# Unified Algorithm for Undirected Discovery of Exception Rules

Einoshin Suzuki[1] and Jan M. Żytkow[2]

[1] Electrical and Computer Engineering, Yokohama National University, 79-5 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan.
suzuki@dnj.ynu.ac.jp
[2] Computer Science Department, UNC Charlotte, Charlotte, N.C. 28223.
zytkow@uncc.edu

**Abstract.** This paper presents an algorithm that seeks every possible exception rule which violates a common sense rule and satisfies several assumptions of simplicity. Exception rules, which represent systematic deviation from common sense rules, are often found interesting. Discovery of pairs that consist of a common sense rule and an exception rule, resulting from undirected search for unexpected exception rules, was successful in various domains. In the past, however, an exception rule represented a change of conclusion caused by adding an extra condition to the premise of a common sense rule. That approach formalized only one type of exceptions, and failed to represent other types. In order to provide a systematic treatment of exceptions, we categorize exception rules into eleven categories, and we propose a unified algorithm for discovering all of them. Preliminary results on fifteen real-world data sets provide an empirical proof of effectiveness of our algorithm in discovering interesting knowledge. The empirical results also match our theoretical analysis of exceptions, showing that the eleven types can be partitioned in three classes according to the frequency with which they occur in data.

**Keywords**: Exception/Deviation Detection, Rule Discovery, Exception Rule, Rule Triplet

## 1 Introduction

Exceptions and/or deviations, which focus on a very small portion of a data set, have long been ignored or mistaken as noise in machine learning. The goal of data mining is broader, however. Exceptions were always interesting to discoverers, as they challenged the existing knowledge and often led to the growth of knowledge in new directions. In addition to predictions, decision optimization is important in data mining [5]. We strongly believe that exception and/or deviation can improve the quality of decisions, and their detection deserves more attention.

An increasing number of studies is devoted to exception/deviation detection. Examples of such studies are outlier discovery [4], OLAP operator for explaining increase/decrease of a continuous attribute [7], and exception rule discovery [6, 8, 10–13].

Exception rules are typically represented as deviational patterns to common sense rules of high generality and accuracy. Exception rule discovery can be classified as either directed [6, 8] or undirected [10–13]. A directed method finds a set of exception rules which deviate from the given common sense rules. In distinction, common sense rules are not given to an undirected method, which finds a set of pairs of a common sense rule and an exception rule. The advantage of an undirected method is that it can discover highly unexpected patterns since it also discovers common sense rules [14].

In undirected discovery of exception rules two natural questions arise: "what other kinds of exception rules can be defined?" and "is there an efficient algorithm to discover all of them?" In this paper we give a constructive answer to both questions. We first define all kinds of exception rules that occur in situations described with three literals. Next we present an efficient algorithm, which is not restricted to binary attributes and thus can be applied to a broad range of ordinary data sets. We finally demonstrate the effectiveness of our approach with experiments using fifteen data sets.

## 2 Categories of Exception Rules

### 2.1 A Rule and a Negative Rule

Let a data set contain $n$ examples, each expressed by $m$ attributes. Let a literal be a conjunction of atoms, while an atom is either a value assignment for a nominal attribute or a range assignment for a continuous attribute. An atom can be also a missing value assignment for any attribute.

In this paper, we define a rule as $u \rightarrow v$, where $u$ and $v$ are literals. We follow the definition of ITRULE [9] and define the generality condition and the accuracy condition of $u \rightarrow v$ as the right-hand sides of (1), where $\theta_{\mathrm{S}}$ and $\theta_{\mathrm{F}}$ are thresholds given by the user, and $\widehat{\Pr}(u)$ is the ratio of examples that satisfy $u$ in the data set:

$$u \rightarrow v \Leftrightarrow \widehat{\Pr}(u) \geq \theta_{\mathrm{S}} \text{ (generality) } \& \ \widehat{\Pr}(v|u) \geq \theta_{\mathrm{F}} \text{ (accuracy)} \tag{1}$$

In association rule discovery[1], support $\widehat{\Pr}(uv)$ is used instead of $\widehat{\Pr}(u)$, where $uv$ represents $u \wedge v$, but the idea of generality is essentially the same.

In this paper, we also introduce a negative rule as $u \not\rightarrow v$, where $\theta_{\mathrm{I}}$ is a threshold given by the user.

$$u \not\rightarrow v \Leftrightarrow \widehat{\Pr}(u) \geq \theta_{\mathrm{S}} \ \& \ \widehat{\Pr}(v|u) \leq \theta_{\mathrm{I}} \tag{2}$$

### 2.2 Rule Triplets

Let $y$ and $z$ be literals, and $x$ and $x'$ be atoms with the same attribute but with a different value. Suzuki [11] has considered the discovery of rule triplets that consists of a common sense rule $y \rightarrow x$, an exception rule $yz \rightarrow x'$ and a reference rule $z \not\rightarrow x'$. Representation of discovered rules is given by

$$(y \rightarrow x,\ yz \rightarrow x',\ z \not\rightarrow x') \tag{3}$$

This pattern can be interpreted as "If $y$ and $z$ then $x'$ and not $x$. This is an interesting exception since usually if $y$ then $x$, and if $z$ then not frequently $x'$."

This kind of pattern holds together several pieces of knowledge, including exception rules as unexpected, surprising, anomalous and thus interesting additions to other rules.

We use the term "common sense rule" because it well-represents a user-given belief in the direct method of search for exceptions, where common sense rules are given in the input. In undirected search, however, there may be little common sense in "common sense rules," as we are concerned with relations between rules, and we do not deal with user's knowledge.

We will now seek a systematic generation of patterns similar to (3), starting from the simplest cases. We formalize the situation as a rule triplet, where a common sense rule is represented by $y \rightarrow x$, and an exception rule and a reference rule are represented by a negative rule $\alpha \not\rightarrow \beta$ and a rule $\gamma \rightarrow \delta$ respectively.

$$t(y, x, \alpha, \beta, \gamma, \delta) = (y \rightarrow x,\ \alpha \not\rightarrow \beta,\ \gamma \rightarrow \delta) \tag{4}$$

Four meta-level variables $\alpha$, $\beta$, $\gamma$, and $\delta$ can be instantiated in different ways, with the use of literals $x$, $y$, and $z$, to form different specific patterns of exceptions, analogous to (3). This rule triplet has a generic reading "it is common that if $y$ then $x$, but we have exceptions that if $\alpha$ then not frequently $\beta$. This is surprising since if $\gamma$ then $\delta$". Note that this terminology differs from (3) in that an exception rule and a reference rule are represented by a negative rule and a rule respectively. Of course, that meta-level reading does not communicate any specific exception, anomaly and surprise, but we will seek those features in different instances of the pattern. Here we only justify this definition by stating that a violation (exception rule) of two rules (common sense rule and reference rule) can be interesting.

In order to systematically categorize exception rules, we restrict our attention to rule triplets with three free literals $x, y, z$. Number three is chosen since it represents the simplest situation which makes sense: a rule triplet with two literals would likely to be meaningless since it tends to be overconstrained, and a rule triplet with more than three literals would likely to be more difficult to interpret.

We assume that a conjunction of two literals can appear only in the premise of an exception rule. This restriction is justified because a conjunction of two literals in the premise makes a good candidate for an exception to a rule that holds one of those literals in the premise.

$$\beta, \gamma, \delta \in \{x, y, z\} \tag{5}$$

$$\alpha \in \{x, y, z, xy, yz, zx\} \tag{6}$$

Since a literal $z$ is not contained in a common sense rule, it must occur both in an exception rule and in a reference rule, otherwise a triplet situation will reduce to the case of two literals $x$ and $y$. Possible candidates for a reference rule are then restricted to $(\gamma, \delta) \in \{(y,z),(z,y),(z,x),(x,z)\}$. If we pair each of these four with the common sense rule $y \to x$, we realize that the case of $(\gamma, \delta) = (x,z)$ is equivalent (isomorphic) to $(\gamma, \delta) = (z,y)$, one case can be produced from the other by renaming the variables. Therefore, we neglect $(\gamma, \delta) = (x,z)$ in this paper, so that

$$(\gamma, \delta) \in \{(y,z),(z,y),(z,x)\}. \tag{7}$$

## 2.3 Rule Triplets without Conjunctions

In this section, we consider the simplest triplets. We categorize rule triplets that do not contain conjunctions of literals. In such a case, possible candidates for an exception rule are restricted to the following four:

$$(\alpha, \beta) \in \{(y,z),(z,y),(z,x),(x,z)\}. \tag{8}$$

Since there are three possible candidates for a reference rule from (7), there are twelve candidates for rule triplets without conjunctions. Note that first, however, three candidates that satisfy $(\alpha, \beta) = (\gamma, \delta)$ show a contradictory relation, i.e. $(\alpha \not\to \beta, \alpha \to \beta)$, and must be removed. Second, consider the three candidates that satisfy $(\alpha, \beta) = (\delta, \gamma)$. In such a case, the exception rule $\alpha \not\to \beta$ and its reference rule $\beta \to \alpha$ jointly mean proper inclusion of $\alpha$ in $\beta$, which is not sufficient for an exception from a common sense rule $y \to x$. We therefore remove these three candidates. Third, as shown in figure 1, $t(y,x,x,z,y,z)$ is equivalent to $t(y,x,z,x,y,z)$ if we exchange $x$ and $z$. Similarly, $t(y,x,z,y,z,x)$ is equivalent to $t(y,x,y,z,z,x)$ if we exchange $z$ and $y$. We conclude that there are $12-3-3-2=4$ kinds of rule triplets without conjunctions. We define type 1, 2, 3, and 4 as shown in figure 1. They are defined by the following instantiations of $\alpha$, $\beta$, $\gamma$, and $\delta$:

$$(\alpha, \beta, \gamma, \delta) \in \{(z,x,y,z),(z,x,z,y),(x,z,z,y),(y,z,z,x)\}. \tag{9}$$

Considering the remaining four types of triplets in figure 1, we can argue that all are interesting as each of them represents a kind of violation. For example, type 1 represents a surprise $(z \not\to x)$ for an expected overlap relation between $z$ and $x$ which can be naturally derived from two rules ($y \to x$ and $y \to z$). Further, we can interpret type 3 and 4 as showing mild exceptions. Type 2, however, shows a violation of transitivity, and can thus be regarded as demonstrating the strongest deviations in these rule triplets.

## 2.4 Rule Triplets with a Conjunction

In this section we categorize rule triplets of which premise of an exception rule is a conjunction of two literals. In such a case, possible candidates for an exception

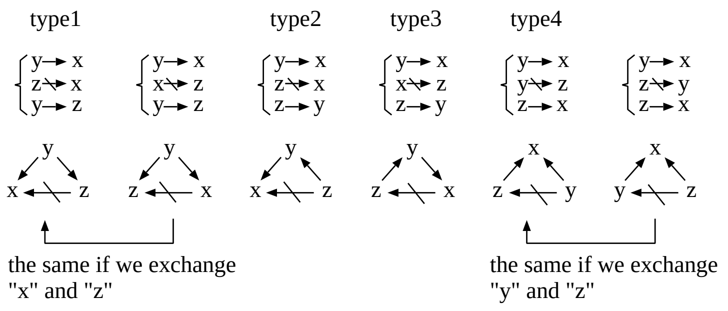
type1
type2
type3
type4
the same if we exchange "x" and "z"
the same if we exchange "y" and "z"

**Fig. 1.** Rule triplets without conjunctions.

rule are restricted to the following three:

$$(\alpha, \beta) \in \{(xy, z), (xz, y), (yz, x)\}. \tag{10}$$

Since there are three possible candidates for a reference rule from (7), there are nine candidates for rule triplets with a conjunction. Note that, as shown in figure 2, $t(y, x, yz, x, y, z)$ is equivalent to $t(y, x, xy, z, y, z)$ if we exchange $x$ and $z$. In addition, $t(y, x, xz, y, z, x)$ is equivalent to $t(y, x, xy, z, z, x)$ if we exchange $y$ and $z$. We here conclude that there are $9 - 2 = 7$ kinds of rule triplets with a conjunction. We define type 5, 6, $\cdots$, and 11 as shown in figure 2:

$$(\alpha, \beta, \gamma, \delta) \in \{(xy, z, y, z), (xz, y, y, z), (xy, z, z, y), (xz, y, z, y), (yz, x, z, y), (xy, z, z, x), (yz, x, z, x)\}. \tag{11}$$

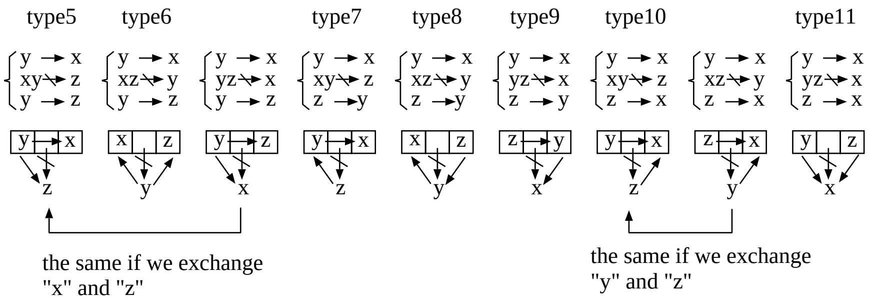
type5
type6
type7
type8
type9
type10
type11
the same if we exchange "x" and "z"
the same if we exchange "y" and "z"

**Fig. 2.** Rule triplets with a conjunction. A rectangle on the top center for each triplet represents a conjunction of literals in the top right and left.

By examining Figure 2, we can interpret type 6, 7, and 10 as interesting triplets, but we can hardly see them as exceptions. Type 5, 8, 9, and 11, however, are different. In each of the four cases the exception rule logically contradicts at least one of the other rules. The situations of type 5, 8, 9, and 11 can only occur if those records that make exceptions to each of the rules in the triplet are distributed in a very specific way, so that the thresholds set by $\theta_{\mathrm{F}}$ and $\theta_{\mathrm{I}}$ can be met. It is an interesting empirical questions whether the triplets derived from data are going to match those expectations.

## 3 Unified Algorithm for All Exception Rules

In this section we propose an algorithm for discovering all the rule triplets which we categorized in the previous section.

### 3.1 Depth-First Search for Rule Triplets

Association rule discovery [1] assumes a transaction data set, which has only binary attributes. Each attribute can take either "y" or "n" as its value, and most of the attribute values are "n". The sparseness of the data set allows to employ breadth-first search. Note that if the breadth-first search was employed for an ordinary data set, the number of large item-sets would be huge. In such a case, space efficiency would be so poor that any breadth-first algorithm would be impractical.

On the other hand, ITRULE [9] assumes an ordinary data set, which can have an attribute with more than two values, and have no assumption on the distribution of attribute values. Since sparseness of the data set is not assumed, it employs a depth-first search.

In this paper, we assume an ordinary data set and propose an algorithm which performs depth-first search for triplets of literals $a, b, c$. We leave an algorithm for a transaction data set for future work. Let $d$ be a candidate for a rule triplet and ! be a logical negation. A threshold vector $\boldsymbol{\theta}$ represents $(\theta_S, \theta_F, \theta_I)$, and $|a|$ is the number of atoms in a literal $a$. Without loss of generality, we assume that every attribute is allocated a unique positive integer $\mathrm{i}(a)$. The main routine of the algorithm is given below, followed by the supporting procedures. Some of them are presented in the next sections.

```
**Algorithm**: rule_triplet_discovery($\boldsymbol{\theta}$, $M$, $D$, $R$).
**Input**: threshold vector $\boldsymbol{\theta}$, maximum length of literals $M$, data set $D$.
**Output**: a set of discovered rule-triplets $R$.
begin
$R, a, b, c := \phi$. //initialization
foreach $a, b, c \in$ the atom set of $D$ such that $\mathrm{i}(a) < \mathrm{i}(b) < \mathrm{i}(c)$ //search
  begin
  $d := (a, b, c)$. //generation of an initial rule triplet
  evaluateRt($d$, $\boldsymbol{\theta}$, $D$, $R$). //evaluation
  if(! Prune($d$, $\boldsymbol{\theta}$, $D$, $R$)) //pruning
    extend($d$, $true$, $true$, $\boldsymbol{\theta}$, $M$, $D$, $R$) //search for depth $\geq 2$
  end
end
```

The routine of search for depth $\geq 2$ is given as follows.

```
**Procedure**: extend($d$, $f_a$, $f_b$, $\boldsymbol{\theta}$, $M$, $D$, $R$).
**Input**: literal triplet $d$, flag $f_a$, flag $f_b$, $\boldsymbol{\theta}$, $M$, $D$.
**Output**: $R$.
begin
if ($|a| > M$) or ($|b| > M$) or ($|c| > M$) //limit of search depth
  return
if ($f_a$) //can extend literal $a_i$
  foreach $a_i \in$ the atom set of $D$ such that $a_i$ does not appear in $d$
    begin
    $d' = (aa_i, b, c)$ //add an atom to $a_i$
    extendSub($d'$, $true$, $true$, $\boldsymbol{\theta}$, $M$, $D$, $R$)
    end
if ($f_b$) //can extend literal $b_i$
  foreach $b_i \in$ the atom set of $D$ such that $b_i$ does not appear in $d$
    begin
    $d' = (a, bb_i, c)$ //add an atom to $b_i$
    extendSub($d'$, $false$, $true$, $\boldsymbol{\theta}$, $M$, $D$, $R$)
    end
foreach $c_i \in$ the atom set of $D$ such that $c_i$ does not appear in $d$
  begin
  $d' = (a, b, cc_i)$ //add an atom to $c_i$
    extendSub($d'$, $false$, $false$, $\boldsymbol{\theta}$, $M$, $D$, $R$)
  end
end

**Procedure**: extendSub($d$, $f_a$, $f_b$, $\boldsymbol{\theta}$, $M$, $D$, $R$).
**Input**: $d$, $f_a$, $f_b$, $\boldsymbol{\theta}$, $M$, $D$.
**Output**: $R$.
begin
evaluateRt($d$). //evaluation
if(! Prune($d$)) //pruning
  extend($d$, $f_a$, $f_b$) //search for depth+1
end
```

Recall that the number of atoms in a literal $a$ is $|a|$. We consider $|a|, |b|, |c| \leq M$ as the search restriction in the above algorithm. For illustration, figure 3 shows the traversal order in the search tree when $M = 2$. Time efficiency of this algorithm is $O(m^{3M})$, where $m$ is the number of attribute in $D$. This is justified since this algorithm is complete in the sense that it discovers all rule triplets. This inefficiency is remedied by the pruning procedure, which will be described in section 3.4.

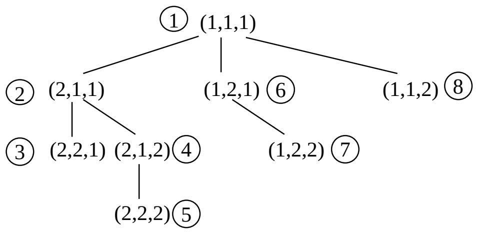

**Fig. 3.** Traversal order in a search tree when $M = 2$, where each circled figure represents the order, and a node represents $(|a|, |b|, |c|)$.

## 3.2 Selection of an Atom for a Continuous Attribute

In this section, we explain how to select an atom given an attribute $x$ [13]. If $x$ is nominal, the algorithm considers all single-value assignments to $x$ as atoms. On the other hand, if $x$ is continuous, the algorithm considers single-range assignments to $x$ as atoms, and these ranges are obtained by discretizing $x$.

Discretization of a continuous attribute can be classified as either global (done before hypothesis construction) or local (done during hypothesis construction) [3]. It can be also classified as either supervised (considers class information) or unsupervised (not using class information) [3]. In this paper, we employ a global unsupervised method due to its time efficiency.

The equal frequency method, when the number of intervals is $k$, divides $n$ examples so that each bin contains $n/k$ (possibly duplicated) adjacent values. The equal frequency method belongs to unsupervised methods and is widely used [3]. We obtain the minimum value and the maximum value of a range by the global equal-frequency method.

In order to exploit the stopping conditions presented in the previous section, ranges are selected as follows. Let the $k$ intervals be $\pi_1, \pi_2, \cdots, \pi_k$ in ascending order. Note that there are two possible ranges, $\pi_1$ to $\pi_{k-1}$ and $\pi_2$ to $\pi_k$, as a range which consists of $k-1$ adjacent intervals. Our algorithm first selects these two as ranges, then selects the ranges of $k-2$ adjacent intervals, i.e. $\pi_1$ to $\pi_{k-2}$, $\pi_2$ to $\pi_{k-1}$, and $\pi_3$ to $\pi_k$. Pruning conditions (13) which will be presented in section 3.4 are employed in ignoring unnecessary ranges. This procedure is iterated by decrementing the number of adjacent intervals in a range until no ranges are left for consideration.

## 3.3 Evaluation of a Literal Triplet

The algorithm in section 3.1 searches for triplets of literals, and rule triplets made of those literal triplets. Note that there are six one-to-one correspondence from a set $\{A, B, C\}$ to a set $\{x, y, z\}$, and there are eleven categories of rule triplets from section 2.3 and 2.4. In the procedure "evaluateRt($d$)", our algorithm considers these $6 * 11 = 66$ possibilities for a literal triplet, and outputs if it satisfies the conditions for a rule triplet.

A rule triplet is evaluated as follows (see section 2.1):

$$t(y,x,\alpha,\beta,\gamma,\delta) \Leftrightarrow (\widehat{\Pr}(y) \geq \theta_{\mathrm{S}},\ \widehat{\Pr}(x|y) \geq \theta_{\mathrm{F}},\ \widehat{\Pr}(\alpha) \geq \theta_{\mathrm{S}},\ \widehat{\Pr}(\beta|\alpha) \leq \theta_{\mathrm{I}}, \\ \widehat{\Pr}(\gamma) \geq \theta_{\mathrm{S}},\ \widehat{\Pr}(\delta|\gamma) \geq \theta_{\mathrm{F}}) \quad (12)$$

Some of these conditions occur more than once in testing different types of rule triplets. Our algorithm applies such conditions only once in order to avoid redundant calculations.

## 3.4 Pruning

Assume the current literal triplet represents a rule triplet $t(y',x',\alpha',\beta',\gamma',\delta')$. It is straightforward to prove that if at least one of (13) holds, no rule triplets $t(y,x,\alpha,\beta,\gamma,\delta)$ in the children nodes satisfy all conditions in the right-hand side of (12). Note that here $y$ expands $y'$.

$$\widehat{\Pr}(y') < \theta_{\mathrm{S}},\ \widehat{\Pr}(x'y') < \theta_{\mathrm{S}}\theta_{\mathrm{F}},\ \widehat{\Pr}(\alpha') < \theta_{\mathrm{S}},\ \widehat{\Pr}(\gamma') < \theta_{\mathrm{S}},\ \widehat{\Pr}(\gamma'\delta') < \theta_{\mathrm{S}}\theta_{\mathrm{F}} \quad (13)$$

As described in section 2.3 and 2.4, there are eleven types of rule triplets. Our algorithm checks the above conditions for all these types. Similarly as in the previous section, our algorithm checks the above conditions for six one-to-one mappings from a set $\{A,B,C\}$ to a set $\{x,y,z\}$ in the procedure "Prune($d$)".

## 4 Experimental Evaluation

We here analyze empirically the statistics of the searched nodes and the discovered rule-triplets. We have chosen UCI data sets [2] since they have served for a long time as benchmark data sets in the machine learning community. In our experiments, we deleted attributes that have only one value in a data set. Table 1 shows characteristics of the data sets employed in the experiments.

In applying our algorithm, the number $k$ of discretization bins was set to 4. Other parameters were set to $\theta_{\mathrm{S}} = 0.025$, $\theta_{\mathrm{F}} = 0.7$, $\theta_{\mathrm{I}} = 0.6$, and $M = 2$. Table 2 shows the results of experiments. Data sets are sorted on the "rule triplets" column, i.e. with respect to the number of discovered rule triplets.

From this table we see a rough correlation between the number of searched nodes and the number of discovered rule triplets. By inspecting Table 1 we see that the number of discovered rule triplets typically increases as the number of attributes, continuous attributes, and values of nominal attributes increase. Data sets "vote", "mushroom", "credit", and "shuttle" are exceptions, and we consider that this is due to the distribution of attribute values.

Table 2 shows that pruning is effective, since without pruning the nodes increase by 5 % ("nursery" and "diabetes") to 285 % ("mushroom"). This is due to the fact that a considerable number of nodes in a search tree tend to have small probabilities for their literals and are thus pruned.

Numbers of discovered rule triplets per types reveal interesting tendencies. From Table 2, we see that type 3, 4, 6, 7, 10 are extremely numerous: they are

**Table 1.** Characteristics of the data sets employed in the experiments, where " ex.", " att.", "c.", and " val." represent the number of examples, the number of attributes, the number of continuous attributes and the number of possible values for the nominal attributes respectively.

| data set | ex. | att. (c.) | val. | data set | ex. | att. (c.) | val. |
|---|---|---|---|---|---|---|---|
| car | 1,728 | 7 ( 0) | 3 - 4 | australian | 690 | 15 ( 6) | 2 - 14 |
| nursery | 12,960 | 9 ( 0) | 2 - 5 | credit | 690 | 16 ( 6) | 2 - 15 |
| postoperative | 90 | 9 ( 1) | 2 - 3 | vote | 435 | 17 ( 0) | 2 - 3 |
| yeast | 1,484 | 9 ( 6) | 2 - 10 | hepatitis | 155 | 20 ( 6) | 2 - 3 |
| diabetes | 768 | 9 ( 8) | 2 | german | 1,000 | 21 ( 6) | 2 - 10 |
| abalone | 4,177 | 9 ( 8) | 3 | mushroom | 8,124 | 22 ( 0) | 2 - 12 |
| breastcancer | 699 | 10 ( 9) | 2 | thyroid | 7,200 | 22 ( 6) | 2 - 3 |
| shuttle | 58,000 | 10 ( 9) | 7 | | | | |

more than $1 * 10^5$ in 11 data sets. Type 1 is also numerous since it is more than $1 * 10^5$ in 3 data sets. On the other hand, type 2 and 9 are modest in number: they never exceed $1 * 10^5$ in any data sets, and exceed $1 * 10^4$ in 9 data sets. Finally, type 11, 8, and 5 are rare in this order: type 11 exceeds $1 * 10^4$ in 1 data set, and type 8 and 5 never exceed $1 * 10^4$ in any data sets. Similar tendencies were observed for $M = 1$. Interestingly, we anticipated the exceptionality of type 2, 5, 8, 9, and 11 as stronger than the other types in section 2.3 and 2.4. We are currently investigating this tendency analytically.

## 5 Conclusion

In this paper, we formalized discovery of interesting exception rules as rule-triplet discovery, and we categorized rule triplets with three literals into eleven types. We also analyzed these eleven types according to their interestingness. Moreover, we proposed an efficient algorithm for simultaneous discovery of all these types based on literal-triplet search and sound pruning.

Our algorithm has been applied to fifteen data sets, and confirmed the analysis on the interestingness of the eleven types. Experimental results clearly show the effectiveness of pruning in reducing the number of searched nodes. The on-going work focuses on relating these rule-triplet types with domain-dependent interestingness in collaboration with experts in various domains. A general interestingness measure for rule triplets represents a promising avenue for future research.

## References

1. R. Agrawal, H. Mannila, R. Srikant *et al.*: Fast Discovery of Association Rules, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, Calif., pp. 307–328 (1996)

**Table 2.** Number of searched nodes and discovered rule triplets for each data set. Here, "unp." represents the number of nodes without pruning divided by the number of nodes with pruning.

| data set | nodes | rule | type1 | type2 | type3 | type4 | type5 | type6 |
|---|---|---|---|---|---|---|---|---|
| | (unp.) | triplets | type7 | type8 | type9 | type10 | type11 | |
| car | 1.06E5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | (1.11) | | 0 | 0 | 0 | 0 | 0 | |
| nursery | 1.12E5 | 71 | 31 | 0 | 0 | 0 | 0 | 40 |
| | (1.05) | | 0 | 0 | 0 | 0 | 0 | |
| postoperative | 4.75E4 | 2.63E4 | 238 | 131 | 1.44E3 | 1.06E4 | 2 | 1.57E3 |
| | (1.42) | | 1.44E3 | 2 | 122 | 1.05E4 | 126 | |
| vote | 6.41E5 | 1.90E5 | 5.94E3 | 2.05E3 | 3.10E4 | 5.11E4 | 0 | 2.96E4 |
| | (1.69) | | 2.53E4 | 2 | 845 | 4.39E4 | 10 | |
| breastcancer | 1.35E6 | 1.65E6 | 3.50E4 | 2.19E4 | 1.98E5 | 5.14E5 | 45 | 1.93E5 |
| | (1.14) | | 1.81E5 | 1 | 1.64E4 | 4.93E5 | 120 | |
| mushroom | 8.64E6 | 2.92E6 | 9.68E4 | 7.50E3 | 1.22E5 | 1.18E6 | 28 | 2.18E5 |
| | (3.85) | | 1.15E5 | 226 | 6.11E3 | 1.16E6 | 4.92E3 | |
| credit | 6.53E6 | 4.25E6 | 4.87E4 | 1.33E4 | 2.27E5 | 1.75E6 | 45 | 2.51E5 |
| | (2.09) | | 2.21E5 | 67 | 1.18E4 | 1.71E6 | 2.43E3 | |
| abalone | 3.25E6 | 4.41E6 | 2.62E5 | 8.53E3 | 4.87E5 | 1.32E6 | 29 | 6.49E5 |
| | (1.08) | | 3.86E5 | 11 | 5.76E3 | 1.29E6 | 1.72E3 | |
| diabetes | 3.97E6 | 4.78E6 | 3.42E4 | 2.24E4 | 3.43E5 | 1.86E6 | 91 | 3.23E5 |
| | (1.05) | | 3.38E5 | 54 | 2.20E4 | 1.83E6 | 4.39E3 | |
| yeast | 3.44E6 | 5.09E6 | 2.66E4 | 1.42E4 | 4.00E5 | 1.93E6 | 0 | 3.88E5 |
| | (1.41) | | 3.99E5 | 9 | 1.31E4 | 1.91E6 | 1.25E3 | |
| australian | 6.81E6 | 5.54E6 | 5.74E4 | 1.58E4 | 3.28E5 | 2.24E6 | 61 | 3.57E5 |
| | (1.45) | | 3.22E5 | 96 | 1.44E4 | 2.19E6 | 3.28E3 | |
| shuttle | 5.47E6 | 7.39E6 | 1.36E5 | 6.64E4 | 6.27E5 | 2.63E6 | 233 | 6.88E5 |
| | (1.15) | | 6.05E5 | 972 | 5.64E4 | 2.57E6 | 7.91E3 | |
| hepatitis | 1.78E7 | 1.65E7 | 1.97E5 | 8.59E4 | 1.07E6 | 6.46E6 | 1.50E3 | 1.21E6 |
| | (1.51) | | 1.06E6 | 2.66E3 | 7.28E4 | 6.33E6 | 7.22E4 | |
| german | 2.05E7 | 1.76E7 | 7.03E4 | 3.39E4 | 8.28E5 | 7.55E6 | 35 | 8.52E5 |
| | (1.32) | | 8.24E5 | 102 | 3.08E4 | 7.40E6 | 8.48E3 | |
| thyroid | 7.00E6 | 1.88E7 | 6.12E4 | 5.78E4 | 1.98E6 | 6.43E6 | 30 | 1.92E6 |
| | (1.56) | | 1.96E6 | 150 | 5.45E4 | 6.34E6 | 1.96E3 | |

2. C.L. Blake and C.J. Merz: "UCI Repository of Machine Learning Databases", *http://www.ics.uci.edu/~mlearn/MLRepository.html*, Dept. of Information and Computer Sci., Univ. of California Irvine (1998).
3. J. Dougherty, R. Kohavi, and M. Sahami: Supervised and Unsupervised Discretization of Continuous Features, in *Proc. Twelfth Int'l Conf. Machine Learning (ICML)*, pp. 194–202 (1995).
4. E.M. Knorr and R.T. Ng: Algorithms for Mining Distance-Based Outliers in Large Datasets, in *Proc. 24th Ann. Int'l Conf. Very Large Data Bases (VLDB)*, pp. 392–403 (1998).
5. T.M. Mitchell: "Machine Learning and Data Mining", *CACM*, Vol. 42, No. 11, pp. 31–36 (1999).
6. B. Padmanabhan and A. Tuzhilin: "A Belief-Driven Method for Discovering Unexpected Patterns", *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, AAAI Press, Menlo Park, Calif., pp. 94–100 (1998).
7. S. Sarawagi: Explaining Differences in Multidimensional Aggregates, in *Proc. 25th Int'l Conf. Very Large Data Bases (VLDB)*, pp. 42–53 (1999).
8. A. Silberschatz and A. Tuzhilin: "What Makes Patterns Interesting in Knowledge Discovery Systems", *IEEE Trans. Knowledge and Data Eng.*, Vol. 8, No. 6, pp. 970–974 (1996).
9. P. Smyth and R.M. Goodman: "An Information Theoretic Approach to Rule Induction from Databases", *IEEE Trans. Knowledge and Data Eng.*, Vol. 4, No. 4, pp. 301–316 (1992).
10. E. Suzuki and M. Shimura : Exceptional Knowledge Discovery in Databases Based on Information Theory, *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, AAAI Press, Menlo Park, Calif., pp. 275–278 (1996).
11. E. Suzuki: "Autonomous Discovery of Reliable Exception Rules", *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD)* , AAAI Press, Menlo Park, Calif., pp. 259–262 (1997).
12. E. Suzuki and Y. Kodratoff: "Discovery of Surprising Exception Rules based on Intensity of Implication", *Principles of Data Mining and Knowledge Discovery, LNAI 1510 (PKDD)*, Springer, Berlin, pp. 10–18 (1998).
13. E. Suzuki: "Scheduled Discovery of Exception Rules", *Discovery Science, LNAI 1721 (DS)*, Springer, Berlin, pp. 184–195 (1999).
14. E. Suzuki and S. Tsumoto: "Evaluating Hypothesis-Driven Exception-Rule Discovery with Medical Data Sets", *Knowledge Discovery and Data Mining, LNAI 1805 (PAKDD)*, Springer, Berlin, pp. 86–97 (2000).