# Finding Errors in New Object Intents

Artem Revenko[1,2]* and Sergei O. Kuznetsov[2]

[1] Technische Universität Dresden
Zellescher Weg 12-14, 01069 Dresden, Germany
[2] National Research University Higher School of Economics
Pokrovskiy bd. 11, 109028 Moscow, Russia
artem_viktorovich.revenko@mailbox.tu-dresden.de,
skuznetsov@hse.ru

**Abstract.** The classification of possible errors in object intents is given and some possibilities of exploring them are discussed. An approach for finding some types of errors in new object intents is introduced. This approach is based on finding implications from an implication basis of the context that are not respected by a new object. It is noted that this approach may lead to a computationally intractable solution. An alternative approach based on computing closure of subsets of object intent is considered. The alternative approach allows one to find a polynomial time algorithm and to deal with inconsistent combination of attributes. This algorithm may also be used to prove object's correctness (existence) or to add missing attributes and remove unnecessary attributes from the object's intent. Experimental results are discussed.

**Keywords:** formal context, implication, error exploration

## 1 Introduction

The work is motivated by the idea of building a multi-user system based on methods of Formal Concept Analysis. Consider two well known multi-user data representation systems: QED and Wikipedia. The aim of the QED project ([1]) is to build a single, distributed, computerized repository that rigorously represents all important, established mathematical knowledge. For this purpose all the input data for QED project should be mathematically formalized. However, it is not always possible in practice. In Wikipedia one may input any data, but no reasoning is supported. To find inferences or consequences from data given in Wikipedia one should read it and process it by own means. However, Formal Concept Analysis offers methods for reasoning with not completely formalized data. Error finding tools are absolutely necessary for such multi-user systems. In Wikipedia the task is solved by moderations, in QED project all the data is checked by automatic provers. To our knowledge there is no FCA-based research on error detection.

* We thank Bernhard Ganter and Sergei Obiedkov for discussion and useful remarks.


ISBN 978–84–695–5252–0, Universidad de Málaga (Dept. Matemática Aplicada)

In this paper we discuss possible error types in new object intents, choose two most important ones, and propose efficient methods for finding them. We provide two different approaches for revealing such errors: one based on computing implication system of the context and another one based on computing the closures of the subsets of the new object intent. Since computing closures may be performed much faster we improve and generalize this approach and finally obtain a procedure for finding all possible errors of considered types. This procedure may also be used to prove correctness of a given object. However, it is not possible to completely escape the necessity of checking some objects by hand. However, the procedure allows one to prove the correctness of input data checking by hand only maximal objects. We finish the paper by discussing experimental results.

In this paper we assume that we are given a context (possibly empty) with correct data and a number of new object intents that may contain errors. This data is taken from some data domain and we may ask an expert whose answers are always correct. However, we should ask as few questions as possible.
All sets and contexts we consider in this paper are assumed to be finite.

## 2 Main Definitions

Let $G$ and $M$ be sets. Let $I \subseteq G \times M$ be a binary relation between $G$ and $M$. Triple $\mathbb{K} := (G, M, I)$ is called a *(formal) context*.
Set $G$ is called a set of *objects*. Set $M$ is called a set of *attributes*.

Consider mappings $\varphi\colon 2^G \to 2^M$ and $\psi\colon 2^M \to 2^G$: $\varphi(X) := \{m \in M \mid gIm \text{ for all } g \in X\}$, $\psi(A) := \{g \in G \mid gIm \text{ for all } m \in A\}$. For any $X_1, X_2 \subseteq G$, $A_1, A_2 \subseteq M$ one has

1. $X_1 \subseteq X_2 \Rightarrow \varphi(X_2) \subseteq \varphi(X_1)$
2. $A_1 \subseteq A_2 \Rightarrow \psi(A_2) \subseteq \psi(A_1)$
3. $X_1 \subseteq \psi\varphi(X_1)$ and $A_1 \subseteq \varphi\psi(A_1)$

Mappings $\varphi$ and $\psi$ define a *Galois connection* between $(2^G, \subseteq)$ and $(2^M, \subseteq)$, i.e. $\varphi(X) \subseteq A \Leftrightarrow \psi(A) \subseteq X$. Usually, instead of $\varphi$ and $\psi$ a single notation $(\cdot)'$ is used. $(\cdot)'$ is sometimes called a *derivation operator*. For $X \subseteq G$ the set $X'$ is called the *intent* of $X$ and is denoted $\mathrm{int}(X)$. Similarly, for $A \subseteq M$ the set $A'$ is called the *extent* of $A$ and is denoted $\mathrm{ext}(A)$.
Let $Z \in M \cup G$. $(Z)''$ is called the *closure* of $Z$ in $\mathbb{K}$. Applying Properties 1 and 2 consequently one gets the *monotonicity* property: for any $Z_1, Z_2 \in G \cup M$ one has $Z_1 \subseteq Z_2 \Rightarrow Z_1'' \subseteq Z_2''$.
Let $m \in M, X \subseteq G$, then $\overline{m}$ is called a *negated* attribute iff $\overline{m} \in X'$ whenever $m \notin X'$.

An *implication* of $\mathbb{K} := (G, M, I)$ is defined as a pair $(A, B)$, written $A \to B$, where $A, B \subseteq M$. $A$ is called the *premise*, $B$ is called the *conclusion* of implication $A \to B$. Implication $A \to B$ is *respected by a set of attributes* $N$ if $A \nsubseteq N$ or $B \subseteq N$. Implication $A \to B$ holds (is valid) in $\mathbb{K}$ if it is respected by

all int($g$), $g \in G$, i.e. every object, that has all the attributes from $A$, also has all the attributes from $B$. Implications satisfy *Armstrong rules*:

$$\frac{}{A \to A} \quad , \quad \frac{A \to B}{A \cup C \to B} \quad , \quad \frac{A \to B, B \cup C \to D}{A \cup C \to D}$$

A *support* of an implication in context $\mathbb{K}$ is the set of all objects of $\mathbb{K}$, whose intents contain the premise and the conclusion of the implication. A *unit implications* is defined as an implication with only one attribute in conclusion, i.e. $A \to b$, where $A \subseteq M,\ b \in M$. Every implication $A \to B$ can be regarded as the set of unit implications $\{A \to b \mid b \in B\}$. One can always observe only unit implications without loss of generality.

An *implication basis* of a context $\mathbb{K}$ is defined as a set $\mathfrak{L}$ of implications of $\mathbb{K}$, from which any valid implication for $\mathbb{K}$ can be deduced by the Armstrong rules and none of the proper subsets of $\mathfrak{L}$ has this property.
A minimal implication basis is an implication basis minimal in the number of implications. A minimal implication basis was defined in [6] and is known as the *canonical implication basis*. In paper [5] the premises of implications from canonical base were characterized in terms of pseudo-intents. A subset of attributes $P \subseteq M$ is called a *pseudo-intent*, if $P \neq P''$ and for every pseudo-intent $Q$ such that $Q \subset P$, one has $Q'' \subset P$. The canonical implication basis looks as follows: $\{P \to (P'' \setminus P) \mid P$ - pseudo-intent$\}$.

We say that an object $g$ is *reducible* in a context $\mathbb{K} := (G, M, I)$ iff $\exists X \subseteq G : g' = \bigcap_{j \in X} j'$.

## 3 Classification of Errors

In this section we use the idea of *data domain dependency*. Usually objects and attributes of a context represent entities (e.g. physical objects, mathematical instances, goods and services, etc.). Dependencies may hold on attributes of such entities (e.g. if an object represents a convex quadrangle, the sum of all angles should be equal to $\pi$). However, such dependencies may not be implications of a context as a result of an error in object intents. Thereby, data domain dependencies are such rules that hold on data represented by objects in a context, but may erroneously be not valid implications of a context. We aim to restore valid dependencies and therefore correct errors.

Every object in a context is described by its intent. In the data domain there may exist dependencies between attributes. In this work we consider only dependencies that do not have negations of attributes in premises. As mentioned above there is no need to specially observe non-unit implications. Consider possible types of such dependencies ($A \subseteq M,\ b, c \in M$):

1. $A \to b$
2. $A \to \overline{b}$
3. $A \to b \vee c$

4. $A \to \Phi$, where $\Phi$ is any logical formula not considered above, for example, $\Phi = a \vee (b \wedge \overline{c})$

If we have no errors in a context, all the dependencies of Type 1 are deducible from implication basis. However, if we have not yet added enough objects in the context, we may get false consequence. Nevertheless, it is guaranteed that none of valid dependencies is lost, and, as we add objects without errors we reduce the number of false consequences from the implication basis.
The situation is different if we add an erroneous object. It may violate a dependency valid in the data domain. In this case, until we find and correct the error, we are not able to deduce all dependencies valid in the data domain from the implication basis, no matter how many correct objects we add afterwards.
Now we take a closer look at various possible cases. If a dependency of Type 3 holds in the data domain, there should be a restriction on reducible objects in the context. However, reducible objects do not change neither the closure system, nor the implication basis of a context. This case would be an interesting direction for further investigation, but now we do not consider this case.
In Type 4 formula $\Phi$ can be represented in conjunctive normal form. That is why we may hope that if it is possible to find and generalize solution for Type 3, we would be able to reveal dependencies of Type 4 as well.
Types 1 and 2 are most simple and common dependencies. In this work we try to find the algorithm to reveal these two types of dependencies and find corresponding errors.

## 4 Finding Errors

We introduce two different approaches to finding errors. The first one is based on inspecting the canonical basis of a context. When adding a new object to the context one may find all implications from the canonical basis of the context such that the implications are not respected by the intent of the new object. These implications are then output as questions to an expert in form of unit implications. If at least one of these implications is accepted, the object intent is erroneous. Since the canonical basis is the most compact (in the number of implications) representation of all valid implications of a context, it is guaranteed that minimal number of questions is asked and no valid dependencies of Type 1 are left out.
Although this approach allows one to reveal all dependencies of Type 1, there are several issues. The problem of producing the canonical basis with known algorithms is intractable. Recent theoretical results suggest that the canonical base can hardly be computed with better better worst-case complexity than that of the existing approaches ([3], [2]). One can use other bases (for example, see progress in computing proper premises [9]), but the algorithms known so far are still too costly and non-minimal bases do not guarantee that the expert is asked minimal sufficient number of questions.

However, since we are only interested in implications corresponding to an object, it may be not necessary to compute a whole implication basis. Here is

the second approach. Let $A \subseteq M$ be the intent of the new object not yet added to the context. $m \in A''$ iff $\forall g \in G : A \subseteq g' \Rightarrow m \in g'$, in other words, $A''$ contains the attributes common to all object intents containing $A$. The set of unit implications $\{A \to b \mid b \in A'' \setminus A\}$ can then be shown to the expert. If all implications are rejected, no attributes are forgotten in the new object intent. Otherwise, the object is erroneous. This approach allows one to find errors of Type 1.

## 5 An Example

Consider the following example with convex quadrangles. The formal context in Fig. 1 contains convex quadrangles and their properties. The context is not full, i.e. not all possible convex quadrangles are considered, and some objects in the context are reducible (they do not bring new information in the implication basis of the context). Seven attributes are considered. Attributes "has equal legs" and "has equal angles" require at least two angles/legs of a quadrangle to be equal. Some dependencies on attributes are obvious, e.g., it is clear that if all angles are equal in a quadrangle, then this quadrangle definitely has equal angles.
Four errors are presented in Fig 2. Errors are added to the context in Fig. 1 one at a time. One should treat an error as an object to be added to the context.
The context without errors in Fig. 1 is denoted $\mathbb{K}$ , $(\cdot)'$ is the corresponding derivation operator.
The context of errors in Fig. 2 is denoted by $\mathbb{K}_\mathrm{e}$ , $(\cdot)^\mathrm{e}$ is the derivation operator for $\mathbb{K}_\mathrm{e}$.

*Example 1.* $\{\text{Erorr 2}\}^\mathrm{e}$ =$\{$has equal legs, has right angle, all legs equal, all angles equal$\}$
$\{\text{Erorr 2}\}^{\mathrm{e}''}$ =$\{$has equal legs, has right angle, all legs equal, all angles equal, has equal angles$\}$

The canonical basis of the context $\mathbb{K}$ in Fig. 1 looks as follows:

1. at least 3 different angles → at least 3 different legs
2. all angles equal → has equal angles, has equal legs, has right angle
3. all legs equal → has equal angles, has equal legs
4. has right angle, at least 3 different legs → at least 3 different angles
5. has equal angles, has equal legs, at least 3 different legs, all legs equal → has right angle, at least 3 different angles, all angles equal
6. has equal angles, has equal legs, at least 3 different legs, all angles equal, has right angle, at least 3 different angles → all legs equal
7. has right angle, has equal legs, all legs equal, has equal angles → all angles equal

Consider Error2. $\{\text{Erorr 2}\}^\mathrm{e}$ =$\{$has equal legs, has right angle, all legs equal, all angles equal$\}$
Using the first approach we find that this object does not respect Implications 2 and 3. It is easy to see that both implications are valid in data domain. Thereby,

| Convex quadrangles | has equal legs | has equal angles | has right angle | all legs equal | all angles equal | at least 3 different angles | at least 3 different legs |
|---|---|---|---|---|---|---|---|
| Square | × | × | × | × | × | | |
| Rectangle | × | × | × | | × | | |
| Quadrangle | | | | | | × | × |
| Rhombus | × | × | | × | | | |
| Parallelogram | × | × | | | | | |
| Rectangular trapezium | | × | × | | | × | × |
| Quadrangle with 2 equal legs and right angle | × | | × | | | × | × |
| Isosceles trapezium | × | × | | | | × | |
| Rectangular trapezium with 2 equal legs | × | × | × | | | × | × |
| Quadrangle with 2 equal angles | | × | | | | × | × |
| Quadrangle with 2 equal legs | × | | | | | × | × |
| Quadrangle with 2 equal legs and 2 equal angles | × | × | | | | × | × |

**Fig. 1.** Context of convex quadrangles $\mathbb{K}$

| Errors | has equal legs | has equal angles | has right angle | all legs equal | all angles equal | at least 3 different angles | at least 3 different legs |
|---|---|---|---|---|---|---|---|
| Error1 | × | | | × | | × | |
| Error2 | × | | × | × | × | | |
| Error3 | | × | × | × | × | × | × |
| Error4 | × | × | | × | | | × |

**Fig. 2.** Context of errors $\mathbb{K}_e$

the expert recognizes object as an error.
As $\{\text{Erorr 2}\}^{e''}$ =$\{$has equal legs, has right angle, all legs equal, all angles equal, has equal angles$\}$, the second approach yields the following implication: $\{$has equal legs, has right angle, all legs equal, all angles equal$\} \to \{$has equal angles$\}$. This implication is also valid in data domain. Although this implication is less general than Implications 2 and 3, it is sufficient to indicate an error.

## 6 Improvements

Obviously, applying the derivation operator two times is much easier task than computing the canonical basis, and can be performed in polynomial time. However, the following case is possible. Let $A \subseteq M$ be the intent of the new object such that $\nexists g \in G : A \subseteq g'$. In this case $A'' = M$ and the implication $A \to A'' \setminus A$ has empty support. This may indicate an error of Type 2, because the object intent contains combination of attributes impossible in the data domain, but the object may be correct as well. An expert could be asked if the combination of attributes in the object intent is consistent in the data domain. For such a question the information already input in the context is not used. More than that, this question is not sufficient to reveal an error of Type 1.

**Proposition 1.** *Let $\mathbb{K} = (G, M, I), A \subseteq M$. The set*

$$\mathcal{I}_A = \{B \to d \mid B \in \mathcal{MC}_A, d \in B'' \setminus A \cup \overline{A \setminus B}\},$$

*where $\mathcal{MC}_A = \{B \in \mathcal{C}_A | \nexists C \in \mathcal{C}_A : B \subset C\}$ and $\mathcal{C}_A = \{A \cap g' \mid g \in G\}$, is the set of all unit implications (or their non-trivial consequences with some attributes added in the premise) of Types 1 and 2 such that implications are valid in $\mathbb{K}$, not respected by $A$, and have not empty support.*

*Proof.* Let $(E \to f) \in \mathcal{I}_A$. As $E = A \cap g'$ for some $g \in G$, $f \in g'$. Consider possible cases:

1. $f \in E'' \setminus A$. As follows from the definition of derivation operator, the implication is valid and $f \in g'$, i.e. at least $g$ is in support of this implication. More than that, $E'' \setminus A \nsubseteq A$ and implication is not respected by $A$.
2. $f \in \overline{A \setminus E}$. Since $E$ is a maximal intersection ($\nexists C \in \mathcal{C}_A : E \subseteq C$), there does not exist object $\hat{g} \in G$ such that $E \cup m \in \hat{g}'$, for any $m \in \overline{A \setminus E}$. This proves that implication is valid and at least $g$ is in support of this implication. More than that, since $A \setminus E \subseteq A$ the implication is not respected by $A$.

Now let $E \to f$ be a valid implication not respected by $A$ with a non-empty support. Then $E \in A, f \notin A$ and there exists $g \in G$ such that $E, f \in g'$. By construction there exists $B \in \mathcal{MC}_A$ such that $E \subseteq A \cap g' \subseteq B$. Consider possible cases:

1. $f \in M$. As implication is valid and not respected by $A$, we have $f \in (A \cap g')'' \setminus A$. From monotonicity property it follows that $f \in B''$. It shows that $(B \to f) \in \mathcal{I}_A$.

2. $f \in \overline{M}$. Let $f = \overline{v}$. As implication is valid, there does not exist $\hat{g} \in G$ such that $v \in \hat{g}'$. Then $v \in A \setminus B$ and $(B \to f) \in \mathcal{I}_A$. □

Proposition 1 allows one to find an algorithm for computing the set of questions to an expert revealing possible errors of Types 1 and 2. The pseudocode is pretty straightforward.

### 6.1 Pseudocode

```
inspect(𝕂:=(G, M, I), A⊆M)
1. if A″=A:
    2. return ∅
3. Candidates = {object′∩A | object∈G}
4. Candidates = {C∈Candidates |
                 ∄B∈Candidates: C⊆B}
5. Result = ∅
6. for Candidate in Candidates:
   7. Result.add({Candidate → d |
                  d∈(Candidate″\A ∪ A \ Candidate)})
8. return Result
```

A is the intent of the new object. In the third line we compute the set of all subsets that can produce the desired implication. In the fourth line we discard all the non-maximal elements. In lines 6 and 7 we compute closures and add the corresponding implications.

If we try to add a new object intent such that it is not contained in any object intent already in the context, we should ask a new question to the previous new object intent. Indeed, now there may exist new valid implications with nonempty support. However, it is easy to see that such a question is exactly the question to the new object intent with negated conclusion. Indeed, let $B \to c$ be the implication representing the question to the new object. If it is rejected and the new object is added to the context, then the new object respects the implication $B \to \overline{c}$. As the implication $B \to c$ was asked, there were no object intents in the context respecting implication $B \to \overline{c}$. That is why the implication $B \to \overline{c}$ was never asked before and should be asked now. Finding such implications does not require any time and guarantees independence of the order of adding new object intents.

It is worth noting that considering only implications with non-empty support is not always safe. On the one hand, it allows one to avoid questions not based on any input information. On the other hand, this consideration does not allow one to state that there are no errors in an object. However, it suffices to check only maximal object intents in the context at any point in time, because only they may contain combinations of attributes not occurring elsewhere in the context. So, in order to avoid doubling the work, one should not consider implications with empty support, checking maximal object intents "by hand" whenever it is needed to show that the context is free from errors of Types 1 and 2.

# 7 Results

For the sake of compactness in this section we present implications in non-unit form. The name `inspect_dg` is used to denote the function implementing the first described approach (involving the canonical basis).

## 7.1 Example

Inspecting Error1:

`inspect_dg`
at least 3 different angles → at least 3 different legs
all legs equal → has equal angles, has equal legs

`inspect`
has equal legs, at least 3 different angles → at least 3 different legs,
all legs equal
has equal legs, all legs equal → has equal angles, at least 3 different angles

Both algorithms reveal possible errors in a similar manner, although there are obvious differences. In the output of `inspect_dg` the premises are smaller than in the output of `inspect`. The latter also reveals dependencies of Type 2. It is easy to see that all output implications hold in data domain. For example, if all legs are equal in a quadrangle, it should have equal angles and should not have 3 different angles. As a corollary this object should be recognized as an error.

Inspecting Error2:

`inspect_dg`
all angles equal → has equal angles, has equal legs, has right angle
all legs equal → has equal angles, has equal legs

`inspect`
has right angle, has equal legs, all legs equal, all angles equal → has equal angles

In this example we are able to ask even less number of questions to an expert using `inspect` as with `inspect_dg`. This is the result of finding implications generated by maximal subsets of object's intent. Again, all implications are valid in data domain. The intent of Error2 occurs in the context (in the intent of Square), that is why we do not get any negated attributes in the output of `inspect`.

Inspecting Error3:

`inspect_dg`
all angles equal → has equal angles, has equal legs, has right angle
all legs equal → has equal angles, has equal legs
`inspect`
has equal angles, has right angle, at least 3 different legs, at least 3 different angles → all angles equal, all legs equal
has equal angles, has right angle, all legs equal, all angles equal → has equal legs, at least 3 different angles, at least 3 different legs

In the case of Error3 we get both implications from the output of `inspect_dg` combined in one implication with a bigger premise in the output of `inspect`. In addition we obtain several implications with negated attributes. It is easy to see that all implications hold in the data domain.

Inspecting Error4:

`inspect_dg`
has equal angles, has equal legs, at least 3 different legs, all legs equal → has right angle, at least 3 different angles, all angles equal
`inspect`
has equal angles, has equal legs, all legs equal → at least 3 different legs
has equal angles, has equal legs, at least 3 different legs → all legs equal

Error4 is a very special case where the corresponding implication from canonical basis has empty support. In the output of `inspect_dg` we obtain all questions possible for this intent. As discussed above these questions are not based on any information input so far. Even if we add attributes "at least 3 different angles" and "all angles equal" and reject the last implication we would not be able to recognize this object as an error. On the contrary `inspect` allows us to recognize errors of Type 2.

### 7.2 Experiment

Below the results of tests on a bigger data are presented. The tests were conducted as follows: all objects one by one are first separated from a context and then added as a new object; all the possible errors of Type 1 and 2 are found and output for this object.

FCA package for Python was used for implementation ([8]). For computing the canonical basis an optimized algorithm based on `Next Closure` was used ([7]). All tests described below were run on computer with Intel Core i7 1.6GHz processor and 4 Gb of RAM running Linux Ubuntu 11.10 x64.

In Fig. 3 the results of running both algorithms on random contexts are presented. For each context the number of objects is equal to 50. Parameter $d$ represents the density of the context, i.e. the probability of having cross in the cross-table representing the relation. This result is presented in the semi-logarithmic scale as the growth of complexity of computing the canonical basis

is nearly exponential in this example. It is easy to note that with the growth of the number of attributes and the density the difference between runtime of two algorithms grows as well.

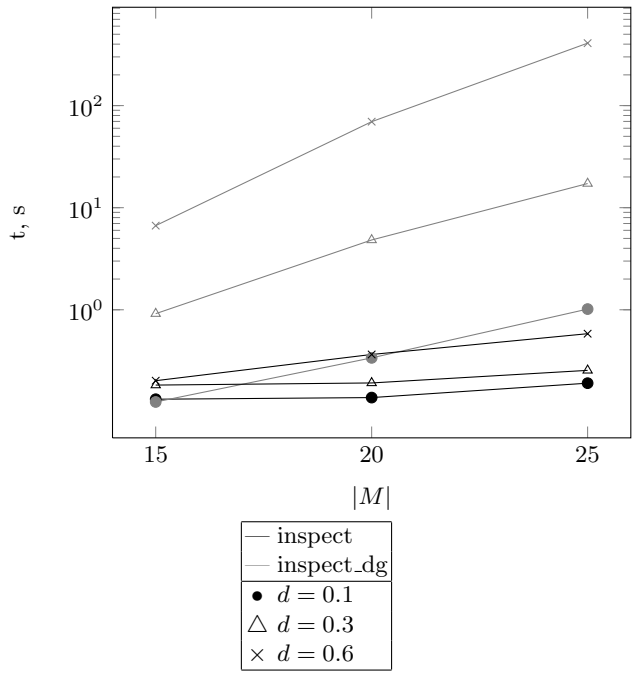


**Fig. 3.** Comparison of runtime on random contexts in semilog scale.

In Table 1 the results of running both algorithms on real data are presented. The data is taken from the UCI repository ([4]). In this tests algorithm `inspect` outperforms algorithm `inspect_dg` as well. Again, with the growth of the number of attributes the difference becomes more noticeable.

| Context Name | $\|G\|$ | $\|M\|$ | inspect (s) | inspect_dg (s) |
|---|---|---|---|---|
| wine | 178 | 68 | 4.674 | 13627.952 |
| house-votes-84 | 435 | 18 | 1.048 | 64.735 |
| SPECT | 266 | 23 | 0.636 | 672.942 |

**Table 1.** Comparison of runtime on real data from UCI

## 8 Conclusion

An algorithm for finding errors of two types in new object intents is presented. As opposed to finding the canonical basis of the context the proposed algorithm terminates in polynomial time. Moreover, after checking only maximal object intents "by hand" it is possible to find all errors of two considered types (or prove their absence).

## References

1. The qed project. http://mizar.org/qed/.
2. M. Babin and S. O. Kuznetsov. Recognizing pseudo-intent is conp-complete. *Proc. 7th International Conference on Concept Lattices and Their Applications, University of Sevilla*, pages 294–301, 2010.
3. Felix Distel and Barış Sertkaya. On the complexity of enumerating pseudo-intents. *Discrete Applied Mathematics*, 159(6):450–466, 2011.
4. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
5. B. Ganter. Two basic algorithms in concept analysis. *Preprint-Nr. 831*, 1984.
6. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Math. Sci. Hum*, 24(95):5–18, 1986.
7. S. Obiedkov and V. Duquenne. Attribute-incremental construction of the canonical implication basis. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):77–99, April 2007.
8. Nikita Romashkin. Python package for formal concept analysis. https://github.com/jupp/fca.
9. Uwe Ryssel, Felix Distel, and Daniel Borchmann. Fast computation of proper premises. In Amedeo Napoli and Vilem Vychodil, editors, *International Conference on Concept Lattices and Their Applications*, pages 101–113. INRIA Nancy – Grand Est and LORIA, 2011.