# Discovering Strategic Behaviors in Multi-Agent Scenarios by Ontology-Driven Mining

Davide Bacciu[1], Andrea Bellandi[1], Barbara Furletti[1],
Valerio Grossi[2] and Andrea Romei[2]
[1]*IMT Institute for Advanced Studies Lucca*
[2]*Department Of Computer Science – University of Pisa*
*Italy*

## 1. Introduction

Providing human users with a structured insight into extensive data collections is a problem that, in the latter years, has gathered increasing attention by the scientific community. An even more daring and ambitious research challenge is the recent attempt to address the same problem within the scope of the so-called multi-agent systems. In this context, multiple autonomous and heterogeneous entities, i.e. the agents, populate the environment performing actions and taking decisions based on internal policies and their knowledge of the world.

The courses of action of the agents and their interaction with the environment and with themselves generates extensive amounts of information. Underneath such flat data collections lies fundamental knowledge concerning strategies and common behaviors emerging from the agent actions. The advantage of extracting such "strategic" knowledge is twofold. On the one hand, it allows users a deeper insight into the multi-agent system, permitting to discover interesting patterns of actions that can be identified as interesting behaviors in the particular problem at hand. Consider, for instance, a multi-agent system simulating negotiation/selling phases in a particular market (Viamonte et al., 2006): the detection of interesting patterns of actions can bring to light specific market strategies that can be exploited to optimize revenues. On the other hand, the identified strategic knowledge can be exploited by the agents themselves to improve their performance, for instance by updating their internal representation of the world and their behavioral policies.

In order to extract such "strategic" knowledge, in this work we take a data mining approach using Association analysis, enriching it with the expressive power and the flexibility of ontologies. Association rules can effectively tackle both aspects of strategic behavior extraction, since they both provide a human understandable representation means for the extracted knowledge and an action rule base that can be used to supply agents with procedural and high-level knowledge concerning the identified strategies.

Ontologies offer a structured description of the domain knowledge while maintaining data and their representation separated. An ontology refers to an "engineering artifact", consisting of a specific vocabulary containing the terms used to describe a certain domain, and a set of explicit assumptions regarding the meaning of vocabulary words. This set of

Open Access Database www.i-techonline.com

Source: Advances in Robotics, Automation and Control, Book edited by: Jesús Arámburo and Antonio Ramírez Treviño,
ISBN 78-953-7619-16-9, pp. 472, October 2008, I-Tech, Vienna, Austria

assumptions has usually the form of a logical theory, where terms correspond to unary or binary predicates, respectively called concepts and relations. For our purposes the logics supplies the formal language, the semantics and a set of proof procedures for representing the various aspects of the knowledge and for reasoning about it, in order to drive the data mining process.

In general, data mining consists of a series of transformation steps, starting from data preprocessing up to post-processing, for converting raw data into useful information. The ontology drives the association rules extraction process intervening in the pre and post processing phases. Complex categories of domain-specific constraints are specified during the preprocessing phase, while during the post-processing phase, the extracted association rules are further filtered for providing better and more fitting results.

Recent works have introduced the use of association rules for imitation learning in robotic agents within a reactive paradigm (Hellström, 2003), as well as in learning simple grasping and picking behaviors in agricultural robots (Zou et al., 2006), while (Makio et al., 2007) exploited association analysis to discover basic motor skills from human motion capture data. However, here we are not interested in determining low level motor behaviors in single agents, e.g. fight or fly reactions; rather, we seek to discover high level strategies underlying the actions of complex multi-agents systems. Within this respect, the introduction of ontologies for constraint-based multi-level association rule mining induces advantages in terms of the human interpretability of the strategies resulting from the Association analysis. Moreover, ontologies offer a simple mean for providing the agents with a representation of the extracted strategies that can be used, for instance, as high level primitives in the planning step of the agents' control hierarchy.

The rest of the chapter is organized as follows: Section 2 introduces the data mining concepts that are used thorough the paper and describes related works in the area of behavior mining in agent-based systems. Section 3 formalizes the proposed model, while in Section 4 we describe the behavior mining approach and we present an example related to the simulated robotic soccer league. Section 5 concludes the paper with final considerations and future developments.

## 2. Background

The following subsections describe the background knowledge as well as the main concepts of Data Mining and Association analysis, together with a brief survey of the application of association rules to Behavior Mining in Agent-based Systems. Moreover we introduce the ontology theory that is used in the definition of the proposed model.

### 2.1 Constrained association rule mining

*Knowledge Discovery in Databases* (KDD) (Tan et al., 2005) is an emerging field that covers a wide range of applicative domains, several models for representing extracted patterns and models, and a large number of algorithms for data preprocessing, model extraction and model reasoning. KDD addresses the task of extracting "meaningful information" from large volumes of data. *Data Mining* (DM) is an important part of the *knowledge discovery process*, whose aims is to convert raw data into useful information. DM consists of a series of transformation steps, from data *pre-processing* to *post-processing*. Pre-processing reduces and transforms raw input data into an appropriate format for the mining step. Post-processing ensures that only valid and useful results are retained.

Association analysis is a central task in DM. It aims at finding strongly correlated sets of events from large databases. In 1993, (Agrawal et al., 1993) introduced association rules for discovering regularities among products in large scale transaction data recorded by point-of-sale (POS) systems in supermarkets. For example, the rule {onions, vegetables} → {beef} indicates that if a customer buys onions and vegetables together, he/she is likely to buy beef as well. Such information can be used as the basis for decisions about marketing activities, such as promotional pricing or product placements. In addition to the above example from market basket analysis, association rules are employed today in many application areas including web mining, intrusion detection, bioinformatics and so on.

Nowadays, one of the most important open issue in association rules mining is due to the large number of extracted rules. In fact, it is not unusual for the collection of rules to be too large for being handled directly by an analyst. Moreover, often one would like to be able to focus the mining process on a particular portion of the search space or, in other words, it is important to have some additional expressiveness in order to define what is interesting and what is not. In order to solve these problems, frameworks for constrained association rule mining have been developed.

In the remaining of this section, we formalize the problem of association rules mining, briefly introducing the Apriori algorithm; finally, we review the constraints-based association rule mining problem.

**Definition.** Let $I = \{i_1, \ldots, i_n\}$ be a set of literals, called *items*. Each item has some predefined attributes (e.g. price, type, etc.). Let $D$ be a set of *transactions*, which form the database that is going to be disclosed. Each transaction $t \in D$ is an *itemset*, such that $t \subseteq I$. A unique identifier (TID) is associated with each transaction. A transaction $t$ supports $X$, i. e. a set of items in $I$, if $X \subseteq t$. An itemset $X$ has *support* $s = supp(X)$ if $s\%$ of the transactions supports $X$. If $|X| = k$, then $X$ is called a *k-itemset*.

An *association rule* (AR) is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \varnothing$. $X$ is said *antecedent* (or *body*) and $Y$ *consequent* (or *head*). An AR has a *support* $s$ on D, if $s\%$ of the transactions in $D$ contain $X \cup Y$. An association rule has a *confidence* $c$ on D if the $c\%$ of the transactions in $D$ containing $X$, also include $Y$. In formulas:

- $supp(X \rightarrow Y) = supp(X \cup Y) = |X \cup Y| / |D|$;
- $conf(X \rightarrow Y) = supp(X \cup Y) / supp(X)$.

It is worth noticing that, while support is a measure of the frequency of a rule, confidence is a measure of the strength of the relation between sets of items.

An itemset is said *frequent*, if its support is higher than a user-specified minimum support. A rule is said *strong* if its support and its confidence are higher than a user-specified minimum support and a user-specified minimum confidence, respectively.

A constraint on itemsets is a function $C_I: 2^I \rightarrow \{true, false\}$. An itemset $I$ satisfies a constraint $C$, if and only if $C_I(I) = true$. Similarly, a rule constraint is a function $C_R: R \rightarrow \{true, false\}$, where $R$ is the set of association rules. In general, given a conjunction of constraints, the constrained association rule mining problem requires to compute all the rules that satisfy the constraints.

**Algorithms for AR mining.** Association rule mining algorithms scan the database of transactions and calculate the support and confidence of the candidate rules to determine whether they are significant or not. Their computational process is composed of two steps: (1) given a minimum support, find all frequent itemsets; (2) given a set of frequent itemsets and a minimum confidence, find all the association rules that are strong. Step (1) is, usually,

harder than rule generation and the main difficulty of itemsets generation comes from the large size of the search space.

Apriori has been the first frequent itemset algorithm to be proposed (Agrawal et al., 1994). It exploits a bottom-up, level-wise exploration of the lattice of frequent itemsets. During a first scan of the dataset, all the frequent singletons, denoted with $L_1$, are found. Next, $L_1$ is used to find the set of frequent 2-itemsets $L_2$, and so on until all the frequent patterns have been discovered.

Each iteration is composed of two steps: *candidate generation* and *support counting*. During the first step, a collection of possibly frequent itemsets $C_k$ of length $k$ is generated, and their actual support is calculated with a single scan over the dataset during the second step. In principle, it would be easy to generate all the possible k-itemsets at each iteration, given that $I$ is known. As a result, we will have an extremely large collection of itemsets that have to pass the support counting phase. The Apriori algorithm exploits a pruning of the search space in order to reduce the number of candidates before each iteration. Pruning is based on the concept that, if $X \in C_k$ has a subset of length $k - 1$, that does not belong to $L_{k-1}$ i.e. the subset is not frequent, $X$ cannot be frequent as well, and it can be removed from $C_k$.

**A survey of constraints-based AR mining.** The fundamental idea inspiring constrained AR mining is to introduce a set of constraints that association rules should satisfy. The analyst can use a conjunction of constraints for specifying the properties of the patterns of interest. As a trivial example, let us focus on market basket analysis. Lets suppose that each item is associated to a fixed price, and that we are interested only on those co-occuring set of items such that their average price is higher than a given threshold. KDD systems should be able to exploit such constraints to speedup the knowledge extraction process.

The most important classes of constraints and their properties are shown in Fig. 1. For a complete and formal definition please refer to (Bonchi & Lucchese, 2007).

The first interesting class is the *anti-monotone* (Agrawal et al., 1993), which was already introduced with the Apriori algorithm (i.e. the minimum frequency is an anti-monotone constraint). It simply states that, if a predicate holds for a set $S$, then it holds for any subset of $S$. At the opposite, the *monotone* (Bucila et al., 2002) constraints state that, if a predicate holds for a set $S$, it holds also for any superset of $S$. For example, the constraint $max(S.price) > 100$ is monotone, while $max(S.price) \leq 100$ is anti-monotone.

The *succinct* constraint (Han et. al., 1999) states that the decision concerning the satisfaction of a predicate can be determined based on the single element of the $S$ set. For instance, the max constraints defined above are both succinct, since a single element of $S$ can be used as a comparator. The $avg(S)$ constraint is not succinct.

The class of *convertible anti-monotone* (resp. *convertible monotone*) (Pei & Han, 2000) includes constraints $C_{CAM}$ (resp. $C_{CM}$) that define an item ordering relation such that whenever an itemset S satisfies $C_{CAM}$ (resp. violates $C_{CM}$), so does any prefix of $S$. For instance, $mean(S.price) > 100$ is a convertible anti-monotone constraint, with respect to the descending price order. In fact, $C_{CAM}(S) \Rightarrow C_{CAM}(T)$, where $S$ is a prefix of $T$.

The class of *loose anti-monotone* constraints, introduced by (Bonchi & Lucchese, 2005), simply states that, given an itemset $S$ with $|S| > 2$, a constraint is loose anti-monotone (denoted $C_{LAM}$) if: $C_{LAM}(S) \Rightarrow \exists i \in S: C_{LAM}(S \setminus \{i\})$. It is trivial to show that $variance(S.price) < 100$ is a loose anti-monotone constraint.

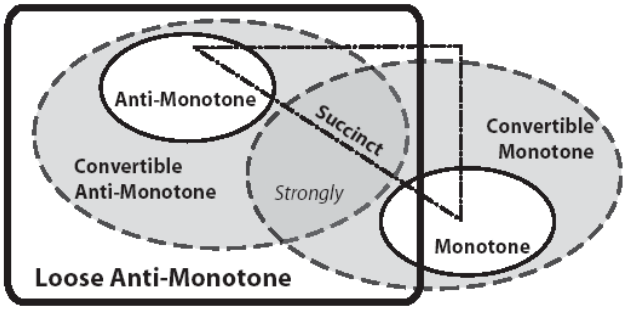


Fig. 1. Characterization of the classes of constraints.

For each of these classes, there exists a specialized algorithm, which is able to take advantage of its peculiar properties. Techniques for constraint-driven pattern discovery can also be classified, depending on the KDD phase in which they are evaluated, in the following categories:

- *pre-processing constraints* are evaluated during the pre-processing phase, restricting the source data to the instances that can generate only patterns satisfying the constraint;
- *mining constraints* are directly integrated into the mining algorithm (e.g. Apriori) used for extracting the rules;
- *post-processing constraints* are evaluated for filtering out patterns after the mining algorithm.

(Wojciechowski & Zakrzewicz, 2002) focuses on improving the efficiency of constraint-based frequent pattern mining, by using dataset filtering techniques that conceptually transform a given data mining task into an equivalent one operating on a smaller dataset.

A first preliminary effort to propose a general framework for constrained frequent itemsets mining is (Bonchi & Lucchese, 2007), in which the authors present the design of *ExAMiner*$^{GEN}$, a general Apriori-like algorithm, which is able to exploit all the possible kinds of constraints presented above. In particular, the authors suggest a data reduction technique for obtaining advantages from the conjunction of monotone and anti-monotone constraints. They observe that a transaction that does not satisfy the monotone constraint CM, can be removed since none of its subsets will satisfy CM either, and therefore the transaction cannot support any valid itemset. This data reduction, in turn, lowers the support of other frequent but invalid itemsets, thus reducing the search space and improving anti-monotone pruning techniques. This virtuous circle is encoded level-wise in the Apriori algorithm.

Constraints can also improve the comprehension of the extracted rules, with the aid of a concept hierarchy. A concept hierarchy is a multilevel organization of the various entities or concepts defined in a particular domain. For example, in the market basket analysis a concept hierarchy has the form of an item taxonomy describing the "is-a" relationships among the items sold in a store, e.g. *"milk"* is a kind of *"food"* and *"DVD"* is a kind of *"home electronics"* equipment. Mining on concept hierarchies is also called constraint-based multilevel DM and association rules extracted with the aid of a taxonomy are also called *multi-level*.

The main advantages of incorporating concept hierarchies into association analysis are: (i) items at lower levels may not have enough support to appear in any frequent itemsets and (ii) rules found at lower levels of a hierarchy tend to be too specific and may not be interesting as rules at higher levels.

Methods to define and integrate item constraints on a concept hierarchy are originally introduced in (Agrawal & Srikant, 1995) and (Fu & Han, 1995). Originally, multilevel constraints are evaluated during the pre-processing phase, generalizing items at bottom level to higher levels of the hierarchy before applying Apriori. Basically, each transaction *t*, is replaced with its "extended transaction", which contains all the items in *t* along with their corresponding ancestors. (Srikant et al., 1997) and (Han et al., 1999) can be seen as the first attempts to integrate both constraint-based and multilevel mining directly into the rule extraction algorithm, employing anti-monotone proprieties existing across levels: "if a high-level item is infrequent, none of its descendants can be frequent".

In (Goethals & Van den Bussche, 2003) the authors integrate the evaluation of constraints into the mining process, so that the amount of computation is proportional to what the user gets, limiting the exponential generation of patterns and rules. The constraints considered in this work are Boolean combinations of atomic conditions, e.g. *Body(i)* (or *Head(i)*) specifies that a certain item *i* occurs in the body (respectively, in the head) of a rule. Another interesting work on integrating constraints evaluation in the mining process is represented by DualMiner (Bucila et al., 2003). This system implements a new algorithm for finding frequent itemsets, that efficiently prunes its search space by employing both monotone and anti-monotone constraints.

Other remarkable constraint-based optimizations can be found in (Gallo et al., 2005) and (Esposito et al., 2006). The former proposes an "incremental" approach exploiting the results of previous queries in order to reduce the response time to new queries. In this context the term *incremental* describes the fact that the outcome of a new query starts from a previous result. The proposed algorithm is able to deal with context-dependent constraints. A similar idea is used in (Esposito et al., 2006), where a new query is rewritten in terms of union and intersection of the result sets of other previously executed and materialized queries. The authors propose a two-step approach. Firstly, they find conditions to apply various typologies of constraints (essentially on items properties), then, by using an optimizer, they recognize equivalent queries avoiding repeated heavy computations.

Finally, (Wang et al., 2003) explores *support constraints*, that specify what minimum support is required for each itemsets, so that only the necessary itemsets are generated. The three constraints $supp_1(B_1, B_3) \geq 0.2$, $supp_2(B_3) \geq 0.4$ and $supp_3(B_2) \geq 0.6$ are examples of support constraints where each $B_i$, called *bin*, is a set of disjoint items which, taken in isolation, do not need to satisfy any minimum support requirement.

In this chapter, we focus on pre-processing and post-processing multi-level constraints. However, we plan to develop further our model by integrating constraints evaluation directly into the mining phase, taking advantage of the peculiar properties of each constraint.

### 2.2 Behavior mining in agent-based systems

In last twenty years, data mining has been extensively applied to provide users with a deep insight into data collections, supplying powerful tools for extracting high-level knowledge from extensive, often poorly structured, information sources. Recently (Cao et al., 2007) attention has started to shift from the classical view of data mining as a tool for transferring knowledge from raw data to human agents (i.e. the users), to a novel interpretation where data mining serves to transfer knowledge from data collections to intelligent artificial agents, under the form of strategic patters of actions, i.e. behaviors.

At the beginning, most of the works focused on learning single agent behaviors from demonstrational data: in (Hellström, 2003), for instance, a robotic system based on the reactive paradigm that learns complex behaviors by imitation is presented. Through a stimuli/response mapping, the system determines the appropriate robot action as a direct function of its sensory inputs. This mapping is provided by a set of association rules that are extracted from sensory/action data gathered as the human user exemplifies the behaviors that have to be learned by the robotic agent. Similarly, (Zou et al., 2006) use association rules to learn picking behaviors for agriculture robots. Other recent works focused on data mining as a tool for extracting motor primitives from motion data: (Makio et al., 2007) discusses how to discover motor skills by mining human motion capture data to extract association rules and frequent patterns that describe the dependencies among the body parts and the relative movements. In (Xu et al., 2007), on the other hand, association rules are used to discover and describe knowledge of chewing behavior in an object oriented framework. (Mori et al., 2005) collects information on the household behavior of people in daily life: by means of association rules, they discover frequent combinations of events, called episodes, that are used to predict and support the daily actions of the user.

The models described so far relate to single agent scenarios (either robotic or purely software), but much of the recent works have been devoted to multi-agent systems. These ones differ from single agent systems in the sense that there is no global control and globally consistent knowledge and, since data and control are distributed, multi-agent models have the inherent advantages of distributed systems, such as scalability, fault-tolerance and parallelism. Notice that the term "multi-agent" refers to two aspects of the problem: the former and most straightforward one relates to the fact that the system itself is, indeed, composed of multiple agents. The latter aspect relates to the fact that the data used in the mining process is produced by multiple sources, i.e. by multiple human or artificial agents. This is the case of several agent based models with application to information retrieval, computational economics and robotics. In (Viamonte et al., 2006), for instance, is presented a Multi-Agent Market Simulator where the agents behavior follows the customers and seller profiles extracted by mining information from the market segment under analysis. (Bezek et al., 2006) presents an algorithm for extracting strategies from a RoboCup soccer scenario comprising several interacting agents with different roles and behaviors. In particular, the model extracts association rules from the sequences of basic multi-agent actions, showing how the constructed rules capture basic soccer strategies such as offensive moves and defense schemes.

More generic models, such as the Agent Academy framework (Mitkas et al., 2004) support design and instantiation of agent communities as well as ontology definition. This system integrates an agent training procedure that is a mechanism for embedding rule-based reasoning capabilities and application-specific knowledge into agents. This is realized by means of a data-mining module that processes available behavioral knowledge in order to extract decision models for the agents. Another general purpose multi-agent framework is (Kaya et al., 2005). It uses association rule mining to estimate agent actions, maintaining an historic database of the past agent actions that is processed to extract association rules that are, in turn, used to update the action selection strategies of the agents. With a similar intent, (Dudek, 2007) proposes the APS (Analysis of Past States) framework, where web browsing agents alternate between work phases, during which actions are performed and information concerning the user preferences are recorded, and stand-by phases, where the agents are

idle and the historic information is mined to extract new user profiles under the form of association rules.

In general, there is an increasing trend of integrating strategic behavioral knowledge into agents by mining repositories containing data collected either exogenously to the system (e.g. by human exemplification) or by analyzing historic data of the agents themselves. Such knowledge extraction mechanism requires tools capable of extracting significant behavioral patterns from large data collections. Additionally, the representation of such behavioral patterns should allow cross interpretability of the data mining results both from the machine and the user perspectives. We advise that association rules can play a key role in this sense, serving two key purposes: as knowledge representation means they provide humans with insight into data collections; as an action rule base, they can be used to supply virtual agents with procedural and high-level knowledge that can effectively be exploited for increasing the agents' performance. In particular, we will focus on how the introduction of ontologies in the association mining process can successfully improve the interpretability of the association analysis's results both from the point of view of the user and from the agents' perspective.

### 2.3 Ontology

The ontology may be defined as the *study of being as such*. The construction of the word itself has remote origins and may be seen as an attempt to modernize the classical debate about metaphysics. One of the first exponent of this new "discipline" was Aristotle that, in (Aristotle, 350 B.C.), presents a first work that can be resembled to an "ontology". He provides a list of categories that represent an inventory of what there is, in terms of the most general kinds of entities. Later on, the Latin term "ontologia" appeared for the first time in the circles of German protestants around the 1600 (Göckelin, 1980) (Lorhard, 1613), while the first English appearance goes back to 1721 in the Baileys dictionary in which it was defined as "an Account of being in the Abstract". The first use of the term ontology in the computer and information science literature occurs instead in 1967, in a work on the foundations of data modeling by S. H. Mealy (Mealy, 1967).

From the scientific point of view, the ontology is, in its first approximation, a table of categories in which every type of entity is captured by some node within a hierarchical tree.

While the philosopher-ontologist has only the goal of establishing the truth about the domain in question, in the world of the information systems the ontology is a software (or formal language) designed with a specific set of uses and computational environments. In this field, the ontologies are rapidly developing (from 1990 up to now) thanks to their focus on classification and their abilities in representing the domains, in sharing the knowledge and in keeping separate the domain knowledge from the operational one. Furthermore, they play a key role in the Semantic Web by providing a source of shared and precisely defined terms that can be used for describing the resources. Reasoning over such descriptions is essential for accessibility purposes, automating processes and discovering new knowledge.

One of the modern ontology definition given by (Maedche, 2003) is reported below. An ontology $O$ is defined as

$$O = \{C, R, A^O\},$$

where:

1. $C$ is a set whose elements are called concepts.

2. $R \subseteq (C \times C)$ is a set whose elements are called relations. For $r = (c_1, c_2) \in R$, one may write $r(c_1) = c_2$.
3. $A^O$ is a set of axioms on $O$.

To cope with the lexical level, the notion of a lexicon is introduced. For an ontology structure $O = \{C, R, A^O\}$ a lexicon L is defined as

$$L = \{L^C, L^R, F, G\},$$

where:

1. $L^C$ is a set whose elements are called lexical entries for concepts.
2. $L^R$ is a set whose elements are called lexical entries for relations.
3. $F \subseteq L^C \times C$ is a reference for concepts such that

$$F(l_C) = \{c \in C: (l_C,c) \in F\} \text{ for all } l_C \in L^C,$$

$$F^{-1}(c) = \{l_C \in L^C: (l_C,c) \in F\} \text{ for all } c \in C.$$

4. $G \subseteq L^R \times R$ is a reference for concepts such that

$$G(l_R) = \{r \in R: (l_R,r) \in G\} \text{ for all } l_R \in L^R,$$

$$G^{-1}(r) = \{l_R \in L^R: (l_R,r) \in G\} \text{ for all } r \in R.$$

We note that a hierarchical structure can be explicitly defined in terms of $R$.

Finally, the mapping from elements in the lexicon to elements in the ontology structure corresponds to the notion of an interpretation in declarative semantics (Genesereth and Nilsson, 1987). Such a (semantic) interpretation associates elements of a language to elements of a conceptualisation. Based on the above definitions, an ontology can be formally defined as the couple $<O, L>$ where $O$ is an ontology structure and $L$ is a corresponding lexicon. The ontology structure $O$ plays the role of an explicit specification of a conceptualisation of some domain while the lexicon $L$ provides the agreed vocabulary to communicate about this conceptualisation.

The following example may be useful for explaining the above definitions.

Let $O = \{C, R, A^O\}$ be an ontology structure such that $C = \{c_1, c_2\}$ and $R = \{r\}$, where $r(c_1) = c_2$. Suppose that $A^O = \varnothing$. Also, let $L = \{L^C, L^R, F, C\}$ be a corresponding lexicon such that $L^C = \{Mouse, Input_device\}$, $L^R = \{is_a\}$, $F(Mouse) = c_1$, $F(Input_device) = c_2$ and $G(is_a) = r$.

Fig. 2 depicts the elementary ontology described above. The top of the figure shows the ontology structure $O$. It consists of two concepts $c_1$, $c_2$ and one relation $r$ that relates them. This corresponds to a conceptualisation of a domain of interest without any lexical reference. The latter is provided by the lexicon $L$ depicted at the bottom of the figure. $L$ provides the lexical references for $O$ by means of $F$ and $G$. $F$ and $G$ map lexical reference strings to the concepts and relations defined in $O$, respectively. For instance, for $r \in R$ one may consider using $G^{-1}(r)$ to get the lexical reference, i.e. "*is_a*", corresponding to $r$ and vice versa.

The Web Ontology Language (OWL) is the current standard provided by the World Wide Web Consortium (W3C) for defining and instantiating Web ontologies. An OWL ontology can include descriptions of classes, properties and their instances. The OWL language provides three increasingly expressive sublanguages:

- OWL Lite supports a classification hierarchy and simple constraint features.

- OWL DL is more expressive than OWL Lite without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. Its name is due to its correspondence with description logics (Baader, 2003), a field of research that has studied a particular decidable fragment of first order logic.
- OWL Full is the most expressive with no computational guarantees.

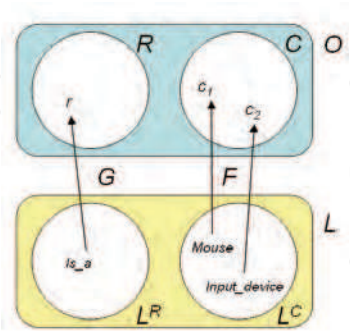


Fig. 2. Graphical representation of an ontology.

We focus mainly on the first two variants of OWL because OWL-Full has a nonstandard semantics that makes the language undecidable and therefore difficult to implement. OWL comes with several syntaxes, all of which are rather verbose. Hence, in this chapter we use the standard DL syntax. For a full DL syntax description, please refer to (Baader, 2003).
The main building blocks of an OWL ontology are (i) concepts (or classes), representing sets of objects, (ii) roles (or properties), representing relationships between objects, and (iii) individuals, representing specific objects. Furthermore they are composed of two parts: intensional and extensional. The former part consists of a *TBox* and an *RBox*, and contains knowledge about concepts (i.e. classes) and the complex relations between them (i.e. roles). The latter one consists of an *ABox*, containing knowledge about entities and the way they are related to the classes as well as roles from the intensional part. In the following is an example of DL formalization (according to the syntax defined in (Baader, 2003)) related to the previous ontology fragment. We add a property specifying the number of keys of each input device

$$TBox = ((Mouse \subseteq Input_device) \cap (Input_device \subseteq Thing) \cap (=1\ hasKeysNumber) \subseteq Input_device \cap (\forall\ hasKeysNumber.integer)).$$

All the possible ontology instances constitute the *ABox* which is interlinked with the intensional knowledge. An example of *ABox* related to the previous *TBox* is the following:

$$ABox = \{Logitech_device : Input_device,\ hasKeysNumber = 88,\ Optical_IBM_Mouse : Mouse,\ hasKeysNumber = 2\}$$

Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. The semantics of OWL DL is fairly standard by DL standards. An example is in Fig. 3.
An interpretation $I = (\Delta^I, \cdot^I)$ is a tuple where $\Delta^I$, the domain of discourse, is the union of two disjoint sets $\Delta_O{}^I$, (the object domain) and $\Delta_D{}^I$ (the data domain), and $I$ is the interpretation function that gives meaning to the entities defined in the ontology. $I$ maps each OWL class $C$ to a subset $C^I \subseteq \Delta_O{}^I$, each object property $P_{Obj}$ to a binary relation $P^I{}_{Obj} \subseteq (\Delta_O{}^I \times \Delta_O{}^I)$, each datatype property $P_{Data}$ to a binary relation $P^I{}_{Data} \subseteq (\Delta_O{}^I \times \Delta_D{}^I)$. The complete definition can be found in the OWL W3C Recommendation[1].

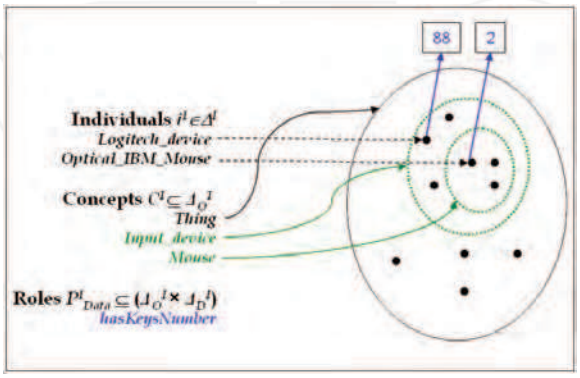


Fig. 3. Example of well defined (model theoretic) semantics.

Concepts and relations represent the explicit knowledge of an ontology, nevertheless another kind of knowledge (i.e. the implicit knowledge) can be "deduced" starting from the known facts. Its existence is implied by or inferred from observable behavior or performance by using the rule inference mechanism. Rules are an extension of the logical core formalism, which can still be interpreted logically. The simplest variant of such rules are expressions of the form $C \Rightarrow D$ where $C$ and $D$ are concepts. The meaning of such a rule is "if an individual is proved to be an instance of $C$, then derive that it is also an instance of $D$". Operationally, the semantics of a finite set $R$ of rules can be described by a forward reasoning process. Starting with an initial knowledge base $K$, a series of knowledge bases $K^{(0)}$, $K^{(1)}$, … is constructed, where $K^{(0)} = K$ and $K^{(i+1)}$ is obtained from $K^{(i)}$ by adding a new assertion $D(a)$ whenever $R$ contains a rule $C \Rightarrow D$ such that $K^{(i)} \leq C(a)$ holds, but $K^{(i)}$ does not contain $D(a)$.
Semantic Web Rule Language (SWRL) is the W3C proposal for the rule language based on a combination of the OWL DL and OWL Lite sublanguages with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language[2]. SWRL allows users to write Horn-like rules that can be expressed in terms of OWL concepts and that can reason about OWL individuals (Horrocks, 2004). According to the SWRL syntax and referring to the previous example of TBox and ABox, the inference rule for deducing that an ontology element is an instance of Mouse stated that is an Input_device and has 2 keys, is reported below:

$$\forall x,y \;\; Input_device(?x) \wedge hasKeysNumber(?x,?y) \wedge equalTo(?y, 2) \rightarrow Mouse(?x),$$

where equalTo(., .) is a SWRL built-in function.

[1] The W3C webpage: http://www.w3c.org/.
[2] Rule Markup Language is specified at www.ruleml.org.

## 3. The system

This section describes in detail our framework and the process for extracting multi-level association rules by using the ontological support (Bellandi et al., 2008).
Our scenario consists of the set of components shown in Fig. 4.

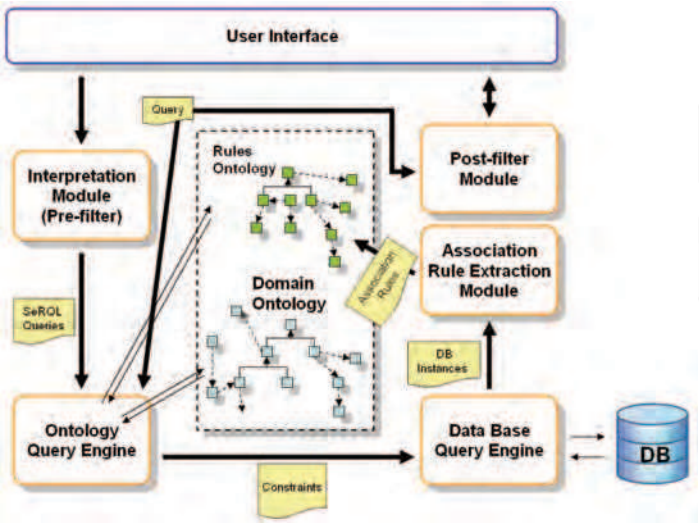


Fig. 4. Architecture of the system.

As in the most common system architectures, our framework is composed of a set of "modules" and a set of repositories. There is a repository for the metadata, i.e. the ontologies, and a repository for the transactions, i.e. the DataBase (DB). The former repository contains the **Domain Ontology** (DO) and the **Rule Ontology** (RO). The DO describes the domain of interest, i.e. all the items that compose the transactions and that are stored in the DB. The RO, instead, is used for storing the association rules extracted during the process. It is different from the DO because it models the structure of an association rule according to its definition (see Section 2.1). The DO is the description model for the collection of data and is not designed for containing actual instances. The RO, instead, is used both for describing and containing the inference rules.
The **User Interface** provides mechanisms for interacting with the system and for visualizing the results. The user can provide the system with two types of input: a set of user constraints for specifying what items must be considered in the discovering process, and another one for filtering the extracted association rules. The user specifies constraints by using the syntax presented in the Table 1. It includes both pruning constraints, used to filter a set of non-interesting items, and abstraction constraints, which permit a generalization of the item on a concept of the ontology. We informally describe each constraint by means of an example.
By using pruning constraints, one can specify a set of items which can be excluded from the input transactions set, and, as a consequence, from the extracted rules. This kind of constraints refers either to a single item, or to an ontology concept, and can include a condition expressed on a set of ontology properties. There exist two kinds of pruning constraints:

$\mathcal{I}$ is the set of items $(i_1, i_2, ...i_n \in \mathcal{I})$.
$\mathcal{C}$ is the set of the concepts of the ontology $(c_1, c_2, ...c_n \in \mathcal{C})$.
$\mathcal{P}_c$ is the set of the properties of the concept $c \in \mathcal{C}$ $(p_1, p_2, ...p_n \in \mathcal{P}_c)$.
$cond_c$ is a Description Logic expression.
$ALL$ represents all the instances defined in the ontology.

A constraint is defined on $\mathcal{I}$, $\mathcal{C}$ and $\mathcal{P}_\mathcal{C}$ in the following form:

1. **Pruning Constraints.** A pruning constraint is of one of the following forms:
   (a) $prune(e)$, where $e \in \mathcal{I} \cup \mathcal{C} \cup \{ALL\}$.
   (b) $prune_{cond_c}(c)$, where $c \in \mathcal{C} \cup \{ALL\}$.
2. **Abstraction Constraints.** An abstraction constraint is of one of the following forms:
   (a) $abstract(e, c)$, where $e \in \mathcal{I} \cup \mathcal{C}$, $c \in \mathcal{C}$ and $c$ is a super-concept of $e$.
   (b) $abstract_{cond_{c_1}}(c_1, c_2)$ where $c_1 \in \mathcal{C} \cup \{ALL\}$, $c_2 \in \mathcal{C}$ and $c_2$ is a super-concept of $c_1$.
   (c) $abstract^l_{cond_e}(e)$, where $e \in \mathcal{I} \cup \mathcal{C} \cup \{ALL\}$, and $l$ is a non-negative integer indicating the level of the hierarchy; $cond$ can be unspecified.

Table 1. Syntax of the Pruning and Abstraction constraints.

- The constraint *prune*(e) excludes a single item or all the items belonging to an ontology concept expressed by "e". As one could expect, removing an ontology concept c ∈ C also implies excluding all the items belonging to the descendent concepts of c.
- The constraint $prune_{cond}$(c) is similar to *prune*(e), except for uninteresting items that are selected according to a condition defined on a set of properties of c. The condition can also refer to a property of an ancestor of c.

For example, let us suppose to have a simple ontology modelling all possible soccer actions performed by soccer agents. Every item represents an action and has a property hasPassesNumber to indicate the number of passes of an action. The ontology has the form of a taxonomy organized according to the kind of action (e.g. Action, GoalAction, Non-GoalAction, CornerAction, etc.). The predicates *prune*(Non-GoalAction) and $prune_{hasPassesNumber>5}$(GoalAction) allow to prune all the "Non-GoalAction" actions and all the "GoalAction" actions with a number of passes greater than 5, from the input transactions. Notice that hasPassesNumber is a property of the upper concept Action, while Non-GoalAction and GoalAction are sub-concepts of Action.

The Abstraction constraints enable to explore different levels of the ontology concepts. The generalization to a predefined level of the hierarchy increases the support of association rules and, consequently, avoids the discovery of a massive quantity of useless rules, especially in case of sparse data. More interesting is the exploitation of rules in which items are extracted at different levels of the hierarchy. The system offers three different predicates:

- The predicate *abstract*(e, c) generalizes a single item e (or all the items belonging to the concept e) to the super-concept c.
- The predicate $abstract_{cond<c1>}(c_1, c_2)$ is similar to the previous one, but it generalizes items according to a condition ($cond<c_1>$) defined on a set of properties of the concept $c_1$. Only items satisfying this condition are abstracted to the super-concept $c_2$. As for $prune_{cond}$(c), this condition can also refer to a property of an ancestor of $c_1$.
- Finally, the $abstract^l_{cond<e>}$(e) generalizes every element satisfying the condition $cond<e>$ to the super-concept placed at level l of the hierarchy. In this case, e can be either a

single item or a concept. If the condition is not specified, all the items belonging to e are generalized to the super-concepts at level l.

The symbol "ALL"in Table 1 can be used to select all the items defined in the ontology.

Suppose that the ontology provides also a description of all the soccer field areas, organized as an hierarchy; this permits to specify in which soccer field area each action is performed.

Lets assume that LittlePenaltyArea is a sub-concept of PenaltyArea. The predicate *abstract*(LittlePenaltyArea, PenaltyArea) generalizes all the actions performed in the "Little Penalty Area" to a more generic "Penalty Area". It is equivalent to the predicate $abstract^5$(LittlePenaltyArea), if PenaltyArea is placed at level 5 of the ontology.

The set of constraints $V_D$ is therefore a conjunction of pruning and abstraction constraints.

The **Interpretation Module** in Fig. 4 translates the requests of an user into a set of formal constrains (defined on the DO) so that they can be supplied to the Ontology Query Engine by means of a suitable query language. In our system we used the Sesame RDF Query Language - SeRQL (Broekstra, 2004).

The **Ontology Query Engine** interacts with the DO by performing some queries and retrieving the desired set of items (IS). This module uses a particular ontology query language according to the ontology management system that supports the operations.

The output is used by the **DB Query Engine** for retrieving the DB transactions that contain the items specified in IS.

The **Association Rule Extraction Module** contains an implementation of the *Apriori* algorithm that computes the set of association rules (AR). Such a rules are then stored in the RO. Finally, the **Post-filter Module** evaluates the post filtering constraints for selecting the most relevant association rules w.r.t. the user needs. Since the extracted rules are stored in the OR, this module interacts with the Ontology Query Engine for retrieving the correct items by means of directed queries. The post-filter module interacts with the GUI for receiving the SERQL queries and for presenting the output rules. It is important to notice that the constraints are conceived not only on the structure of the rules, but also on their semantics, since the domain ontology provides a mean to express the semantics of single items belonging to the rules. For example, let us consider the case in which the ontology models all the possible soccer actions performed by soccer agents. Association rules can be mined to highlight relations among the actions performed by all players. Given a set of output rules, we are able to focus only on those in which the antecedent contains at least one kick action made by a player having a particular role. In this case, the semantics of the item "kick" and "player role" are encoded in the domain ontology.

## 4. Discovering strategic behaviors by association rules mining

In this section we describe how the association rule mining system presented above can be instantiated in the context of behavior discovery in multi-agents system (in Section 4.1), and we describe (in Section 4.2) an application example in the field of simulated robotic soccer.

### 4.1 Multi-agent planning with ontology-based association rules

Earlier in the chapter, we have discussed the increasing trend of behavior discovery in multi-agent systems. Here, we focus on the role of association rule mining as a means for extracting such strategic action patterns: in particular, we point out the capital importance of an ontology based approach for extracting high-level planning knowledge that can

understood both by the human users as well as by the software agents. For this reason, we present an ontology-based planning scheme for intelligent computational entities.

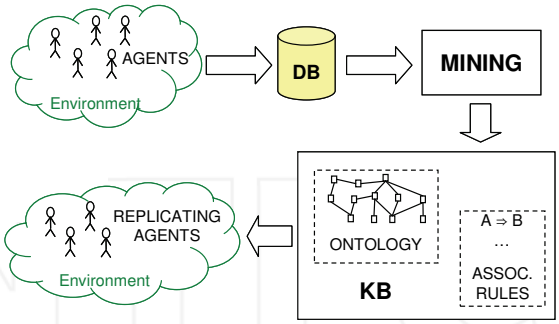


Fig. 5. Behavior mining with association rules in a multi-agent system

Consider, for instance the scenario in Fig. 5: a set of reference agents, either human or artificial, is interacting within an environment, each of them pursuing a particular goal based on its internally defined policy. The *configurations* of the system, including the courses of actions of the agents as well as characteristic information (e.g. spatial location, battery level and signal strength in radio agents) are monitored and logged to a database (DB in Fig. 5). After a sufficient amount of time, the database will contain massive amounts of *flat* information referring to the development of the agents' behavior in time, as well as the strategies used to pursue their goals. Our work aims at analyzing such large data repository to extract a knowledge base (KB in Fig. 5) holding higher level information content that can be used to refine the planning strategies of the *replicating* intelligent agents in Fig. 5.

Consider, for instance, the elements in Fig. 5 as being players from two soccer teams: by monitoring their actions during the whole match we can log interesting information, such as player location, speed, stamina, as well as events of interest, e.g. kick, pass, header, slide, dash, etc. Then, by applying association analysis, we can extract a set of association rules that describes the most likely decisions to be taken for the most frequent configurations of the game. For instance, the knowledge base can end up containing rules such as

> ***IF*** *LeftPlayer10.location IS RightQuarterMidfield AND LeftPlayer10.pass AND LeftPlayer11.location IS RightPenaltyArea*
> ***THEN*** *LeftPlayer11.dash;*

which describes a typical soccer attack strategy, that is "when player 10 is on the attack and passes the ball, then player 11 should run for the pass in the penalty area". The extracted rules are, in turn, used to populate our knowledge repository KB, which includes a description of our reference *world* given by the *domain ontology.*

For the particular problem at hand, i.e. soccer matches, the ontology can include, for instance, a taxonomy of the areas of the field, a description of the players (e.g. in terms of physical/physiological parameters such as height, weight, stamina, speed, etc.) as well as a multi-layered description of the actions and events (e.g. the pass action that can be further differentiated in long and short pass, heal pass, etc.).

The novelty of our approach lies in the full integration between the ontology and the association analysis: the ontological description can be used to navigate the results of the association analysis at a given level of abstraction by imposing ontology-based constraints to filter the association rules in the KB. More formally, this is obtained by applying the *rule ontology* population scheme in Fig. 4: by exploiting the knowledge representation in the *domain ontology* we perform queries that filter the association rules at particular level of abstraction. The resulting rules serve to populate the rule ontology that is in turn used for the decision making process.

Association rules filtering is of capital importance when approaching behavior mining in a multi-agent system, due to the explosive rule base complexity that may arise as a result of the association analysis. The interaction of multiple agents, in fact, generates a massive amount of data that populates the DB; as a consequence the traditional association analysis can generate a large number of rules addressing the behavior of several agents. Thus, a pruning technique is needed to reduce the cardinality of the association rules by filtering the results of the association analysis to obtain personalized views of the discovered procedural knowledge. In our approach this mechanism is implemented by means of ontology-based filtering. For instance, if we are interested in knowledge regarding heal pass actions performed in opponent's team area, we apply the ontological definitions of heal pass and opponent's area to the association rules in the knowledge repository. As a result only those rules satisfying the constraints are retained and used to plan the next decision. This approach offers a two-fold advantage: on the one hand, it allows human users to selectively tune their view on the extracted knowledge so that their analysis of the data can be more tightly focused. On the other hand, such an approach is fully exploitable by intelligent software agents to obtain behavioral patterns that are tailored to the particular decision scenario they are addressing. The ontological description of the world can, in fact, be easily coded into the software agent. The agent itself can then use it to gather knowledge from the mare-magnum of the association rule base at the required descriptive level (i.e. abstraction constraints) or for a particular ontological relation (i.e. property constraints). By this means, we can define intelligent software agents that exploit their world description, i.e. the ontology, to learn useful behavioral patterns, i.e. the ontology-filtered rules, from a vast amount of flat data.

Going back to our soccer example, we can interpret the replicating agents in Fig. 5 as being players in a virtual soccer match: the rule base extracted from the *actual* soccer players is accessed by each replicating agent that applies its ontology-based filter to extract those behavioral patterns that best fit the current game configuration. Then, the planning module of the replicating agent determines the next action (or sequence of actions) based on the retrieved decision rules (see Fig. 6 ). In a sense, the replicating agent is learning a behavioral pattern by analyzing the most frequent configurations generated by the *actual* agents and selecting only those that fits best its role and its personal view of the world. Notice that, in general, nothing prevents the real agents to be the same as the replicating agents; hence, by devising *smart* association rule filters, an agent can be made to learn from the history of its past actions.

Fig. 6 depicts a prototypical planning module for a replicating agent that is based on the proposed ontology-driven behavior mining model. The agent builds the current configuration descriptor by gathering information from the environment, the proprioceptive

sensors as well as from its internal state. This information is used to determine the preconditions for the decision rules and is exploited by the planner to elaborate the next course of actions. In order to do so, the agent planner queries its knowledge base at the required level of abstraction and obtains a set of decision rules that match the current system configuration. Based on these rules, the planner determines the next actions that are forwarded to the actuator sub-system.

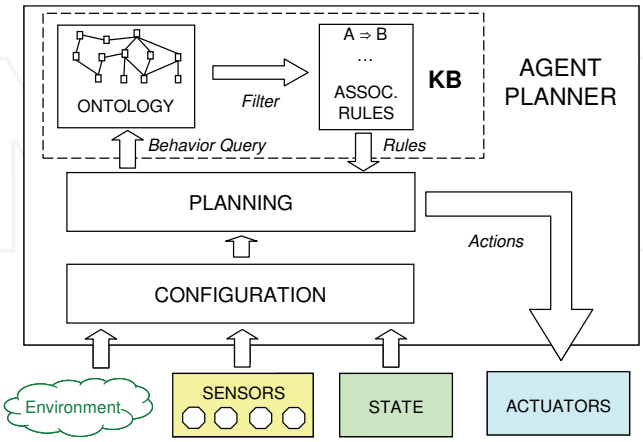


Fig. 6. Agent planner based on the ontology-driven behavior mining model

The simple soccer scenario that we have adopted to introduce the behavior mining scheme is not far from a real-world application of the model. Several soccer teams, in fact, have started monitoring their players with sensors both during training sessions and during games (e.g. see the MilanLab[3] structure of AC Milan). Such information, if accessible, can in principle be used to extract strategic action patterns that can be exploited, for instance, to devise intelligent software agents for virtual soccer matches, such as in the Robocup league. Unfortunately, such data has not been disclosed, yet: hence, in the next section, we present a behavior extraction example based on the freely available logs from the Simulated Soccer League.

### 4.2 Extracting strategic behaviors from robot soccer data: an example

In this paragraph we provide a schematic example describing the steps of the analysis process. The example is based on data collected from the RoboCup simulated league. RoboCup is an international initiative to foster artificial intelligence and intelligent robotics by providing a standard test bed where several technologies can be integrated and examined. The RoboCup simulated league consists of teams of autonomous software agents competing against one another in soccer games. Logs were taken from each simulation game consisting of about 6000 simulation cycles. Each cycle contains information about player location and orientation, ball location, ball possessor, game

[3] http://www.acmilan.com/InfoPage.aspx?id=565

score and play modes such as throw-ins, goal-kicks and so on. Since our aim is to "mine" the behaviour of a team in the soccer RoboCup league in the 2004 edition[4], we have collected all the log events related to the match played by the winner team, focusing on position and action information.

**Step1: The ontology definition.**

We defined an ontology for describing the entities of the soccer domain, focusing on the variables actually contained in the collected data. We built the ontology by creating a taxonomy of concepts, adding interesting data as well as object properties according to the target of our analysis.

In Fig. 7 is reported the ontology of the main concepts (superClasses) and their data and object properties. For our purposes it was necessary to model the concepts *Player*, *Kick*, *Body*, *Ball*, *PlayerRole* and *FootballFieldArea*. Concerning the kick action, it is interesting to know the actor and the receiver (2 players), where it is directed (the field area) as well as the body part that has produced it. The corresponding object properties are *madeBy*, *towards*, *hasDestinationArea*, *madeByUsing*.

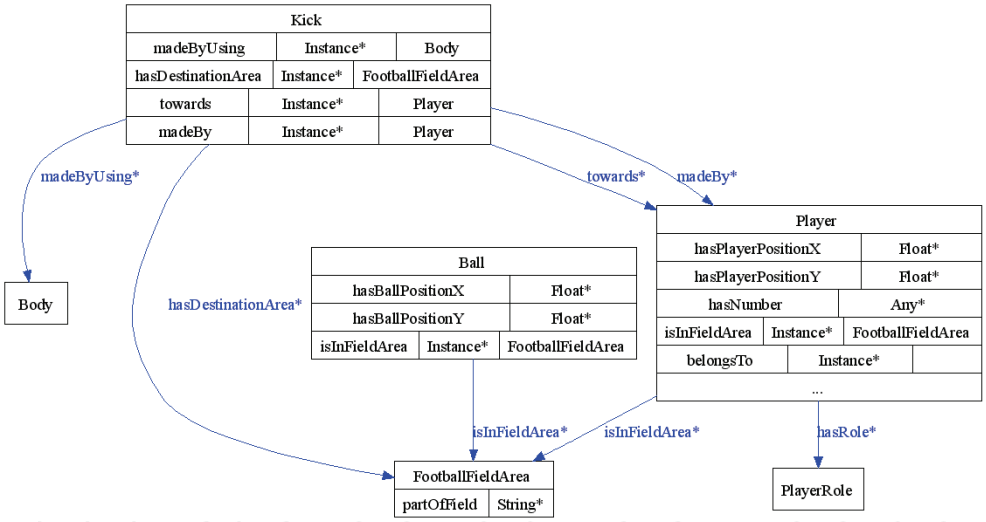


Fig. 7. The domain ontology for the soccer domain: the super-concepts.

For a player is interesting to know its role and its position. The latter information is of interest also for the ball concept.

For the concepts *Body*, *FootballFieldArea* and *PlayerRole* we realized three taxonomies of sub-concepts (Fig. 8, Fig. 9 and Fig. 10 respectively), that in the following will be used to express AR constraints.

[4] http://staff.science.uva.nl/~jellekok/robocup/rc04/

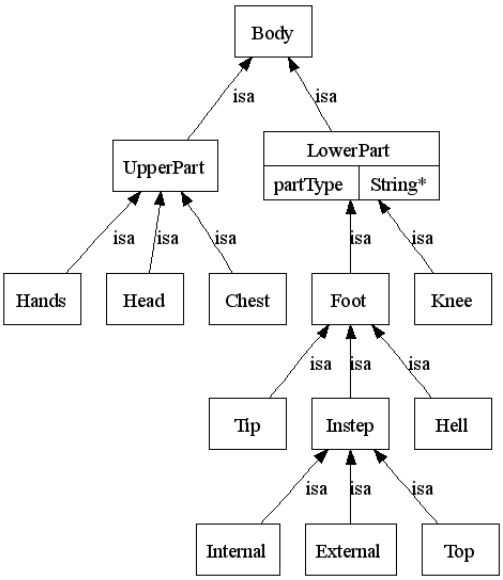


Fig. 8. The *Body* taxonomy.

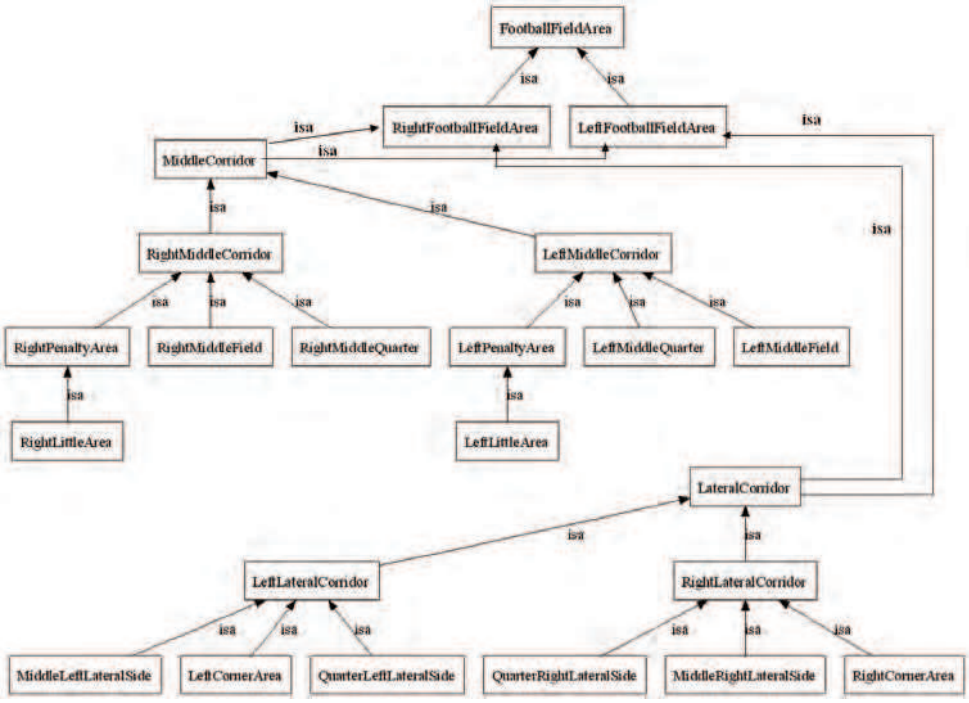


Fig. 9. The *FootballFieldArea* taxonomy.

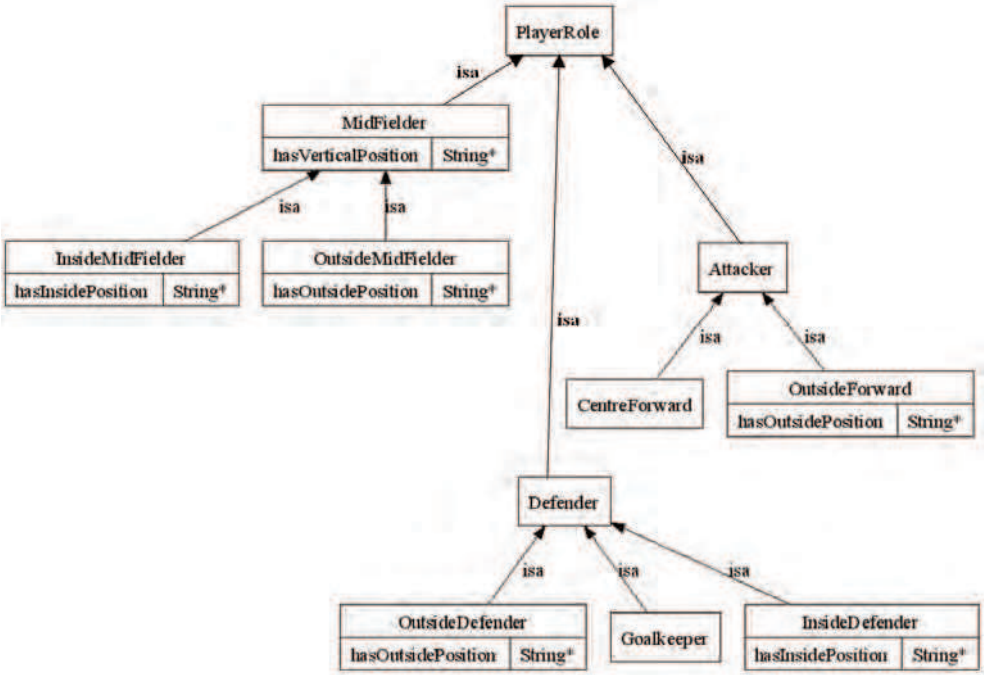


Fig. 10. The *PlayerRole* taxonomy.

The information related to all the previous ontological entities, was not explicitly represented in the DB. Starting from the information contained in the DB, we inferred the new information according to our ontology entities. For instance, in the *FootBallFieldArea* taxonomy, we split the soccer field into 24 disjointed parts, that are used to characterize players and ball positions, as showed in figure Fig. 11.

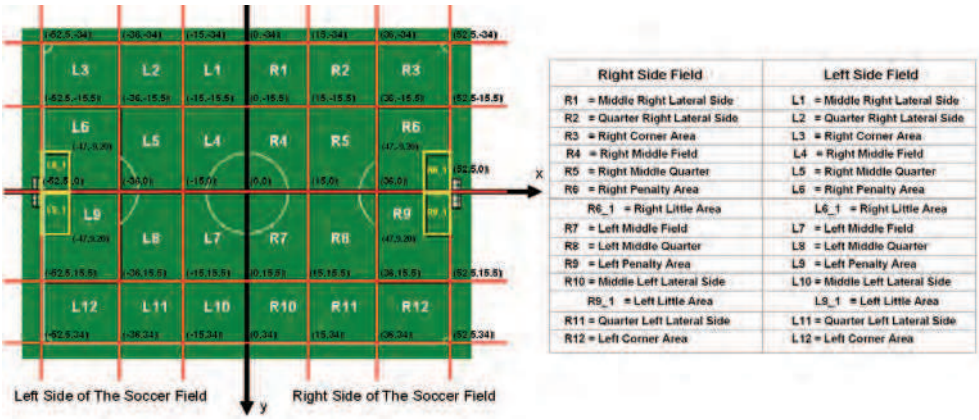


Fig. 11. Parts of the soccer field.

Starting from the coordinates of each position, inference rules are used for identifying each part of the soccer field. The SWRL rules have the following form:

$$\forall x,y,z \;\; Player(?z) \wedge hasPlayerPositionX(?z, ?x) \wedge hasPlayerPositionY(?z, ?y) \wedge \\ \text{GreaterOrEqualThan}(?x, -52.5) \wedge \text{LessOrEqualThan}(?x, -47) \wedge \\ \text{GreaterThan}(?y, -9.2) \wedge \text{LessThan}(?y, 0) \\ \rightarrow isInFieldArea(?z, LLA_L).$$

This rule states that a certain player (identified by the variable *z*), belonging to the *Left* team, is in the left part of the little area (*LLA_L*), where *LLA_L* is an individual of the *LeftLittleArea* ontology class. Similar rules have been used for deducting all the other ontology entities. According to the architectural description in Section 3, we introduce an ontology for storing the ARs, that is shown below in Fig. 12.

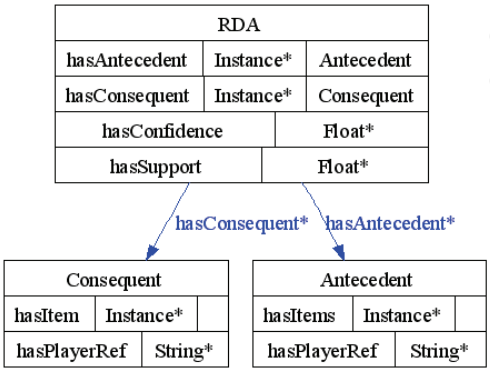


Fig. 12. The Rule Ontology.

This ontology models the structure of an association rule. The concept *RDA* identifies a rule: it has the reference to the confidence and the support, as well as the link (1:n relation) to concepts *Antecedent* and *Consequent*, that are, in turn, used to model the left and right hand side of a rule.

**Step2: Data collection and preparation.**

Raw data consists of about 65.000 transactions and 108 attributes. The structure of each one of the 22 players is reported in Table 2. Moreover, we have some actual attributes related to

| Attribute | Type | Description |
|---|---|---|
| *pos_x* | real | X Coordinate of a player position in the football field |
| *pos_y* | real | Y Coordinate of a player position in the football field |
| *actions_kick* | {0, 1} | 1 if actual action is equal to kick, 0 otherwise |
| *actions_dash* | {0, 1} | 1 if actual action is equal to dash, 0 otherwise |
| *actions_turn* | {0, 1} | 1 if actual action is equal to turn, 0 otherwise |
| *actions_catch* | {0, 1} | 1 if actual action is equal to catch, 0 otherwise |
| *actions_move* | {0, 1} | 1 if actual action is equal to move, 0 otherwise |
| *calculated_possessor* | {0, 1} | 1 if player has ball, 0 otherwise |

Table 2. Player Attributes.

the ball state, such as the coordinate of the ball in the football field (*ball_position*), the distance of the ball from a goal (*ball_goal_distance*) and the possessor team of the ball (*ball_possessor*).

The rule extraction algorithm is not applied directly to the collected data described above, whereas such information is abstracted or filtered in agreement with the user specified constraints. For example a player position, expressed by x-y coordinates, can be abstracted to whatever field zone defined by the ontology.

We selected all the actions where the team under analysis is in "offensive state". To determine an "offensive state", we abstract the *ball_position* attribute to an ontology high level concept considering only those actions in which the ball is into a specific part of the football field, for example *PenaltyArea*, or simply in the opponent midfield. Then, we filtered the data considering only the actions where at least one player executes a kick. Finally, we observed the behavior of the forward players (e.g. with numbers from 9 to 11) related to the opposite defenders (e.g. with numbers from 2 to 5).

It is important to notice that all the data transformations introduced above are expressed by means of constraints defined on the basis of a well-defined domain ontology, and automatically executed by the software interpreting the results provided by the ontology.

**Step3: The constraints definition.**

Our simple experimentation concerns two queries. In the first case, the objective is to find frequent itemsets involving "kicking players" and actions performed in the right middle soccer field. In the second case, we are interested in these specific kicking players: PlayerL9, PlayerL10, PlayerL11, PlayerR2, PlayerR3 and PlayerR4. By using our constraint language, an user can formulate the first query in the following way:

$$
\begin{aligned}
query1 = \; & (prune_{(belongsTo = 'R')}(\text{Player}) \\
& \wedge prune_{(hasNumber > 5 \text{ or } hasNumber = 1)}(\text{Player}) \\
& \wedge prune_{(belongsTo = 'L')}(\text{Player}) \\
& \wedge prune_{(hasNumber < 9)}(\text{Player}) \\
& \wedge prune_{(isInField = RightPart)}(\text{Player}) \\
& \wedge prune_{(madeBy\text{-}1 = 'null')}(\text{Player}) \\
& \wedge abstract^{3}(\text{FootBallFieldArea}))
\end{aligned}
$$

In *query1*, the first four clauses are used for pruning all the players *PlayerR1, PlayerR5, PlayerR6, PlayerR7, PlayerR8, PlayerR9, PlayerR10, PlayerR11, PlayerL1, PlayerL2, PlayerL3, PlayerL4, PlayerL5, PlayerL6, PlayerL7* and *PlayerL8*. The 5th clause permits to prune all the players performing actions in the left part of the soccer field, while the 6th one permits to prune all the players not kicking (by using the inverse property of madeBy). The last clause abstracts all the positions in the filed to level 3 of the FootBallFielArea taxonomy.

The second example is defined by the following query:

$$
\begin{aligned}
query2 = \; & (prune_{(belongsTo = 'R'}(\text{Player}) \\
& \wedge prune_{(hasNumber > 5 \text{ or } hasNumber = 1)}(\text{Player}) \\
& \wedge prune_{(belongsTo = 'L')}(\text{Player}) \\
& \wedge prune_{(hasNumber < 9)}(\text{Player}) \\
& \wedge prune_{(isInField = RightPart)}(\text{Player}) \\
& \wedge prune_{(madeBy\text{-}1 = L1 \text{ or } madeBy\text{-}1 = L2 \text{ or } madeBy\text{-}1 = L3 \text{ or} \\
& \quad madeBy\text{-}1 = L4 \text{ or } madeBy\text{-}1 = L5 \text{ or } madeBy\text{-}1 = L6 \text{ or} \\
& \quad madeBy\text{-}1 = L7 \text{ or } madeBy\text{-}1 = L8)}(\text{Player})
\end{aligned}
$$

$\wedge\ prune_{(madeBy\text{-}1 = R1 \text{ or } madeBy\text{-}1 = R5 \text{ or } madeBy\text{-}1 = R6 \text{ or } madeBy\text{-}1 = R7 \text{ or } madeBy\text{-}1 = R8 \text{ or } madeBy\text{-}1 = R9 \text{ or } madeBy\text{-}1 = R10 \text{ or } madeBy\text{-}1 = R11)}(\text{Player})$

$\wedge\ abstract^3(\text{FootBallFieldArea}))$.

*Query2* is very similar to *query1*, but instead of selecting all the kicking players, it retrieves only those items in which either *PlayerL9*, *PlayerL10*, *PlayerL11*, *PlayerR2*, *PlayerR3* or *PlayerR4* performs the kick.

**Step4: Extracting the Behavior Patterns.**

Once that the ontology-based query has gone through all the steps discussed above, we obtain a populated rule ontology specifying a set of association rules that describes the behavioral patterns of interest.

As an example, let us focus on defensive strategies by centering our analysis on the behavior of the defenders *D = {PlayerR1, PlayerR2, PlayerR3, PlayerR4}* in the right-side team with respect to the forwards *F = {PlayerL9, PlayerL10, PlayerL11}* of the left team. To study the positioning of the players at a sufficient level of detail, we abstract to the bottommost layer of the taxonomy in Fig. 9 (see Fig. 11 for a visual representation of the field areas). Moreover, to quantize actions to *relevant* time instants we filter out those transactions where the ball is not kicked by any player. If we are interested in discovering knowledge regulating the in-field positioning of the defenders, we can query for relevant association rules regarding the players position on the field. An example of high-confidence ontological rules produced by this association analysis is:

*a.* *PlayerL10_RMQ_R, PlayerL9_LMQ_R, PlayerR2_QLLS_R, PlayerR4_RMQ_R --> PlayerR3_LMQ_R;*

*b.* *PlayerL10_RMQ_R, PlayerL11_LMQ_R, PlayerR2_LMQ_R --> PlayerR4_RMQ_R;*

*c.* *PlayerL11_LPA_R, PlayerL9_LPA_R, PlayerR3_LPA_R PlayerR5_RPA_R, --> PlayerR2_LPA_R;*

*d.* *PlayerL10_RMF_R, PlayerR5_MRLS_R --> PlayerR4_RMF_R.*

Rule (a), for instance, describes the typical defense scheme where a player, i.e. *PlayerL10*, is controlled by the defender *PlayerR4*, while a second attacker, i.e. *PlayerL9*, has to be defended by *PlayerR3* which, indeed, has to position in the same area of the field of *PlayerL9*. Notice that the rule states that a third defender, i.e. *PlayerR2*, is already holding a position and should not defend on *PlayerL9*. Similarly, rule (b) requires *PlayerR4* to defend on *PlayerL10* while *PlayerR2* is on *PlayerL11*. These kinds of rules can be used by those agents implementing the defender players to determine their position on the field in standard defense schemes. Rule (c) describes a dangerous situation in which two forwards, i.e. *PlayerL11* and *PlayerL9*, are in the left side of the penalty area: *PlayerR3* is defending on both players while *PlayerR4* is on the right side of the penalty. Hence, a smart defense strategy requires *PlayerR2* to help *PlayerR3* on the left side. On the other hand, rule (d) describes a typical pattern where *PlayerR5* should keep its position, not defending on *PlayerL10,* while *PlayerR4* should localize in the same field position as *PlayerL10*.

If we are interested in discovering action patterns correlated with the defense schemes, we can let action concepts into the itemset together with the players' positions. An example of rules that we obtained from the simulated soccer data is:

*e.* *PlayerR2_LMQ_R_dash, PlayerR3_LMQ_R_dash --> PlayerR4_RMQ_R_dash;*

*f.* *PlayerL9_RMQ_R_dash --> PlayerR4_RMQ_R_dash.*

The former rule describes a typical joint defensive movement, that is: when two defenders located in the same field area dash, then neighbor players, i.e. *PlayerR4*, should follow them.

The latter rule, on the other hand, states probably the most typical defensive movement: if a forward *PlayerL9* dashes in the field area of a defender *PlayerR4*, then *PlayerR4* should follow him.

The same approach that has lead us to discover defensive patterns can be used to extract knowledge concerning attack strategies. For instance, let us study first the positioning on the field for the forwards *F = {PlayerL9, PlayerL10, PlayerL11}* of the left team. By querying the rule ontology we obtain rules such as:

*g.* *PlayerL11_QLLS_R, PlayerL9_LMQ_R, PlayerR4_LMQ_R --> PlayerL10_RMQ_R;*

*h.* *PlayerL10_hasBall, PlayerL9_RMQ_R --> PlayerL11_LMQ_R;*

i. *PlayerL10_RPA_R, PlayerL10_hasBall, PlayerL9_RPA_R, PlayerR4_RPA_R --> PlayerL11_LPA_R.*

Rule (g) describes a basic attack strategy where the forward *PlayerL11* is off-center on the left lateral side while *PlayerL9* is controlled by the defender *PlayerR4* on the center-left; therefore, the third forward *PlayerL10* should chase the space on the center-right area. Rule (h) identifies another fundamental, yet very simple, attack strategy that is: if *PlayerL10* has the ball and is about to kick (notice that we have filtered kick-only timeframes) then the two other forwards, i.e. *PlayerL9* and *PlayerL11*, should position outside the penalty area, respectively on the right and on the left, waiting for a pass or a rebound. Rule (i), on the other hand, depicts a clear scoring chance where *PlayerL10* is holding the ball in the right penalty area and *PlayerL9* is defended by *PlayerR4* on the same position, hence *PlayerL11* should chase the space, and a possible rebound or pass, on the left of the penalty.

Likewise with defensive strategies we can query for attack action patterns by including action concepts in the itemset. Notice that there are two approaches for achieving this: the former, that was already shown in rules (e)-(f) considers items as a concatenation of actions and positions; the latter considers actions and positions separately. An example of attack strategies is:

*j.* *PlayerL9_LMQ_R_dash --> PlayerL10_RMQ_R_dash;*

*k.* *PlayerL10_hasBall, PlayerL9_RMQ_R --> PlayerL10_kick, PlayerL11_LMQ_R;*

*l.* *PlayerL11_hasBall, PlayerL9_LMQ_R, PlayerR3_LMQ_R --> PlayerL10_RMQ_R, PlayerL11_kick.*

Rule (j) is, again, an example of the first approach: it simply states that *PlayerL10*, positioned on the right of the penalty area, should follow *PlayerL9* when he dashes outside the left of the penalty area (i.e. joint attack). Rules (k) and (l), on the other hand, are two examples of the second approach. The first rule, for instance, states that, if *PlayerL10* has the ball and *PlayerL9* is positioned outside the right of the penalty area, then *PlayerL11* has to chase the space on the left and *PlayerL10* should shoot or pass. Taking a closer look at the database transactions that falls in the support of rule (k) confirms that, actually, *PlayerL10* tries to pass or shoot the ball whenever this rule is enabled. Similarly, rule (l) states that *PlayerL10* should occupy the area on the right of the penalty and *PlayerL11* should pass or shoot if *PlayerL9* is defended by *PlayerR3* on the left.

The rules described above are just a small sample of those obtained by the proposed ontology-based association analysis on the simulated soccer data. In general, each agent implementing a particular role should filter the rules of its interest, that is those containing interesting information on the consequent. By exploiting its internally defined ontology, each agent can obtain personalized views of the knowledge concealed in the flat data.

Typically, an agent could be interested only in association rules whose consequent refers to a specific player, e.g. *PlayerL10*. By using our constraint language, the agent can express such a post-processing constraint in the following way:

$$\text{query_post} = prune_{(\text{hasPlayerRef} <> \text{PlayerL10})}(\text{Consequent}).$$

Indeed, the value of the retrieved expert knowledge will depend strictly on the quality on the world description, i.e. the domain ontology, as well as on the available data, that is to say, if we rely solely on the strategies of loser teams, we will most probably obtain also-ran agents.

## 5. Conclusion

We have introduced an ontology-based approach for association analysis in the context of behavior mining in multi-agent systems. Our proposal is based on the idea of the ontological description of the domain as an essential via-point for accessing the expert knowledge concealed underneath massive amounts of "flat" data. The introduction of a multi-layered and multi-relational representation of the domain allows approaching the information content from several, diverse, viewpoints. Within the multi-agent area, this approach offers considerable advantages since it allows the agents to gather personalized views of the extracted knowledge, represented by means of rule ontologies. By exploiting this "relativistic" representation, an agent can dynamically generate and selectively access, at the desired level of abstraction, the knowledge that is of higher relevance for its current decision-making activity. Besides presenting the clear advantage of offering personalized views of the world, which is totally consistent with the multi-agent model, this approach relieves the agents from the burden of acquiring, maintaining and mastering a monolithic, and encyclopedic, representation of their knowledge.

The model presented in this chapter tackles the association analysis task with the standard static approach where each transaction is considered in isolation and not as part of a spatiotemporal trajectory. Sequence mining, on the other hand, studies how to approach the problem of finding frequent patterns for trajectory data. An interesting future development for our model would be to extend it to sequence mining (Agrawal & Srikant, 1995): in particular, this would be of great interest for behavioral pattern mining, since it can naturally tackle the problem of planning medium to long term strategies comprising lengthy sequences of inter-dependent actions. However, we would like to point out that our model already offers a means for processing multiple transactions in a sort of time trajectory. Through the domain ontology is possible, in fact, to describe concepts whose definition transcends the single transaction. Consider, for instance, the pass concept in the soccer ontology: since it requires to specify source and destination of the action, its instantiation needs to process multiple transactions to find all the requested information, e.g. the receiver of the pass.

## 6. References

Agrawal, R.; Imielinski, T. & Swami, A. (1993). Mining Association Rules Between Sets of Items in Large Databases. *Proceedings of the SIGMOD Conference* 1993, pp. 207-216, Washington, D.C., May 1993

Agrawal, R.; Methta M.; Shafer, J. & Srikant, R (1994). Fast algorithms for mining association rules in large databases. *Proceeding of the 20th International Conference on Very Large Databases (VLDB'94)*, pp. 478–499, ISBN 1-55860-153-8, Santiago de Chile, Chile, September 1994, Morgan Kaufmann, San Francisco, CA

Agrawal, R. & Srikant, R. (1995). Mining Generalized Association Rules. *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pp. 407-419, ISBN 1-55860-379-4, Zurich, Switzerland, September 1995, Morgan Kaufmann, San Francisco, CA

Agrawal, R. & Srikant, R. (1995). Mining Sequential Patterns. *Proceedings of the Eleventh International Conference on Data Engineering,* pp. 3-14, ISBN 0-8186-6910-1, Taipei, Taiwan, May 1995

Aristotle. (350 B.C.) Categories, The Internet Classic Archive

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D. & Patel-Schneider, P. (2003) The description logic handbook. Cambridge University Press (2003)

Bellandi, A.; Furletti, B.; Grossi, V. & Romei, A. (2008) Ontological Support for Association Rule Mining, In: *Proceedings of IASTED International conference on Artificial Intelligence and Applications – AIA 2008*

Bezek, A.; Gams, M. & Bratko, I. (2006) Multi-agent strategic modeling in a robotic soccer domain, In: *Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multiagent Systems*, Hakodate, Japan, May 2006, 457-464, ACM, New York, NY

Boekstra, J. & Kampman, A. (2004) SeRQL: An RDF Query and Transformation Language

Cao, L.; Luo, C. & Zhang, C. (2007) Autonomous Intelligent Systems: Multi-Agents and Data Mining, LNCS, Vol. 4476, 60-73, Springer, Berlin

Bonchi, F. & Lucchese, C. (2005). Pushing tougher constraints in frequent pattern mining. *Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '05)*,pp. 114-124, ISBN 3-540-26076-5, Hanoi, Vietnam, May 2005, Springer

Bonchi, F. & Lucchese C. (2007). Extending the state-of-the-art of constraint-based pattern discovery, In: *Data and Knowledge Engineering*, Vol. 60, No. 2, (February 2007) pp. 377–399, ISSN 0169-023X

Bucila, C.; Gehrke, J.; Kifer, D. & White, W. (2002). Dualminer: a dual-pruning algorithm for itemsets with constraints. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 2002)*, pp. 42–51, Edmonton, Alberta, Canada , July 2002

Dudek, D. (2007) Using Data Mining Algorithms for Statistical Learning of a Software Agent, In: Agent and Multi-Agent Systems: Technologies and Applications, LNCS, Vol. 4496, 111-120, Springer, Berlin

Esposito, R.; Meo, R. & Botta, M. (2006). Answering constraint-based mining queries on itemsets using previous materialized results. In: *Journal of Intelligent Information Systems,* Vol 26, No. 1, (January 2006) pp. 95-111, ISSN 0925-9902

Fu, Y. and Han, J. (1995). Discovery of multiple-level association rules from large databases, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB '95)*, pp. 420-431, ISBN 1-55860-379-4, Zurich, Switzerland, September 1995, Morgan Kaufmann, San Francisco, CA

Gallo A., Esposito R., Meo, R. & Botta, M. (2005). Optimization of Association Rules Extraction Through Exploitation of Context Dependent Constraints. *Proceedings of the 9th Congress of the Italian Association for Artificial Intelligence (*AI*IA 2005), pp. 258-269, Milan, Italy, September 2005

Göckelin, R. (1980) Lexicon philosophicum. Reprinted by Georg Olms 2 edition 1980.

Goethals, B. & Van den Bussche, J. (2000). Interactive Constrained Association Rule Mining. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (DaWaK 2000),* pp. 307-316, ISBN 3-540-67980-4, London, UK, September 2000, Springer-Verlag, London, UK

Han, J.; Lakshmanan, L. V. S.; Ng, R. T. & Pang, A. (1999). Optimization of constrained frequent set queries with 2-variable constraints. *Proceedings of the 1999 ACM SIGMOD international conference on Management of data (SIGMOD '99)*, pp. 157–168, ISBN 1-58113-084-8, Philadelphia, PA, ACM, New York, NY

Han, J.; Lakshmanan, L. V. S. & Ng, R. T. (1999). Constraint-Based Multidimensional Data Mining. In: *IEEE Computer*, Vol. 32, No. 8, (August 1999) pp. 46-50, ISSN 0018-9162

Hellström, T. (2003). Learning robotic behaviors with association rules. *WSEAS Transactions on Circuits and Systems,* Vol. 2, No. 3, July 2003, 534-546, ISSN 1109-2734

Horrocks, I. (2004). SWRL: A Semantic Web Rule language Combining OWL and RuleML. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

Kaya, M. & Alhajj, R. (2005) "Fuzzy OLAP association rules mining-based modular reinforcement learning approach for multiagent systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 2, April 2005, 326-338

Lorhard J. (1613) Theatrum philosophicum, Basilia

Makio, K.; Tanaka, Y. & Uehara, K. (2007) Discovery of Skills from Motion Data, In: *New Frontiers in Artificial Intelligence*, LNCS, Vol. 3609, 266-282, Springer, Berlin

Mealy G. H. (1967) Another look at data, In *Proceedings of the Fall Joint Computer Conference*, Vol 31, 525-534. Thompson Books, London: Accademic Press

Mitkas, P.A.; Kehagias, D.; Symeonidis, A.L. & Athanasiadis I.N. (2004) A Framework for Constructing Multi-agent Applications and Training Intelligent Agents, In: *Agent-Oriented Software Engineering IV*, LNCS, Vol. 2935, 255-290, Springer, Berlin

Mori, T.; Takada, A.; Noguchi, H.; Harada, T. & Sato, T. (2005) Behavior prediction based on daily-life record database in distributed sensing space, *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 1703-1709, 2-6 Aug. 2005, IEEE

Pei, J. & Han, J. (2000). Can we push more constraints into frequent pattern mining? *Proceedings of the sixth ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pp. 350–354, ISBN 1-58113-233-6, Boston, MA, August 2000, ACM, New York, NY

Srikant R.; Vu, Q. & Agrawal, R. (1997). Mining Association Rules with Item Constraints. *Proceedings of the 3rd International Conference of Knowledge Discovery and Data Mining (KDD '97)*, pp. 67-73, ISBN 1-58113-233-6, Newport Beach, CA, August 1997, ACM, New York, NY

Tan, P. N.; Steinbach M. & Kumar V. (2005). *Introduction to Data Mining (First Edition)*, Addison-Wesley Longman Publishing Co., ISBN 0321321367, Inc. Boston, MA, USA

Viamonte, M.J.; Ramos, C.; Rodrigues, F. & Cardoso, J.C. (2006) ISEM: a multiagent Simulator for testing agent market strategies, *IEEE Transactions on Systems, Man, and Cybernetics, Part C,* vol. 36, no. 1, Jan. 2006, 107-113

Wang, K.; He, Y. & Han, J. (2003). Pushing Support Constraints Into Association Rules Mining. In: *IEEE Transactions on Knowledge and Data Engineering,* Vol. 15, No. 3, (May-June 2003) pp. 642-658, ISSN: 1041-4347

Wojciechowski, M., Zakrzewicz, M., Dataset Filtering Techniques in Constraint-Based Frequent Pattern Mining, In: *Lecture Notes in Computer Science*, Vol. 2447, pp. 77-91, ISBN 3-540-44148-4, Springer-Verlag, London, UK

Xu W.L., Kuhnert L., Foster K., Bronlund J., Potgieter J. & Diegel O. (2007) Object-oriented knowledge representation and discovery of human chewing behaviours, *Engineering Applications of Artificial Intelligence,* Vol. 20, No. 7, October 2007, 1000-1012

Zou, X.; Lu, J.; Luo, L.; Luo, X. & Zhou, Y. (2006) Extracting Behavior Knowledge and Modeling Based on Virtual Agricultural Mobile Robot, In: *Advances in Artificial Reality and Tele-Existence*, LNCS, Vol. 4282, 29-37, Springer, Berlin

**Advances in Robotics, Automation and Control**

Edited by Jesus Aramburo and Antonio Ramirez Trevino

ISBN 978-953-7619-16-9
Hard cover, 472 pages
**Publisher** InTech
**Published online** 01, October, 2008
**Published in print edition** October, 2008

The book presents an excellent overview of the recent developments in the different areas of Robotics, Automation and Control. Through its 24 chapters, this book presents topics related to control and robot design; it also introduces new mathematical tools and techniques devoted to improve the system modeling and control. An important point is the use of rational agents and heuristic techniques to cope with the computational complexity required for controlling complex systems. Through this book, we also find navigation and vision algorithms, automatic handwritten comprehension and speech recognition systems that will be included in the next generation of productive systems developed by man.

**How to reference**

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Davide Bacciu, Andrea Bellandi, Barbara Furletti, Valerio Grossi and Andrea Romei (2008). Discovering Strategic Behaviors in Multi-Agent Scenarios by Ontology-Driven Mining, Advances in Robotics, Automation and Control, Jesus Aramburo and Antonio Ramirez Trevino (Ed.), ISBN: 978-953-7619-16-9, InTech, Available from:
http://www.intechopen.com/books/advances_in_robotics_automation_and_control/discovering_strategic_behaviors_in_multi-agent_scenarios_by_ontology-driven_mining

INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821