semantic data integration,
medical rule-based systems, uncertainty

Beata JANKOWSKA[1], Magdalena SZYMKOWIAK[2]

DESIGNING MEDICAL PRODUCTION RULES FROM SEMANTICALLY INTEGRATED DATA

In the paper an algorithm for automatic knowledge acquisition is proposed. The knowledge is acquired from aggregate data stored in different repositories. The algorithm operates by means of semantic data integration, allowing both syntax and semantic differences between data coming from different sources. If only we know data taxonomies, can interpret data schemas and design schema mappings, then the differences are not an obstacle to integration. The acquired knowledge is being defined in a form of production rules with uncertainty. The considerations are illustrated with medical examples.

1. INTRODUCTION

The problem of discovering knowledge that is hidden in evidence has been studied for many years. The diversity of solutions results from varying character of evidence and also from varying applications of the knowledge being discovered. The main attributes of evidence are the following ones: consistency or inconsistency of data stored, their syntactic homogeneity or heterogeneity, and their semantic uniqueness or ambiguity. Saying of knowledge application, we mean the necessity of designing knowledge that is certain or uncertain, universal or specific. These attributes influence both a method of data exploring and also a way of knowledge representation.

In the paper we propose an inductive learning algorithm for exploring such domain data that are stored in various tuple formats. One of the formats is chosen as a point of reference and all the mappings between this reference format and the remaining formats must be given. All the data must be based on similar ontologies (see subsection 3.1), and taxonomical dependencies between domain individuals must be known. The data can be incomplete (lacking attributes prove ignorance on attributes' values).

The algorithm is based on using a multisort algebra of domain individuals, with sorts corresponding to categories of individuals, a partial-order relation of subsumption, and union/ intersection operations being counterparts of classical operations from the set theory. The relation of subsumption enables expressing taxonomical dependencies between the sets of individuals; the union and intersection operations are used as well for defining mappings between different data schemas, as for implementing the algorithm's kernel. The algorithm's performance is illustrated with examples from the domain of medicine.

Related work. Inductive learning methods received a great deal of attention and were reported in many papers and surveys. Commonly known algorithms of discovering association rules [1, 2, 18, 6] are based on using factors of support and confidence - the most popular measures of significance and interest. The two factors do not occur in the association rules explicitly – they are hidden. Another kind of inductive learning is being used when designing rough set decision rules from data written in a form of attribute tables [13, 15]. The decision rules can be generated as certain or approximate ones. However, the level of uncertainty cannot be exposed, same as in the previous case of association rules. There are different algorithms of managing incomplete data, i.e. data with missing attribute values, when designing rough set decision rules [5].

Since early nineties, various data integration techniques have been developed very intensively. The proposed solutions are based on relational data model and SQL query language [10, 7, 4] or XML data

[1] Institute of Control and Information Engineering, Pl. Skłodowskiej-Curie 5, 60-965 Poznań, Poland, email: beata.jankowska@put.poznan.pl.
[2] Poznan University of Technology, Institute of Mathematics, ul. Piotrowo 3a, 60-965 Poznań, Poland email: magdalena.szymkowiak@put.poznan.pl.

model and XPath language [3, 12]. While it is true that the goal of such integration is obtaining "full data" only, yet these data present a valuable source of knowledge.

In the last decade, also ontological sciences developed very rapidly. There are many languages and many tools, e.g. [11, 17] that help us to correctly categorize or conceptualize domains – to understand the semantics of individuals, categories and various relations between them. They can be useful for processing heterogeneous data from the domain.

2. FORMAT OF PRODUCTION RULES

Production rules, used in rule-based systems (RBSs) for automatic reasonings, have a form of implications with premises and conclusions, and represent theses (classical RBSs) or hypotheses (RBSs with uncertainty). The knowledge based on experience (e.g. medical knowledge) is usually being expressed in a form of production rules with uncertainty, that are provided with two visible factors of reliability. Contrary to association rules, low values of these factors do not necessarily reduce production rules' quality. In case of absence of very reliable hypotheses, even a less reliable one can be useful (e.g. an initial diagnostic hypothesis made by a General Practitioner). Also, a low level of the conclusion's reliability does not forejudge a low quality of the rule as a whole (e.g. a hypothesis on frequency of side effects of treatment). The discussed production rules take the form (2):

$$\begin{aligned} &\text{it happens with grf} = p_3 : \\ &\quad \text{if } P_1, \ldots, P_n \\ &\quad \text{then C with irf} = p_4, \end{aligned} \tag{2}$$

where $P_1, \ldots, P_n$ stand for premises, and C – for an uncertain conclusion. The two factors grf and irf are being calculated by means of probabilistic and statistician methods. They stand for: grf – global rule's reliability, determining the priority of the rule in comparison to other rules from the knowledge base of RBS, irf – internal rule's reliability, defining probability of the conclusion C given the certain occurrence of the premises $P_1, \ldots, P_n$. An actual reliability of the conclusion C is usually calculated from the formula (3) [9]:

$$p = p_3 \cdot \min \{p(P_1), \ldots, p(P_n)\} \cdot p_4 \tag{3}$$

where $p(P_i)$, $i = 1, \ldots, n$ stands for the probability of occurrence of the premise P_i.

3. DESIGNING PRODUCTION RULES

The most valuable production rules can be obtained neither by book learning nor by knowledge acquisition from domain experts, but by means of global data mining. We mean here as well individual data stored in distributed databases as aggregate data stored in special repositories. Considering that the last data are general in form, we will only show how to design production rules with uncertainty from the aggregate data from repositories. We will allow heterogeneous data schemas, incompleteness of data specification, and also dynamic changes of data values. In order to make the whole process automated, we will put some constraints on the format of data representation.

3.1. FORMAT OF EVIDENCE

We will assume that each data in a repository represents a fact relating to a number of people, things or events (e.g. fact specifying a result of clinical trials performed for a number of patients). The number usually varies from 10 to 200. The data inside the repository are homogeneous: they differ neither in syntax, nor in semantics. All of them are built according to the same tuple schema, determined by one of domain ontologies. Repositories can differ one from another in tuple schemas used. However, the following constraint must be fulfilled: all domain ontologies addressed by the schemas must be based on the same set of domain individuals.

For the needs of our algorithm for designing production rules with uncertainty, we will set some tuple schema as a reference one. It will comprise a wide range of attributes characterizing people, things

or events being the object of the considered data tuples. We will allow each data tuple (in short – tuple) to be incomplete, i.e. some tuple's attributes can have their values unknown. The above attributes will be invisible in specification. Apart from the 'attribute value', each visible attribute has an additional 'attribute count' parameter, being an equivalent of aggregating 'count' operators from query languages. Besides, in each tuple a nonempty subset of common-key attributes has to be defined. They are the attributes of primary importance for the tuple. The common-key attributes must be attributes of the whole group of people, things or events "caught" in the tuple. The remaining visible attributes belong to a set of common-not-key ones or to a set of discriminatory ones. All common-key and common-not-key attributes have their 'attribute count' parameters identical and maximal in the tuple (N). In discriminatory attributes, their 'attribute count' parameters take values from the range $<0;N)$. The discussed tuple format is universal enough to represent also data of an individual person, thing or event.

Symbolically, a tuple T_i, of a given reference schema S, can be written as (4):

$$T_i = <A_1{:}v©_{i1}/N_i, ..., A_m{:}v©_{im}/N_i , a_1{:}v©_{i(m+1)}/N_i, ..., a_n{:}v©_{i(m+n)}/N_i , b_1{:}v©_{i(m+n+1)})/q_{i1}, ..., b_p{:}v©_{i(m+n+p)}/q_{ip}> \quad (4)$$

where $A_1, ..., A_m, a_1, ..., a_n, b_1, ..., b_p$ belong to the set of attributes characteristic of the reference schema S; $A_1, ..., A_m$ stand for attributes that are common-key in the tuple T_i; $a_1, ..., a_n$ – for attributes that are common-not-key in the tuple T_i; $b_1, ..., b_p$ – for attributes that are discriminatory in the tuple T_i; $v©_{i1}, ..., v©_{i(n+m+p)}$ – for 'attribute values' of all the attributes mentioned above; $N_i, q_{i1}, ..., q_{ip}$ – for 'attribute counts' of the attributes in the tuple T_i. In each pair $v©_{ij}$, for $1 \le j \le n+m+p$, v stands for a value set, and © – for so called 'value qualifier', taking a form of: $\odot$ – for a set symbolizing the conjunction of elementary values, and $\oplus$ – for a set symbolizing the disjunction of elementary values. As it was said before, attributes taking a value "any" are not being shown (we mean here: a value set consisting of all elementary values from the attribute's domain – in case of the qualifier $\oplus$, and an empty set – in case of the qualifier $\odot$).

Let us remark that, depending on the semantics of tuples, the same attributes can occur in a set of common ones in some tuples, and in a set of discriminatory ones – in others. For example, let us imagine an experiment testing the dependency of addictions in adults on environmental and genetic factors. In a tuple specifying basic experiment's results, the attribute 'addictions' would be a discriminatory one. For a change, in an "opposite" tuple specifying reasons for the addictions, this attribute would be a common one.

In case of heterogeneous evidence, all data should be transformed to the chosen reference schema at the beginning. The transformation should be based on mappings between data schemas [12].

3.2. ALGEBRA OF AGGREGATE DATA

As it was considered in [8], aggregate data and operations performed on these data can be given an algebraic-taxonomical interpretation. First, each attribute A has its model in a form of the qualified power set of a domain $qVAL_A = 2^{VAL_A} \times \{\oplus, \odot\}$. For the attribute A, a simple algebra S_A can be defined, with a partial order relation $\subseteq_{qA}$, an operation of union $\cup_{qA}$, and an operation of intersection $\cap_{qA}$, being defined on the set $qVAL_A$. The algebra S_A is a lattice.

Next, an aggregate data built under a schema $S = (A_1,..., A_m, a_1,..., a_n, b_1,..., b_p)$ can be seen as a list of qualified set values, being subsets of $qVAL_{A1}$, ..., $qVAL_{Am}$, $qVAL_{a1}$, ..., $qVAL_{an}$, $qVAL_{b1}$, ..., $qVAL_{bp}$, with their countings (an ordered pair (vq_A, c_A) is being written for simplicity as $v©_A/c_A$, as it is in (4)). For the schema S, an algebra $A_S = (VAL_S, \subseteq_{cS}, \cup_{cS}, \cap_{cS})$, being a product of similar algebras-lattices Sc_{A1}, …, Sc_{Am}, Sc_{a1}, …, Sc_{an}, Sc_{b1}, …, Sc_{bp} can be defined. The algebra A_S is also a lattice.

Let D_0 stands for a set of domain data matching the reference schema S. In the light of the above comments, the data from D_0 can be compared, joined or processed in any other way by means of the partial order relation $\subseteq_{cS}$, and the operations of union $\cup_{cS}$ and intersection $\cap_{cS}$. As it was shown in [8], the two operations $\cup_{cS}$ and $\cap_{cS}$ are necessary and sufficient for transforming any data matching a reference schema S_i compatible with S (based on the same set of individuals, but possibly different from S) into its counterpart from D_0. As a consequence, it is finally possible to define an algebra-lattice for the set D of all data compatible with data from D_0. Formally, it has the following form (5):

$$A_D = (D, \subseteq c_S, \cup c_S, \cap c_S). \quad (5)$$

3.3. ALGORITHM FOR DESIGNING PRODUCTION RULES

Now, let us demonstrate how from tuples of a fixed schema we come to production rules with uncertainty (in case of heterogeneous evidence, the generalized approach should be used – see the subsection 3.2). In each pass of the algorithm, an appointed initial tuple will be joined with each other tuple that ”preserves” values of all required attributes from the initial tuple. Among the required attributes are: a number of ”if” attributes (containing all common-key and some chosen common-not-key attributes) and one ”then” attribute (being a chosen discriminate or common-not-key attribute). In order to meet this requirement, all corresponding ’attribute values’ of ”if” and “then” attributes from the initial one and attached tuples have to be in an appropriate relation $\subseteq q_A$ (the index A depends on the considered attribute: not its role, but its name). What is more, for the chosen “then” attribute, in case if the corresponding attribute from the attached tuple is a discriminate one, the both ’attribute values’ have to be identical, i.e. an appropriate relation $\subseteq q_A$ should hold in both directions. The relation $\subseteq q_A$ holds:

- for sets symbolizing conjunctions of values – if and only if ’attribute value’ from the attached tuple is a superset of ’attribute value’ from the initial tuple;
- for sets symbolizing disjunctions of values – if and only if it is its subset.

For the remaining common-not-key and discriminate attributes from the initial tuple (the ones not chosen as ”if” or ”then” attributes), updates of ’attribute values’ are being performed. Such the update consists in:

- calculating a union of sets by means of an appropriate $\cup q_A$ operation – in case if the current tuple’s attribute is a common-not-key one and the attached tuple’s attribute is a common-key or common-not-key one, and the both ’attribute values’ symbolize disjunctions of values;
- calculating an intersection of sets by means of an appropriate $\cap q_A$ operation – in case if the current tuple’s attribute is a common-not-key one and the attached tuple’s attribute is a common-key or common-not-key one, and the both ’attribute values’ symbolize conjunctions of values;
- preserving the ’attribute value’ of the current tuple’s attribute – in case if:
    - this attribute is a common-not-key one and the attached tuples’ attribute is a discriminate one, and their ’attribute values’ are equal;
    - this attribute is a discriminate one and the attached tuples’ attribute is a common-not-key one, and the relation $\subseteq q_A$ between the two ’attribute values’ holds;
    - the both attributes are discriminate ones, and their ’attribute values’ are equal;
- setting a value ”any” to the current tuple’s attribute and, next, removing this attribute from the tuple – in other cases.

Let us note that, while integrating, initial sharp ’attribute values’ of common-not-key attributes not chosen as ”if” or ”then” ones are often being fuzzified. As concerns ’attribute counts’ parameters, their values increase while integrating. The values are calculated by means of the classical sum operator, applied to corresponding ’attribute counts’ from the initial one and all attached tuples.

A production rule designed from the set of described tuples has a form of an uncertain implication, with:

- a number of premises, corresponding to the attributes being common-key or common-not-key in the final, integrated tuple (except the one chosen as ”then” at the beginning, in case it is a common-not-key attribute in this tuple), and
- one conclusion, corresponding to the attribute chosen as ”then” at the beginning.

The algorithm for designing production rules from a homogeneous data repository **T** can be formally defined as follows:

For each tuple T_i from the homogeneous data repository **T**, with a set of common-key attributes $\mathbf{K_i} = \{A_{i1}, A_{i2}, \ldots, A_{im}\}$, a set of common-not-key attributes $\mathbf{U_i} = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$, and a set of discriminate attributes $\mathbf{S_i} = \{b_{i1}, b_{i2}, \ldots, b_{ip}\}$, do the following:
{ for each subset $\mathbf{E_{ij}}$ of the set $\mathbf{U_i}$, including the empty and full ones, do the following actions:
{ for any attribute f_{ijk} from the set $\mathbf{U_i \backslash E_{ij}} \cup \mathbf{S_i}$:
{ $T_{curr} = T_i$;

perform an integration $\mathbf{K}_i$–$\mathbf{E}_{ij}$–f_{ijk} of the tuple T_{curr} with each other tuple T_h, having a set of common-key attributes $\mathbf{K_h} = \{A_{h1}, A_{h2}, \ldots, A_{ht}\}$, a set of common-not-key attributes $\mathbf{U_h} = \{a_{h1}, a_{h2}, \ldots, a_{hu}\}$, and a set of discriminate attributes $\mathbf{S_h} = \{b_{h1}, b_{h2}, \ldots, b_{hv}\}$, and fulfilling the constraints:

1. for each $A_{ik} \in \mathbf{K_i}$, k=1,…, m, it exists $d \in <1; t>$, such that $A_{ik} = A_{hd}$ and $pres(T_i, T_h, A_{ik})$,
2. for each $a_{ik} \in \mathbf{E_{ij}}$, it exists $d \in <1; t>$, such that $a_{ik} = A_{hd}$ and $pres(T_i, T_h, a_{ik})$ or it exists $d \in <1;u>$, such that $a_{ik} = a_{hd}$ and $pres(T_i, T_h, a_{ik})$,
3. for the attribute f_{ijk} it exists $d \in <1;u>$, such that $f_{ijk} = a_{hd}$ and $pres(T_i, T_h, f_{ijk})$ or it exists $d \in <1;v>$, such that $f_{ijk} = b_{hd}$ and $pres(T_i, T_h, f_{ijk})$ and $pres(T_h, T_i, f_{ijk})$;

carry it out according to the schema:
$T_{curr} = integs(T_{curr}, T_h, \mathbf{K_i} \cup \mathbf{E_{ij}} \cup \{f_{ijk}\})\}$;

formulate a production rule $r_{ijk} = forms(T_{curr}, \mathbf{K_i} \cup \mathbf{U}_{curr} \backslash f_{ijk}, f_{ijk})\}$,
where $\mathbf{U}_{curr}$ stands for a set of all visible, common-not-key attributes in the final integrated tuple T_{curr} .

Algorithm 1. Designing production rules from data in homogeneous repository.

Let **F** stand for the set of all attributes used in the tuples from **T**. The semantics of the function: pres: **T**×**T**×**F** → {true, false}, intended for checking if 'attribute values' are preserved, can be defined as follows:
if $v©_i$ and $v©_h$ are values of the same attribute A in both tuples T_i and T_h, and the relation $\subseteq q_A$ is appropriate to describe the attribute A, then it holds (6):

$$pres(T_i, T_h, A) = \begin{cases} \text{true, if } v©_i \subseteq q_A \; v©_h, \\ \text{false, in opp. case .} \end{cases} \tag{6}$$

The function integs : T×T×2F → T calculates the result of integrating the first tuple with a new one, under the subset-criterion of "if" and "then" attributes from the first tuple. The function forms: T×2F×F → T generates a production rule with uncertainty for: a given integrated tuple, a proper subset of its attributes, and one additional tuple's attribute, not belonging to the subset mentioned before. It results in a full text of the wanted production rule – if it can be designed; an empty string ' ' – in the opposite case. The function forms is based on using an auxiliary function gtext, being a concatenation of varying numbers of parameters of string or integer types. For lack of place, the functions are not defined in detail.

3.4. EXAMPLES OF DESIGNING PRODUCTION RULES

In order to illustrate the algorithm operation, let us consider a few examples of integrating homogeneous medical data and, next, generating production rules from the obtained integrated results. The data come from a medical repository, namely – the repository of CTRs (Clinical Trials Registers), storing reports of different clinical experiments. Each data represents an aggregate report of one clinical trial, carried out for verifying a medical hypothesis on a group numbering from 10 to 200 participants. It has a form of the tuple (4), with attributes specifying guidelines (personal and clinical features of participants, treatment rules) and results (values of clinically essential outcomes) of the trial. The data in the repository differ neither in syntax, nor in semantics. All of them are built according to the same tuple schema. All the data considered in this paper refer to young patients, hospitalized (in different time periods and different clinics) for the reason of bronchial asthma exacerbation [14].

Table 1. Three exemplary tuples reporting experiments on treating children with asthma exacerbation. Asterisks (*) point to the attributes assumed as common-key in the particular tuples.

attribute names	attribute values – T_1	attribute values – T_2	attribute values – T_3
general_diagnosis	*{pediatric_asthma}⊙/17	*{pediatric_asthma}⊙/18	*{pediatric_asthma}⊙/89
current_health_state	*{acute_asthma_exacerb}⊙/17	*{acute_asthma_attack}⊙/18	*{acute_asthma_exacerb}⊙/89
standard_drug	*{short-act_B2_agonist}⊙/17	*{short-act_B2_agonist}⊙/18	*{short-act_B2_agonist}⊙/89
additional_drug	*{inhaled_anticholin_md}⊙/17	*{inhaled_anticholin_md}⊙/18	*{inhaled_anticholin_md}⊙/89
co_intervention	{systemic_corticosteroid}⊙/17	*{systemic_corticosteroid}⊙/18	{no_corticosteroid}⊙/89
age_range	{3, …,17}⊕/17	{4, …,15}⊕/18	{1, …,18}⊕/89
severity_of_diagn_illness	{mild, moderate}⊕/17	{moderate}⊕/18	{mild, moderate, severe}⊕/89
symptoms	{coughing}⊙/17	{coughing, wheezing}⊙/18	{coughing}⊙/89
treatment_effects	{no_hospital_admission}⊙/16	{no_hospital_admission, stability_of FEV1}⊙/18	{no_hospital_admission}⊙/77
adverse_effects	{vomiting}⊙/3	{vomiting}⊙/4	{vomiting, shivering}⊙/5
relapse	{next_asthma_exacerb_ in_72_hours}⊙/1		

The first data (T_1) shown in the Table 1 reports the result of a clinical trial carried out on a group of 17 patients. Let us denote a set of common-key attributes of the tuple T_1 by $\mathbf{K_1}$, a set of its common-not-key attributes by $\mathbf{U_1}$, and a set of its discriminate attributes by $\mathbf{S_1}$. They are as follows:

$\mathbf{K_1}$ = {general_diagnosis, current_health_state, standard_drug, additional_drug},
$\mathbf{U_1}$ = {co_intervention, age_range, severity_of_diagn_illness, symptoms},
$\mathbf{S_1}$ = {treatment_effects, adverse_effects, relapse}.

Only two from among the listed attributes (age_range, severity_of_diagn_illness) have their values in a form of value sets symbolizing disjunctions of elements. All the remaining attributes are assigned values in a form of value sets symbolizing conjunctions of elements.

The next data (T_2 and T_3) refer to 18–person and 89–person groups of patients, respectively. Let us try to integrate the initial tuple T_1 with the tuples T_2 and T_3, according to the subset-criterion (the third parameter of the function integs), in sequence:

(a) $\mathbf{C_{1\text{-}a}} = \mathbf{K_1} \cup$ {age_range, symptoms} $\cup$ {adverse_effects};
(b) $\mathbf{C_{1\text{-}b}} = \mathbf{K_1} \cup$ {co_intervention, severity_of_diagn_illness} $\cup$ {treatment_effects};
(c) $\mathbf{C_{1\text{-}c}} = \mathbf{K_1} \cup \{\} \cup$ {symptoms}.

It turns out, that in the first two cases (a) and (b), the initial tuple T_1 can be integrated with the tuple T_2, but cannot be integrated with the tuple T_3. In the both cases, 'attribute values' of all the "if" attributes from the tuple T_1 subsume 'attribute values' of the corresponding attributes from the tuple T_2 (fulfilling the constraints formulated in 1. and 2. points of the proposed algorithm 1. As far as the relation is clear for most of these attributes (e.g. {3,...,17}⊕ subsumes {4,...,15}⊕, {mild, moderate}⊕ subsumes {moderate}⊕, and {coughing}⊙ subsumes {coughing, wheezing}⊙), it can be questionable for the common-key attribute current_health_state. The fact: {acute_asthma_ exacerbation}⊙ subsumes {acute_asthma_attack}⊙ results from the following taxonomical dependency between the value sets' elements: acute_asthma_exacerbation ≤ acute_asthma_attack.

Besides, as it is required, in the case (a) – the 'attribute value' of the chosen "then" attribute from the tuple T_1 (adverse_effects) is equal to the corresponding 'attribute value' from the tuple T_2 (fulfilling the constraint formulated in the second clause of 3. point of the algorithm); and, in the case (b) – the 'attribute value' of the chosen "then" attribute from the tuple T_1 (treatment_effects) subsumes the corresponding 'attribute value' from the tuple T_2, {no hospital admission}⊙ subsumes {no_hospital_ admission, stability_of_FEV1}⊙ (fulfilling the constraint formulated in the first clause of 3. point of the algorithm). The integration of the initial tuple T_1 with the tuple T_3 under the subset-criterion (a) is impossible both for the sake of not holding the subsumption between ranges {3,...,17}⊕ and {1,...,18}⊕, being 'attribute values' of age_range; and for the sake of not holding equality between the 'attribute value' {vomiting}⊙ of the chosen "then" attribute adverse_effects from the tuple T_1 and the corresponding 'attribute value' {vomiting, tremor}⊙ from the tuple T_3. In turn, 'attribute values' of co_intervention and severity_of diagn_illness stand on the way of integration T_1 with T_3 under the subset-criterion (b). As relates to the attribute co_intervention, the individuals systemic_corticosteroid and no_corticosteroid define two different, mutually exclusive methods of pharmacological treatment. Obviously, they cannot be in the taxonomical dependency. As a consequence, the required constraint: {systemic corticosteroid}⊙ subsumes {no corticosteroid}⊙ does not hold.

Table 2. The results of integrating the tuple T_1 with the T_2 ($T_{1\text{-}2}$), and the tuple T_1 with the tuple T_2 and T_3 ($T_{1\text{-}2\text{-}3}$).

attribute names	attribute values – $T_{1\text{-}2}$	attribute values – $T_{1\text{-}2\text{-}3}$
general_diagnosis	*{pediatric_asthma}⊙ /35	*{pediatric_asthma}⊙ /124
current_health_state	*{acute_asthma_exacerb}⊙ /35	*{acute_asthma_exacerb}⊙ /124
standard_drug	*{short-act_B2_agonist}⊙ /35	*{short-act_B2_agonist}⊙ /124
additional_drug	*{inhaled_anticholin_md}⊙ /35	*{inhaled_anticholin_md}⊙ /124
co_intervention	{systemic_corticosteroid}⊙ /35	
age_range	{3 , …,17}⊕/35	{1 , …,18}⊕/124
severity_of_diagn_illness	{mild, moderate}⊕/35	{mild, moderate, severe}⊕/124
symptoms	{coughing}⊙ /35	{coughing}⊙ /124
treatment_effects	{no_hospital_admission}⊙ /34	{no_hospital_admission}⊙ /111
adverse_effects	{vomiting}⊙ /7	

It can be easily proved that, in case if integration is possible, the result of tuple integration does not depend on the subset-criterion of integration, but only on the form of tuples being integrated. That is why, in the both cases (a) and (b), we will obtain the same final result of integration $T_{1\text{-}2}$ (Table 2).

After having done a similar attributes' analysis for the case (c), we come to the conclusion that, under this subset-criterion, the initial tuple T_1 can be integrated with the both tuples T_2 and T_3, giving the final result $T_{1\text{-}2\text{-}3}$ (Table 2).

Let us shortly summarize the main features of the algorithm presented in the subsection 3.3:

- the result of a single integration, being done in one algorithm's pass, depends strongly on the initial tuple T_i (that is why, the trials of integrating: the initial tuple T_1 with T_2 and T_3; the initial tuple T_2 with T_1 and T_3; and the initial tuple T_3 with T_1 and T_2 under the same criterion (a), end with the final integrated tuples $T_{1\text{-}2}$, T_2 and T_3, respectively, that are different one from each other);
- the order in which the initial tuple T_i attempts to join with remaining tuples has no influence on the final result of integration (the final tuples $T_{1\text{-}2\text{-}3}$ and $T_{1\text{-}3\text{-}2}$ obtained under the criterion (c) would be equal; that is why, the joining is always being performed in a predetermined order);
- in case if integration of the initial tuple T_i with some other tuples is possible, the final result of this integration does not depend on the criterion of integration (that is why, the final tuples $T_{1\text{-}2}$ obtained under the different criterions (a) and (b) are equal);
- for a given set of homogeneous tuples – a great number of integrated results will be obtained; however, it is not guaranteed that the results will be different one from each other.

Table 3. Production rules obtained from the integrated tuples $T_{1\text{-}2}$ ($r_{1\text{-}2,1}$, $r_{1\text{-}2,2}$), and $T_{1\text{-}2\text{-}3}$($r_{1\text{-}2\text{-}3,1}$).

production rule $r_{1\text{-}2,1}$	production rule $r_{1\text{-}2,2}$	production rule $r_{1\text{-}2\text{-}3,1}$
it happens with grf = 0.80 : if (pediatric_asthma) and (acute_asthma_exacerb) and (short-act_B2_agonist) and (inhaled_anticholin_md) and (systemic_corticosteroid) and (age_range = [3 ; 17]) and (severity_of_diagn_illness = (mild or moderate)) and (coughing) Then (vomiting) with irf=0.20	it happens with grf = 0.89 : if (pediatric_asthma) and (acute_asthma_exacerb) and (short-act_B2_agonist) and (inhaled_anticholin_md) and (systemic_corticosteroid) and (age_range = [3 ; 17]) and (severity_of_diagn_illness = (mild or moderate)) and (coughing) then (no_hospital_admission) with irf=0.97	it happens with grf = 0.89 : if (pediatric_asthma) and (acute_asthma_exacerb) and (short-act_B2_agonist) and (inhaled_anticholin_md) and (age_range = [1 ; 18]) and (severity_of_diagn_illness = (mild or moderate or severe)) and then (coughing) with irf=0.90

For any final integrated tuple T, we can generate a production rule with uncertainty $r = forms(T,P, f)$, such that P is a set of all common (optionally except one common-not-key) attributes from the tuple T, and f is the chosen common-not-key or discriminate attribute from this tuple. For example, for the final integrated tuple $T_{1\text{-}2}$, the set $P_{1\text{-}2}$ of all its common attributes, and its discriminate attributes $f_{1\text{-}2,1}$ = adverse_effects and $f_{1\text{-}2,2}$ = treatment_effects – production rules will take forms, respectively $r_{1\text{-}2,1}$ and $r_{1\text{-}2,2}$ (Table 3). Again, for the final integrated tuple $T_{1\text{-}2\text{-}3}$, the set $P_{1\text{-}2\text{-}3,1}$ of all its common attributes except the attribute symptoms, being the third argument of the function forms - a production rule will take a form $r_{1\text{-}2\text{-}3,1}$ (Table 3). The values of the factors grf and irf have been calculated according to the schema proposed in [16].

4. CONCLUSIONS

From the point of view of the expert system's efficiency, a knowledge base is the system's bottleneck. It is commonly known that experts in the domain are usually unwilling to co-operate with knowledge engineers. Also, there are objective difficulties in their communicating with one another. The mentioned problems are serious arguments for appreciating real evidence as the very valuable source of domain knowledge, that can be explored one way or another.

The presented algorithm for automatic knowledge acquisition from a number of repositories of aggregate tuples is based on semantic data integration. Allowing syntax and semantic differences between input tuples, the algorithm can operate on data coming from various sources. This way, it can give reliable results. After a small adaptation, we could obtain such production rules with uncertainty that might be ready for use in knowledge bases designed by expert system shells like FuzzyCLIPS or Jess.

The algorithm's complexity depends polynomially on the number of aggregate tuples in repositories. Besides, it depends exponentially on the maximum number max_A of not-key attributes occurring in these tuples. Assuming that this number is not high (at most – about 10), we can regard $c = 2^{max_A}$ as a constant factor of the algorithm's complexity $O(cn^2)$.

ACKNOWLEDGEMENTS

The research has been partially supported by the Polish Ministry of Science and Higher Education under grant N516 369536 and by Poznan University of Technology under grants PB 43-070/10 and BW 45-087/10.

BIBLIOGRAPHY

[1] AGRAWAL R., IMIELINSKI T., SWAMI A., Mining Association Rules Between Sets of Items in Large Databases, SIGMOD Record 22(2), 1993, pp. 805-810.
[2] AGRAWAL R., SRIKANT R., Fast algorithms for mining association rules in large databases, in: Bocca J.B., Jarke M., Zaniolo C. (Eds.), Proc. of the 20th Int. Conference on Very Large Data Bases, Morgan Kaufmann, Santiago de Chile, 1994, pp. 487-499.
[3] ARENAS M., LIBKIN L., XML Data Exchange: Consistency and Query Answering, in: LI Ch. (Ed.), Proc. of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM, Baltimore, 2005, pp.13-24.
[4] CHEN H., Rewriting Queries Using View for RDF/RDFS-Based Relational Data Integration, LNCS 3816, 2005, pp. 243-254.
[5] GRZYMALA-BUSSE J., GRZYMALA-BUSSE W., An experimental comparison of three rough set approaches to missing attribute values, Trans. on Rough Sets 6, 2007, pp. 1-47.
[6] HAN J., PEI J., YIN Y., MAO R., Mining frequent patterns without candidate generation, Data Mining and Knowledge Discovery 8, 2004, pp. 53-87.
[7] HAAS L.M., HERNANDEZ M.A., HO H., POPA L., ROTH M., Clio grows up: from research prototype to industrial tool, in: Özcan F. (Ed.), Proc. of ACM SIGMOD Conference on Management of Data, ACM, Baltimore, 2005, pp. 805-810.
[8] JANKOWSKA B., On Integrating Medical Data by means of an Algebraic Lattice, in: Informatyczne systemy zarządzania wiedzą, Akademicka Oficyna Wydawnicza EXIT, (in Polish).
[9] JANKOWSKA, B., SZYMKOWIAK, M.: How to Acquire and Structuralize Knowledge for Medical Rule-Based Systems?, in: Studies in Computational Intelligence, Kacprzyk J. (ed.), Springer Series, Berlin /Heidelberg, 2008, pp. 115-130
[10] MILLER R.J. et al., The Clio Project: Managing Heterogeneity, SIGMOD Record 30(1), 2001, pp. 77-83.
[11] OWL Web Ontology Language, http://www.w3.org/TR/2004/REC-owl-features-20040210/, 2010.
[12] PANKOWSKI T., XML data integration in SixP2P - a theoretical framework, in: Doucet A., Gançarski S., Pacitti E., (Eds.), Proc. of the 2008 Int. Workshop on Data Management in Peer-to-Peer Systems, ACM Series, Nantes, 2008, pp. 11-18.
[13] PAWLAK Z., Rough sets, Int. Journal of Information & Computer Science 11, 1982, pp. 341-356.
[14] PLOTNICK L.H., DUCHARME F.M., Combined inhaled anticholinergics and beta2-agonists for initial treatment of acute asthma in children, in: The Cochrane Library, 2005.
[15] PREDKI B., SLOWINSKI R., STEFANOWSKI J., SUSMAGA R., WILK Sz., ROSE. Software Implementation of the Rough Set Theory, LNCS 1424, 1998, pp. 605-608.
[16] SZYMKOWIAK, M., JANKOWSKA, B.: Reliability of Medical Production Rules Obtained by means of Aggregate Data Mining, submitted to the conf. MIT 2010.
[17] The Protégé Ontology Editor, http://protege.stanford.edu/, 2010.
[18] ZAKI M.J., Scalable algorithms for association mining, IEEE Trans. on Knowledge and Data Engineering 12(3), 2000, pp.372-390.