consequent ($c(d)$). If $p(d)$ is known and $j(d)$ is consistent with W then $c(d)$ may be inferred.

Stratification of a default theory is based on finding an ordering of default rules that would allow more efficient reasoning (Cholewinski, 1995). Each default rule d is assigned to a strata $str(d)$, which holds the following properties:

$$\begin{aligned} str(d_1) = str(d_2) &\Leftrightarrow \exists a \ a \in c(d_1) \wedge a \in c(d_2) \\ a \in c(d_1) \wedge a \in j(d_2) &\Rightarrow str(d_1) < str(d_2) \end{aligned}$$

where a is a symbol of a predicate and $c(d)$ is the consequent of a rule d and $j(d)$ is its justification. Such an ordering can improve efficiency of the task of finding extensions for some classes of default theories (Cholewinski, 1993).

Now we will show algorithms for efficient reasoning in partitioned default theories. We will start by showing an ideal partition, which can be used, if we want to split logical theory between the agents in a multi-agent system; then we will show how to act if we are faced with a partitioning which can contain overlapping data and random placement of rules in the system. Basically, we will try to improve the quality of such partitioning. We will also show how the algorithm can be used by agents to reason in a multi-agent environment. Below new notions regarding distributed default logic will be defined. We start with extending the original Reiter's formalism of defaults to contain special metadata.

Definition 3.3: *Extended default template* T is a rule in the form

$$\frac{\alpha(x) : \beta_1(x)_{1}^{L_1}, \dots, \beta_m(x)_{m}^{L_m}}{\omega(x)}$$

where $L_k \subseteq MAS$ and α, β, ω have the same meaning as in standard default rules.

Definition 3.4: *Extended default rule* is a following rule

$$\frac{\alpha(x) : \beta_1(x)_{1}^{L_1}, \dots, \beta_m(x)_{m}^{L_m}}{\omega(x)}$$

where $l_k \in L_k$ and α, β, ω have the same meaning as in standard default rules.

We can now define the Distributed Default Theory in the following way:

Definition 3.5: *Distributed Default Theory (DDT)* is a set $\{\Delta_1, \Delta_2, \dots, \Delta_n\}$, where index n enumerates nodes of the distributed system and $\Delta_i = (D_i, W_i)$ is a default theory stored at node $i \in \langle 1, n \rangle$. Each of the sets D_i contains extended default templates.

The model proposes a multi-agent system as a collection of agents maintaining an internal logical database, where the domain knowledge and environmental knowledge is distinguished. The domain knowledge contains clauses and rules specific to the agents domain. The environmental one contains information about the system in which agents operate. It includes constraints that the agents have to obey, as well as information about other agents and their properties.

However, as shown below, the domain and environmental knowledge intersect. The agents use environmental information to find out, where they should search for particular domain knowledge. On the other hand, the domain knowledge is written in the form of extended defaults which contain information about agents that possess some unique knowledge required to use the particular rule.

Database architecture. Let us consider an agent maintaining the internal logical database. In the complex environment with many autonomous operating entities, the agent may overlook many facts, which are known by other agents. Additionally, the existing facts may be subject to change without being noticed by everyone. The logical system of an individual agent is a default reasoning system based on local knowledge and extendible by knowledge of external agents.

The logical database of a particular agent will consist of a set of facts about the agent's world, domain knowledge rules, and semantic constraints.

Confidence measure. In MAS, each agent usually has to cooperate with numerous other members of the system. The information received from other entities may have a different, sometimes contradictory, impact on the agent's beliefs. Therefore a measure of confidence concerning other agents is introduced in order to manage incoming information and rely on the most reliable one.

For each agent, the confidence function C_n will be defined $C_n : MAS \rightarrow [0; 1]$. Clearly, for $A_i \in MAS$, we have $C_i(A_i) = 1$. The confidence functions will be further used in the distributed reasoning process (Section 4). C_n is just one of the properties that can be maintained by an agent. Additionally, we can measure response time, price etc. Hence we define a property matrix, which together with the confidence function, provides guidance for choosing other agents to cooperate.

Definition 3.6: For n agents with m properties we define a *property matrix* $V = \{v_{ij}\}$, $1 \leq i \leq n$, $1 \leq j \leq m$, where v_{ij} is a value of property j for agent i . Hence, each column represents one agent, while each row refers to a single property.

The property matrix, like the confidence function, will be maintained by each agent, and special procedures for calculating its values should be provided.

Let A be a distributed default theory divided into n partitions:

$$A = \bigcup_{i=1}^n A_i$$

$\{A_i\}_{i \leq n}$ will be called a *partitioning* of default theory A .

The partitioning can be represented by a labeled directed graph $G = (V, E, l)$, which we call the *intersection graph*. In the graph, each node corresponds to the individual partition A_i . Two nodes i, j are linked by a directed edge leading from i to j if

$$\text{either } \exists d_i \in D_i, d_j \in D_j, p \in L(D, W) : p \in c(D_i) \wedge p \in p(D_j), \\ \text{or } \exists d_j \in D_j, p \in L(D, W) : p \in W_i \wedge p \in p(D_j)$$

The edges are labeled with the set of symbols that A_i and A_j share ($l(i, j) = L(A_i) \cap L(A_j)$). $l(i, j)$ will be called *communication language* between partitions A_i and A_j .

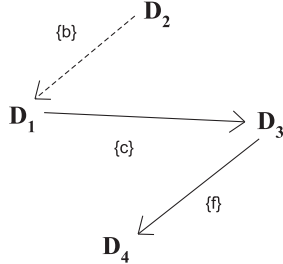
Before being used by the reasoning engine, the extended default templates have to be changed into extended default rules. This process, called materialization, leads to no more than one agent signature attached to each of the justifications.

Definition 3.7: We say that an extended rule r is a *materialization* of T iff

- (i) $p(r) = p(T)$
- (ii) $c(r) = c(T)$
- (iii) $\forall \beta_m^{L_m}(x) \in j(T) \exists \beta_m^{l_m}(x) \in j(r) : l_m \in L_m$

where $p(r), c(r), j(r)$ are prerequisite, consequence and justification of the rule.

Each *template rule* and *extended rule* in D_i can contain references to other partitions in its justification if a particular symbol appears in other partitions signature. As a result in addition to the graph G we will have edges representing these references. These edges will be distinguished from the edges of graph G .



Example: For the Distributed Default Theory having defined

$$D_1 = \{a : b(D2)/c\} W_1 = \{a\}$$

$$D_2 = \{k / - b\} W_2 = \{\}$$

$$D_3 = \{c : e/f\} W_3 = \{e\}$$

$$D_4 = \{f : g/h\} W_4 = \{\}$$

the intersection graph will look like on Fig.1.

FIGURE 1: Intersection graph

Definition 3.8: The intersection graph G will be called *well-ordered* iff:

- (i) G is a tree, therefore there exists such a numbering of nodes that all edges lead from a lower to a higher value of this numbering,
- (ii) all references leading from justification of default templates lead to a node with a higher number,
- (iii) there are no conflicts such that there are two nodes with the same symbols in conclusions of their rules.

The numbering (i) is called *well-ordered numbering*.

The following theorems describe important features of a well-ordered graph.

Theorem 3.1 Let $A = \bigcup_{i=1}^n A_i$ be a partitioned default theory with the intersection graph G . Then, if G is *well-ordered*, A is stratified according to a well-ordered numbering. *Proof:* In Ryzko (2006).

Theorem 3.2 Having a stratified logical theory with well-ordered intersection

graph G , we can find all extensions of theory (D, W) by using Algorithm 3.1 for each node, starting from the node with the lowest rank. *Proof:* In Ryzko (2006).

Algorithm 3.1 Generate extension
 Input: (D_n, W_n)
 Output: Ext (D_n, W_n)
 Begin
 DR = Materialize (D_n)
 //generate rules from templates
 While (extension not ready)
 loop//rule loop
 R = GetRule (DR)
 //Get rule with highest priority
 If $(p(R)$ consistent with $W_n)$
 For $(J$ from $j(R))$ loop
 //precondition loop
 If $(J$ holds)
 Send (agent, J)
 Wait (answer)
 If $(answer \neq TRUE)$
 Continue (while loop)
 End if
 End if
 End loop
 Add (conclusions, W_n)
 Add (conclusions, Result)
 End if
 End loop
 Send (Result, parent agent)
 End

Algorithm 3.2 Justify
 Input: L - list of predicates
 Output: True/False
 Begin
 For (each predicate P in L) loop
 If $(P$ does not hold)
 Return False
 End if
 End loop
 Return True
 End
Algorithm 3.3 Materialization of extended default framework
 Input: EDT - extended default template
 Output: ED - extended default
 Begin
 For (each justification in EDT) loop
 For (each agent on list) loop
 For (each attribute) loop
 If not (conditions satisfied)
 Delete (agent, agent list)
 Continue with agent loop
 End if
 End loop
 End loop
 If $(length$ (agent list) $>1)$
 Choose by confidence
 Else if $(length$ (agent list) $<1)$
 Return null
 End if
 End loop
 Return ET
 End

Algorithm 3.2 is used by external agents to generate justifications passed to them.

As mentioned already, the agent's database may generate many extensions. Approaches proposed in the literature vary in respect of how many extensions should be found, and how many should be accepted. Keeping multiple belief sets implies extra costs for invalidating defaults. On the other hand it guarantees that none of the possible interpretations of default theory is missing. In Ryzko and Rybinski (2006) we presented how we take advantage of the extended defaults formalism. Let us recall now the priority definition:

Definition 3.9: Priority P_d of an extended default $d = \alpha : \beta_1^{A_1}, \dots, \beta_n^{A_n} / \gamma$ is

$$P_d = \prod_{i=1..n} C(A_i)$$

So, the priority of rule d in reasoning depends on confidences of agents to which d refers. As we can see, the materialization process has a great impact on the choice of extension generated by an agent. Algorithm 3.3 calculates the materialization.

At a particular time an agent will have a set of constraints on some of the properties appearing in the agents' matrix. Each constraint will be defined by conditions restricting their values. For each of the justifications, agents which properties values satisfy these conditions should be chosen. If more then one agent satisfies conditions we use the one with the highest confidence.

4 Learning of confidence function

In Section 3 we described the process of Distributed Default Reasoning in MAS. We have shown that the priorities of applying defaults are based on the confidence function and other properties of agents. We have also assumed that agents operate in a changing environment with the possibility of entering and exiting the system by its different members. Hence, agents should be able to learn environmental knowledge. We assume that they learn while operating in MAS, and no teacher or training data is available. This narrows the number of learning methods which can be applied. The only guidance that agents can use in order to improve their performance is success or failure of their actions based on reasoning. This learning model is similar to reinforcement learning (RL). However, here the situation is more complicated, as many agents participate in the distributed reasoning process. We propose a novel approach to RL, in which reward received by externally criticized agent is further on distributed, as an agent becomes a critic itself. This process is continued as the reward is "consumed" when passing it down the chain of agents, and finally is "eaten up" completely. As queries can be entered into the system at various nodes, the roles of critic/criticized can switch from one query to another. For an agent A we can write the following rule of the reward distribution

$$R = R_A + \sum_{n \in \{A1..Ai\}} R_{An}$$

with R being the total reward for the answer to query, equals to the reward consumed by A , plus the sum of the rewards distributed to the other members of the system who participated in the process.

Based on the reward received after a particular distributed reasoning process, an agent modifies its confidence function, so that priorities of default rules can change. Receiving a positive reward means that information provided from other agents (if any) has led to successful reasoning, and the confidence of the given agent for them should increase. Some reward must be also distributed to these agents, according to the equation above, so that they can adjust their confidence function as well. In this way any success/failure leads to changes in the confidence function, which in turn might lead to generation of new extensions based on the properties of DDL. Finally, the overall performance of the system increases in time.

In Ryzko (2004) we have introduced a multi-agent version of Explanation Based Learning called here MEBL. Here we modify the approach. First, let us observe that by establishing a common language between each of the partitions we can describe the state of other agents by checking which symbols are currently valid. Second, rather than storing proof trees, we analyze particular extensions which are generated from internal and external knowledge.

Based on these observations we can describe the state of an agent by memorizing current query and symbols available from other agents. The agent should learn a sequence of actions leading to a desired extension. This process repeated in each node leads to an improvement of the global reasoning of MAS.

Such MEBL learning memorizes the trace for generating appropriate extensions. When a new query arrives the agent will first check for stored traces to speed up the answering process. However, for this learning method the correct answers should be generated beforehand. If distributed rules have wrong priorities

the appropriate extension will never be generated. Therefore MEBL should be supported by RL to learn priorities, as described earlier in this Section.

As it is shown in Dietterich (1997) both Reinforcement Learning and Explanation Based Learning can be unified in the form of Explanation-Based Reinforcement Learning, to benefit from both of these approaches. Also in our research we utilize both approaches to cover regions of reasoning (EBL) while adjusting attributes of other agents which influence extensions to be generated (RL).

5 Preliminary Experimental Results

As priorities are calculated based on the properties of other agents, the agent property matrix is simulated. Also, a special stub has been provided to generate other agent's replies for the default reasoning process.

The system was tested in a simple environment for route planning, represented as a directed graph with nodes representing locations and edges indicating transitions. After each query the agent was given a positive reward for a proper reply, or a negative reward for the wrong one. It was assumed that performing the full reasoning process costs more than using a stored extension.

Among others we performed an experiment for comparing pure EBL learning agent with an agent that uses combination of EBL and RL. Fig. 2 shows, that by the combination of RL and MEBL the agent learns more efficiently and achieves a higher value of total reward.

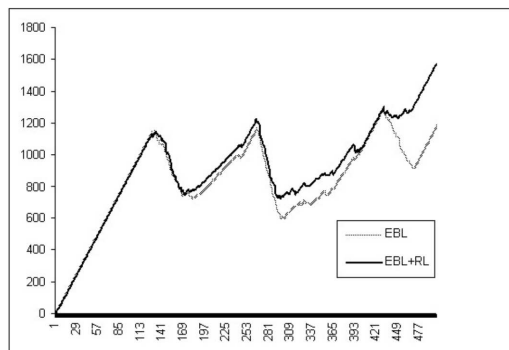


FIGURE 2: Learning quality

For the purpose of demonstration of the theoretical results, a program simulating an agent has been built. The agent maintains internal database in the form of default theory with rule templates. During reasoning templates are dynamically materialized and their priorities are calculated. Then extensions can be generated using rule priorities.

6 Architecture for DDL reasoning agents

The experimental environment consists of two primary elements. First, there is an agent platform, which is used to manage an agent's life cycle, enabling exchange of messages between agents. The agent platform should also provide a white pages service as a means of finding agents. Second, a knowledge base is needed to store domain and environmental knowledge, and perform reasoning. In this work, the knowledge base is our primary focus, however, distributed reasoning requires it to be combined with the agent's ability to interact with other agents.

The current knowledge base implementation is based on an extended version of Prolog. Standard Prolog applies closed world assumption to all predicates. In or-

der to be able to express explicit negative information, apart of negation-as-failure, classical negation had to be introduced to the logic programming language. The classical negation is added according to the extended programming definition (Gelfond, 1991). Thus one of three answers can be obtained: *true*, *false* or *unknown*, for a sentence being true or false, or neither true nor false, respectively.

It is implemented by transforming an extended logic program with classical negation to a general logic program by substituting all negative literals $\neg p(x)$ with positive forms $p'(x)$. Next, the symbol $neg_p(x)$ is created, which is understandable by the Prolog engine. When querying the knowledge base, two queries are executed, namely $p(x)$ and $neg_p(x)$. If only one of them returns *yes*, the answer is *true* or *false*, respectively. If both queries fail, *unknown* is returned. Both answers being positive means that the theory is inconsistent.

As the agent platform we are using JADE (Bellifemine, 2003), which is a mature open-source product. The platform provides the functionality for running a distributed agent environment and for exchanging messages between agents. JADE uses FIPA standard messages, out of which we use only two: *query-if* and *response*. The platform is Java-based, so a method of interacting with Prolog was needed. InterProlog (Celejo, 2004) is a good choice, as it enables two-way communication between Java and Prolog. The library uses XSB Prolog as the underlying implementation. Prolog goals can be executed from a Java program, and the execution of a Prolog program can be intercepted, passing the control back to Java. This feature enables performing arbitrary tasks in the middle of a suspended reasoning process. This is the place where an appropriate agent is chosen for querying based of the confidence function and other agent properties. The result of the query is passed back to the Prolog reasoning process. The interception time can also be used to cache or store responses of other agents, learn and reason about the confidence towards other agents.

7 Conclusions

We have analyzed how learning supports the model for DDL as a tool for building multi-agent systems. Agents can answer queries using their own knowledge, as well as, by sharing it with other members. The domain knowledge is mixed with environmental metadata, which allows locating information in MAS. This is further used together with the property matrix to generate extensions of the default theory maintained by agents and to answer queries. The algorithms for distributed reasoning based on communication with other agents have been presented.

In "real life" situations, agents can be faced with undesired knowledge distribution. In such cases the defined reasoning process is still correct, although some of the extensions may not be generated. In Ryzko and Rybinski (2006) we show how the improvement of knowledge distribution can be performed in such cases.

The proposed approach based on DDL enables metadata to be stored within default templates. This additional information gives guidance on the knowledge distribution in the system. Templates are materialized at runtime into distributed default rules. Another feature is using defaults priorities, calculated from the properties of the other agents, among them the most important is the confidence

measure which has a direct influence on ordering defaults for the reasoning process.

The learning capabilities enable agents to fine tune this process by adjusting priorities of agents and memorizing most useful extensions. We have presented some pre-liminary experimental results and described an environment for implementing full featured DDL agents with learning.

References

- Eyal AMIR and Sheila MCLRAITH (2005), Partition-based logical reasoning for first-order and propositional theories, *Artificial Intelligence*, 162:49–88.
- Fabio BELLIFEMINE and Giovanni CAIRE and Giovanni RIMASSA and Paola TURCI (2003), JADE - a white paper, *Journal of Telecom Italia Lab.*, 3:6–19.
- Miguel CALEJO (2004), InterProlog: towards a declarative embedding of logic programming in Java, in *Proceedings of Logics in Artificial Intelligence 2004*.
- Pawel CHOLEWINSKI (1993), Seminormal stratified default theories, *Technical Report, 238-93, University of Kentucky, Lexington*.
- Pawel CHOLEWINSKI (1995), Reasoning with stratified default theories, in *Proceedings of 3rd Int'l Conf. on Logical Programming and Nonmonotonic Reasoning 1995*.
- Thomas G. DIETTERICH and Nicholas S FLANN (1997), Explanantion-based learning and reinforcement learning: a unified view, *Machine Learning*, 28:169–210.
- Virginia DIGNUM (2003), Using agent societies to support knowledge sharing, in *Proceedings of AAMAS-03 Workshop on Autonomy, Delegation and Control 2003*.
- Michael GELFOND and Vladimir LIFSCHITZ (1991), Classical negation in logic programs and disjunctive databases, *New Generation Computing*, 9:365–385.
- Yarden KATZ and Jennifer GOLBECK (2006), Social network-based trust in prioritized default logic, in *Proceedings of 21st Nat'l Conf. on AI 2006*.
- Robert KOWALSKI and Fariba SADRI (1999), From logic programming to multi-agent systems, *Annals of Mathematics and Artificial Intelligence*, 25:391–419.
- Witold LUKASZEWICZ (1988), Considerations on default logic: an alternative approach, *Computational Intelligence*, 4:1–16.
- Sarah MERCER and Sue GREENWOOD (2002), A multi-agent architecture for knowledge sharing, in *Proceedings of From Agent Theory to Agent Implementation 2002*.
- Agostino POGGI and Giovanni RIMASSA and Paola TURCI (2002), An intranet based multi-agent system for corporate memory management, in *Proceedings of AAMAS 2002*, pp.1039–1040.
- Halina PRZYMUSINSKA and Teodor PRZYMUSINSKI (1994), Stationary default extensions, *Fund. Informaticae*, 21:67–87.
- Raymond REITER (1988), A logic for default reasoning, *Artificial Intelligence*, 13:81–132.
- Dominik RYZKO and Henryk RYBINSKI (2003), Knowledge sharing in default reasoning based multi-agent systems, in *Proceedings of 3rd Int'l Conf. IAT 2003*, pp. 576–579.
- Dominik RYZKO and Henryk RYBINSKI (2004), Exchange of knowledge and learning in default reasoning based agent systems, in *Proceedings of Int'l IIS: IIPWM 2004*.
- Dominik RYZKO and Henryk RYBINSKI (2006), Distributed default logic for multi-agent system, in *Proceedings of Intelligent Agent Technology 2006*.
- Dominik RYZKO (2006), *Default Logics for Reasoning and Knowledge Sharing in Multi-Agent Systems*, Ph.D. thesis, Warsaw University of Technology.