Towards Handling Potential Issues in Business Rule Bases

Carl Corea
Institute for Information Systems Research, University of Koblenz, Germany
ccorea@uni-koblenz.de

Matthias Thimm
Artificial Intelligence Group, University of Hagen, Germany
matthias.thimm@fernuni-hagen.de

Abstract

(Business) Rule Bases are knowledge representation formalisms with a distinction between rules and facts. In this work, we address the issue of analyzing *potential issues* in rule bases, i. e., sets of rules which cause an inconsistency only *if* they are activated together. The problem of dealing with potential issues is very common in industrial domains of knowledge representation and reasoning, in particular in business rules management. Here, it is often not clear during modelling which rules will be activated together. We introduce a formalization of the above-mentioned problem and present means for the prioritization of such issues. Also, we investigate central aspects related to computational complexity and present an ad-hoc (online) approach for computing the set of the most severe potential issues. We implement and evaluate our approach with real-life datasets.

1 Introduction

(Business) rule bases are commonly used knowledge representation formalisms in industrial domains such as Business Process Management [21, 31, 17]. In rule bases, knowledge is represented with a distinction between rules and facts, where sets of (provided) facts are evaluated against the set of business rules to make inferences.

As an illustrative example, consider the following rule base $\mathcal{B}_1$ from the financial sector (we will formalize syntax and semantics later) with the intuitive meaning

We thank the referees for their help and comments in improving this manuscript. This work was supported by the Deutsche Forschungsgemeinschaft (grant DE 1983/9-2).

that we have two rules stating that 1) platinum customers are credit worthy, and 2) customers that are on a ban-list are not credit worthy:

$$\mathcal{B}_1 = \{creditWorthy \leftarrow platinumCustomer; \\ \neg creditWorthy \leftarrow banList\}$$

This rule base can be used to reason about loan applications. For this, sets of facts (stemming from concrete customer loan applications, e. g., information about a corresponding customer) can be provided for making inferences. For example, consider a provided fact set $\mathcal{F}_1 = \{platinumCustomer, \neg banList\}$. Then, using $\mathcal{B}_1' = \mathcal{B}_1 \cup \mathcal{F}_1$, this allows to entail that the corresponding customer is in fact creditworthy.

Consider however a second example: If a loan application is filed by a customer who has a platinum status but was also found in a ban list, this yields an inconsistency (due to contradictory conclusions $creditWorthy$, $\neg creditWorthy$). This inconsistency arises due to an unforseen interaction of input facts, in this case: $platinumCustomer$ and $banList$. In this work, we refer to such a problematic combination of rules as a *potential issue* (i. e., rules that could yield an inconsistency if activated together).

Potential issues can easily arise during rule management and revision, as it is hard, if not impossible, for human modellers to consider all combinations of rules and check for any unwanted interactions between them. Thus, results are needed that allow to identify all potential issues (**RQ1**).

In this context, it may however not be plausible to "simply" present the modeller with a list of all potential issues, as there could be exponentially many (and it could be highly unfeasible for the human to process this amount of information). Therefore, further results are needed that allow to prioritize potential issues and present these to the human ranked by their severity (**RQ2**). This is also necessary in light of the complexity of computing all potential issues: As we will show, computing *all* potential issues is intractable.

In this work, we address the research gaps raised via the above research questions RQ1 and RQ2. In particular, our contributions are as follows:

1. We define the notion of potential issues (Section 3).
2. We investigate several aspects of computational complexity pertaining to the analysis of potential issues (Section 4).
3. Towards application in practice, we present an initial ad-hoc approach for computing a set of most severe potential issues (Section 5). Importantly, this approach allows to compute the set of most severe potential issues without

the need to precompute all potential issues. We implement and evaluate our approach with real-life datasets.

Our discussion is based on preliminaries shown in Section 2 and is concluded in Section 6.

The notion of potential issues introduced in this work is related to, yet, conceptually different from the previously introduced notion of quasi-inconsistency [8]. Quasi-inconsistency describes situations where contradictory rules will always be activated together (and cannot be activated independently from another). On the contrary, for potential issues as discussed in this work, the individual rules can be activated individually and can be used for reasoning in a meaningful way in such cases. Thus, an entirely new question arises of having to analyze how likely certain combinations of rules are activated together, which we address in this paper.

2 Preliminaries

2.1 (Business) Rule Bases

To describe rule bases, we consider a general monotonic rule-based knowledge representation formalism [20]. For this, let $\mathcal{A}$ be a set of propositional atoms and $\mathcal{L}$ the corresponding set of literals, i. e., $\mathcal{L} = \{a, \neg a \mid a \in \mathcal{A}\}$ where $\neg$ is interpreted as classical negation. Then, a rule r has the form

$$r : \quad l_0 \leftarrow l_1, \ldots, l_m. \tag{1}$$

with $l_0, \ldots, l_m \in \mathcal{L}$ and $m \geq 1$ (we require rules to have a non-empty premise). Note that the "$\leftarrow$" is not equivalent to an implication in propositional logic but is interpreted as a production rule (definite clauses). For example, as stated in [4] rules of the form $\neg a \leftarrow a$ are therefore not plausible in this setting. We denote $head(r) = l_0$ and $body(r) = \{l_1, \ldots, l_m\}$, and $\mathcal{R}_\mathcal{L}$ as the set of all rules. A *rule base* $\mathcal{B}$ is then a set of rules, i. e., $\mathcal{B} \subseteq \mathcal{R}_\mathcal{L}$. As shown in the introduction, fact sets can be evaluated against the rule bases for reasoning. For this, let $F \subseteq \mathcal{L}$ be a set of facts.

A set $M \subseteq \mathcal{L}$ of literals is *closed* wrt. a rule base $\mathcal{B}$ and a fact set F, iff $F \subseteq M$ and for every rule of the form in (1), if $l_1, \ldots, l_m \in M$ then $l_0 \in M$. Here, the *F-minimal model* M of a rule base $\mathcal{B}$, is the smallest closed set of literals (wrt. set inclusion). We say a set of literals M is *consistent* if it does not contain both a and $\neg a$ for an atom a. Furthermore, we say a rule base $\mathcal{B}$ is *F-consistent* if its F-minimal model is consistent. If not, we say that $\mathcal{B}$ is *F-inconsistent*. A rule base $\mathcal{B}$ is called *minimally F-inconsistent* if

1. F is consistent
2. $\mathcal{B}$ is F-inconsistent, and
3. for every $F' \subsetneq F$, $\mathcal{B}$ is F'-*consistent.*

Example 1. *Consider the rule base $\mathcal{B}_2$ and fact set F_1, defined via*

$$\mathcal{B}_2 = \{\neg c \leftarrow a; \quad c \leftarrow d; \quad d \leftarrow b\}$$
$$F_1 = \{a, b\}$$

Then we have F_1 is a consistent set of facts, $\mathcal{B}_2$ is $\{a, b\}$-inconsistent, and there is no proper subset S of F_1 s.t. $\mathcal{B}_2$ is S-inconsistent, thus, $\mathcal{B}_2$ is minimally $\{a, b\}$-inconsistent.

We refer the reader to [8] for some other properties of F-consistency. Note that in this work, we consider the rule base to be acyclic, i.e., there are no cycles within the rules (such as $a \leftarrow b; b \leftarrow a$).[1] Some general observations for cyclic rule bases are also provided in [8].

The introduced formalism and notion of F-inconsistency is mainly geared towards *definitional rules*, i.e., rules that define structure [21]. In this sense, given a rule *"a platinum customer is creditworthy"*, a potential issue would exist when a second rule *"a customer found on a ban-list is not creditworthy"* is introduced (cf. Section 1). Note however that the contents discussed in this work can also be extended to *behavioral rules*, e.g., interpreting rules as "follows"-relations over a sequence of time.

2.2 Related Work and Contributions

We recall the rule base $\mathcal{B}_1$. As discussed in Section 1, $\mathcal{B}_1$ contains what we call a *potential issue*, which is a combination of rules that *might* become inconsistent, depending on the instance-dependent fact input. The phenomenon of potential issues is inherently related to knowledge- and rule engineering and touches multiple related fields such as business rules management [21, 7], belief revision [2, 23], inconsistency diagnosis/handling [27, 11], and conceptual modelling [5]. In particular, the core problem of potential issues is the risk of invoking (classic-logical) *inconsistency.*

The central distinction to related works on inconsistency is satisfiability: For knowledge that is inconsistent, there can exist no model. So in a sense, it is clear that

[1] Formally, for acyclic rule bases, consider the *dependency graph* $G_\mathcal{B}$ of a rule base $\mathcal{B}$ as the directed graph $G_\mathcal{B} = (\mathcal{B}, E_\mathcal{B})$ where $(r_1, r_2) \in E_\mathcal{B}$ for $r_1, r_2 \in \mathcal{B}$ iff $head(r_1) \in body(r_2)$. Then, a rule base is acyclic if the graph $G_\mathcal{B}$ is acyclic.

this problem needs to be resolved for using the knowledge for (classical) reasoning as intended – see e.g. [27], who shows how minimal correction sets can be used to resolve inconsistency. On the contrary, if knowledge is classically *con*sistent but contains a potential issue, there is an almost arbitrary number of possible models. So a novel problem of identifying such potential issues (in otherwise consistent knowledge) arises here, which we address in this work. A related work in this direction is [4], who studies the notion of *rule inconsistency* (vs. logical inconsistency). It seems from this that for rule-systems, there are some problems "beyond" classic-logical inconsistency, which is exactly what we address in this work, with potential issues. Other related notions are that of incoherence [19, 13, 14] and quasi-inconsistency [8].

Regarding inconsistency analysis in general, quantitative approaches have been extensively studied in the context of *inconsistency measurement* [29]. In this regard, a central problem which arises in our setting is that there can be vastly too many potential issues to present to the human modeller, and thus approaches are needed that prioritize issues, e.g., for determining resolution strategies. In general, the problem of "recommending" an order in which elements/issues should be attended to in the scope of re-modelling can be described as a bit underdeveloped in inconsistency measurement research (see [22] for an early work in this direction). Here, we propose an approach for computing the *top-k* potential issues—intuitively, a set of "most severe" potential issues—which is needed in our setting where it may be unfeasible to attend to all issues individually. W.r.t. the classification on *dimensions* on inconsistency from [32], the approach for computing the top-k potential issues presented in this work focuses on the aspect of *actions*, in particular *learning and re-modelling*.

Having the ability to rank potential issues by some form of severity also allows for advanced means for inconsistency handling [24, 16, 22]. This related to the question of what happens to the potential issues, once they are detected. The underlying idea of our approach is very in line with ideas of paraconsistent reasoning techniques, in such that we enable the user to determine a threshold w.r.t. "living with" certain risks of potential issues, in favor of not having to process all potential issues, which might be cognitively or computationally unfeasible. On the other hand, if the severity or risk of certain issues is considered too high, suitable means for inconsistency handling can be applied, e.g., various change operations such as deleting or rewriting rules, modelling exceptions, or re-modelling the rule base in general.

3 Potential Issues in Business Rule Bases

We recall $\mathcal{B}_1$ from the introduction. As we have seen, this rule base contains a modelling flaw, as there exists a fact combination F, s.t. $\mathcal{B}_1$ is F-inconsistent. Intuitively, we therefore say that $\mathcal{B}_1$ contains a potential issue. Especially in industrial application scenarios, identifying potential issues is of high interest during rule modelling, as it is unclear at this point which combination of instance-dependent facts will be observed "later" (during run-time). So experts may need to be aware of such risks, such that they can assess these potential issues, e. g., as a basis for re-modelling and improving the rule base.

For this purpose, we will define the notion of potential issues in the following. For that, we need some further notation.

Definition 1 (Rule activation)**.** *A set of facts X* activates *a finite set of rules R iff there is a sequence $< r_1, \ldots, r_n >$ with $\{r_1, \ldots, r_n\} = R$ s.t.*

1. $body(r_1) \subseteq X$
2. *for all $i = 2, \ldots, n$ we have $body(r_i) \subseteq \{head(r_1), \ldots, head(r_{i-1})\} \cup X$*

A set of facts X minimally (w.r.t. set-inclusion) activates *a set of rules R iff X activates R and there is no proper subset of X that activates R. If X (minimally) activates R we also say that X is a* (minimal) activation set *of R.*

Intuitively, X is a set of facts for deriving all conclusions of rules in R.

Example 2. *We recall the rule base $\mathcal{B}_2$*

$$\mathcal{B}_2 = \{\neg c \leftarrow a; \quad c \leftarrow d; \quad d \leftarrow b\}.$$

For each individual rule, its activation set consists simply of the body of the rule, i. e., $\{a\}$ is an activation set of $\{\neg c \leftarrow a\}$. Furthermore, the set $\{b\}$ also activates both rules $\{ \quad c \leftarrow d; \quad d \leftarrow b\}$.

We are now ready to define *potential issues*. The intuition we want to capture is that we want to present the modeller with the two sets of rules that can potentially "clash", i. e., can become inconsistent during run-time, given a certain combination of facts is observed. So in a potential issue, we denote two sets of rules, which relate to the "opposing" rules of the rule base that can contradict each other. Furthermore, we denote for each of these two opposing rule sets the concrete facts that would be needed to activate the rule sets. This duality allows the modeller to

quickly understand the opposing sets of facts which, in combination, would yield an inconsistency.[2] In result, a potential issue is a quadruple as follows.

Definition 2 (Potential Issue)**.** *Let $R_1, R_2 \subseteq \mathcal{R}_\mathcal{L}$ be sets of rules and X_1, X_2 be consistent sets of literals. A tuple (R_1, X_1, R_2, X_2) is called a* potential issue *iff*

1. $X_1 \setminus X_2 \neq \emptyset$ *and* $X_2 \setminus X_1 \neq \emptyset$
2. X_1 *minimally activates* R_1*.*
3. X_2 *minimally activates* R_2*.*
4. R_1 *is* X_1*-consistent and* R_2 *is* X_2*-consistent.*
5. $R_1 \cup R_2$ *is* $\{X_1 \cup X_2\}$*-inconsistent.*

A tuple (R_1, X_1, R_2, X_2) is called a minimal potential issue *iff there are no R_1', R_2', X_1', X_2' with $R_1' \subseteq R_1$ and $R_2' \subseteq R_2$ (one of these set inclusions being proper) such that (R_1', X_1', R_2', X_2') is also a potential issue. Let $\mathsf{PotIssues}(\mathcal{B})$ be the set of all potential issues in $\mathcal{B}$, and $\mathsf{MinPotIssues}(\mathcal{B})$ be the set of all minimal potential issues in $\mathcal{B}$.*

In other words, a potential issue (R_1, X_1, R_2, X_2) describes a case where the occurrence of a certain fact combination $X_1 \cup X_2$ relative to a rule base $\mathcal{B}$ would activate a combination of rules in $\mathcal{B}$ which entail an inconsistent conclusion. But importantly, these rule sets are activated by different fact sets (which are, at least partially, non-overlapping). So this means there *could* well be situations where the individual rules in R_1, or R_2, respectively, can be activated on their own and used properly for reasoning, however, if X_1 and X_2 are evaluated against the rule base simultaneously, *then* an inconsistency will always arise. Hence the term: *potential* issue.

Example 3. *We recall the business rule base $\mathcal{B}_1$ from the introduction*

$$\begin{aligned}\mathcal{B}_1 = \{&creditworthy \leftarrow platinumCustomer;\\ &\neg creditworthy \leftarrow banList\}.\end{aligned}$$

[2]Note that denoting the opposing facts in two separate subsets is an important design-choice to allow the modeller to consider a tradeoff if it possible/likely for these facts to occur simultaneously. We will talk more about probabilities in Section 5.

Then we have that

$$\begin{aligned} t_1 =&(\{creditworthy \leftarrow platinumCustomer\},\\ &\{platinumCustomer\},\\ &\{\neg creditworthy \leftarrow banList\},\\ &\{banList\}) \end{aligned}$$

is a potential issue of $\mathcal{B}_1$. The tuple t_1 can be read by the experts s.t. X_1 and X_2 in combination (here: a customer who is both a platinum customer and is on a ban-list) will trigger the corresponding, inconsistent rules. The expert can then decide if this fact combination is e.g. likely to occur an whether the rules should be re-modelled accordingly to account for this case.

The introduced notion of potential issues is related to, yet, conceptually different from the previously introduced notion of quasi-inconsistency [8]. In short, quasi-inconsistency refers to cases where there are rules in a rule base that will *always* be activated together, but yield a contradictory conclusions, should they be activated. For this, given $R_1, R_2 \subseteq \mathcal{R}_\mathcal{L}$ and two consistent sets of literals X_1, X_2 as before, a "quasi-inconsistent tuple" (also referred to as an actual issues in [8]), is defined as a tuple (R_1, X_1, R_2, X_2) s.t. conditions 2-5 from Definition 2 hold and $X_1 \subseteq X_2$. In words, we have a situation where the rules in R_1 and R_2 cannot be activated individually wrt. X_2. This is an important distinction. With potential issues, we have the situation that a certain fact combination *could* activate contradictory rules, yet, it is in fact possible that the corresponding rules can be activated individually, allowing to draw meaningful conclusions from them. This makes the debugging of potential issues more challenging, as it requires to differentiate how likely certain rules might be activated together. In summary, potential issues and actual issues describe different types of problematic rule combinations with inherently different resolution strategies.

Example 4. *Consider the following rule bases $\mathcal{B}_3$–$\mathcal{B}_6$, defined via*

$$\begin{aligned} \mathcal{B}_3 &= \{c \leftarrow a; \quad \neg c \leftarrow b\}\\ \mathcal{B}_4 &= \{c \leftarrow a; \quad \neg c \leftarrow a\}\\ \mathcal{B}_5 &= \{c \leftarrow a; \quad \neg c \leftarrow a, b\}\\ \mathcal{B}_6 &= \{c \leftarrow a, d; \quad \neg c \leftarrow a, b\} \end{aligned}$$

Then for

$$t_2 = (\{c \leftarrow a\}, \{a\}, \{\neg c \leftarrow b\}, \{b\})$$
$$t_3 = (\{c \leftarrow a\}, \{a\}, \{\neg c \leftarrow a\}, \{a\})$$
$$t_4 = (\{c \leftarrow a\}, \{a\}, \{\neg c \leftarrow a, b\}, \{a, b\})$$
$$t_5 = (\{c \leftarrow a, d\}, \{a, d\}, \{\neg c \leftarrow a, b\}, \{a, b\})$$

we have that t_2 is a potential issue of $\mathcal{B}_3$ (but not an actual issue), t_3 is an actual issue of $\mathcal{B}_4$ (but not a potential issue), t_4 is an actual issue of $\mathcal{B}_5$ (but not a potential issue), and t_5 is a potential issue of $\mathcal{B}_6$ (but not an actual issue). As a further example, note that a tuple $t_6 = (\{c \leftarrow a\}, \{a\}, \{\neg c \leftarrow a, b\}, \{b\})$ would be neither a potential or actual issue of $\mathcal{B}_5$, as $\{b\}$ does not minimally activate $\{\neg c \leftarrow a, b\}$.

Corollary 1. *Let $\mathcal{B}$ be a rule base as before and $\mathsf{ActIssues}(\mathcal{B})$ be the set of all (minimal) actual issues in $\mathcal{B}$, then $\mathsf{PotIssues}(\mathcal{B}) \cap \mathsf{ActIssues}(\mathcal{B}) = \emptyset$.*

4 Computational Complexity

We now continue with an investigation of central problems related to analyzing potential issues in rule bases. For that, we assume familiarity with basic concepts of computational complexity and basic complexity classes such as P and NP, see [25] for an introduction.

We start with analysing the complexity of verification tasks pertaining to issues.

Lemma 1. *Let $\mathcal{B}$ be a rule base, $R_1, R_2 \subseteq \mathcal{B}$, and X_1, X_2 consistent sets of literals. Checking whether (R_1, X_1, R_2, X_2) is a potential issue can be done in polynomial time (in the size of $\mathcal{B}$) and (R_1, X_1, R_2, X_2)).*

Proof. We go through the properties of an issue step by step (compare with Definition 2):

1. Checking $X_i \setminus X_j \neq \emptyset$ is polynomial.
2. Checking that X_1 activates R_1 is simple forward propagation (check every rule whether it can be activated with X_1 alone; if yes add the head of that rule to X_1 and continue). This is polynomial in the number of rules in R_1, the number of literals in the bodies of rules in R_1, and the size of X_1. In order to check that X_1 minimally activates R_1 it suffices to check whether $X_1' \subseteq X_1$ where X_1' has exactly one fact less than X_1 does not activate R_1 (for all such X_1', which are exactly $|X_1|$ many)

3. Analogous for X_2/R_2.

4. Checking whether R_1 is X_1-consistent is simple forward propagation and checking whether the set of derived literals is consistent; analogous for R_2 and X_2.

5. Checking whether $R_1 \cup R_2$ is $\{X_1 \cup X_2\}$-inconsistent is analogous. □

This also holds for the verification of minimal potential issues.

Lemma 2. *Let $\mathcal{B}$ be a rule base, $R_1, R_2 \subseteq \mathcal{B}$, and X_1, X_2 consistent sets of literals. Checking whether (R_1, X_1, R_2, X_2) is a minimal potential issue an be done in polynomial time.*

Proof. For checking whether (R_1, X_1, R_2, X_2) is an issue cf. Lemma 1. For minimality, check for each $R_1' \subseteq R_1, R_2' \subseteq R_2$, where exactly one of R_1', R_2' contains one rule less, whether $R_1' \cup R_2'$ is $X_1 \cup X_2$-consistent. Should this be the case, (R_1, X_1, R_2, X_2) cannot be a minimal issue. Note that there are polynomially many tuples ($|R_1| + |R_2|$) to check, each check being polynomial, cf. Lemma 1. □

For any rule base $\mathcal{B}$, we are now interested if there is at least one potential issue (as this allows to distinguish from a sound rule base). We refer to this verification task as Dec-PI.

Dec-PI	**Input**:	Rule base $\mathcal{B}$
	Output:	TRUE iff there is a potential issue (R_1, X_1, R_2, X_2) with $R_1, R_2 \subseteq \mathcal{B}$

The above lemmas transfer to this verification task as follows.

Proposition 1. *Dec-PI is NP-complete.*

Proof. To show NP-membership, consider the following non-deterministic algorithm: On input $\mathcal{B}$, guess sets of rules R_1, R_2 with $R_1, R_2 \subseteq \mathcal{B}$ and consistent sets of literals X_1, X_2. Then, if $t = (R_1, X_1, R_2, X_2)$ is an issue of $\mathcal{B}$, return TRUE, otherwise return FALSE. Cf. Lemma 1 to see that this can be done in polynomial time. In result, we have that Dec-PI $\in$ NP.

To show NP-hardness, we use a reduction of the classical satisfiability problem Sat to Dec-PI. For this, let Φ be an instance of Sat over the signature A (=set of atoms), i. e., $\Phi = C_1 \wedge \ldots \wedge C_n$ where each $C_i = l_{i,1} \vee \ldots \vee l_{i,n(i)}$ with literals $l_{i,1}, \ldots, l_{i,n(i)}$ over A for $i = 1, \ldots, n$ (recall that a literal is either an atom $a \in A$ or its negation $\neg a$). The satisfiability problem Sat is then whether Φ is satisfiable, i. e., whether there exists an interpretation $I : A \to \{\mathsf{T}, \mathsf{F}\}$ s. t. for all $i = 1, \ldots, n$

there is a $k \in \{1, \ldots, n(i)\}$ with $I(l) = \mathsf{T}$ (if $l = l_{i,k}$ is an atom) or $I(l') = \mathsf{F}$ (if $\neg l' = l_{i,k}$ is a negated atom). To show this, on input Φ, we construct a rule base $\mathcal{B}_\Phi$ as follows: For each clause C_i for $i = 1, \ldots, n$, we create new atoms α_i and α'_i (with the informal meaning that α_i/α'_i is derivable in $\mathcal{B}_\Phi$ if C_i is satisfied). For each clause $C_i = l_{i,1} \vee \ldots \vee l_{i,n(i)}$ we then construct $n(i)$ rules of the form

$$\mathcal{B}_i = \{\alpha_i \leftarrow l_{i,1}; \quad \ldots \quad ; \quad \alpha_i \leftarrow l_{i,n(i)}; \quad \alpha'_i \leftarrow l_{i,1}; \quad \ldots \quad ; \quad \alpha'_i \leftarrow l_{i,n(i)}\}$$

Then, we create three new atoms π, β_1, β_2 and define $\mathcal{B}_\Phi$ to be composed of the above rules and two further rules:

$$\mathcal{B}_\Phi = \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_n \cup \{\pi \leftarrow \alpha_1, \ldots, \alpha_n, \beta_1; \quad \neg\pi \leftarrow \alpha'_1, \ldots, \alpha'_n, \beta_2\}$$

Now, we claim that Φ is satisfiable iff $\mathcal{B}_\Phi$ contains a potential issue. We begin by showing that satisfiability of Φ implies that there exists a potential issue in $\mathcal{B}_\Phi$. For this, let I be an interpretation that satisfies Φ. Without loss of generality, for each clause C_i, let $l_{1,i}$ be the literal that is satisfied by I (can be easily achieved by reordering the literals in each clause). Define $X = \{l_{1,1}, \ldots, l_{1,n}\}$ to be the set of all these literals. Furthermore, define

$$R = \{\alpha_1 \leftarrow l_{1,1}; \quad \ldots; \quad \alpha_n \leftarrow l_{1,n}\} \qquad R' = \{\alpha'_1 \leftarrow l_{1,1}; \quad \ldots; \quad \alpha'_n \leftarrow l_{1,n}\}$$
$$R_1 = R \cup \{\pi \leftarrow \alpha_1, \ldots, \alpha_n, \beta_1\} \qquad R_2 = R' \cup \{\neg\pi \leftarrow \alpha'_1, \ldots, \alpha'_n, \beta_2\}$$

By construction, X activates R and R'. Now let $X_1 \subseteq X$ s. t. X_1 minimally activates R, resp., let $X_2 \subseteq X$ s. t. X_2 minimally activates R'. Then, define $X'_1 = X_1 \cup \{\beta_1\}$, and $X'_2 = X_2 \cup \{\beta_2\}$. Note that $X_1 \neq X_2$. It follows that (R, X'_1, R', X'_2) is a potential issue of $\mathcal{B}_\Phi$.

For the other direction, assume that $\mathcal{B}_\Phi$ contains a potential issue and let (R_1, X_1, R_2, X_2) be an issue of $\mathcal{B}_\Phi$. Observe that the only rules able to derive contradictory claims in $\mathcal{B}_\Phi$ are the two rules with heads π and $\neg\pi$, respectively. So one of these rules must be in R_1 and the other in R_2 (if they both would be in one of them this would violate condition 4 of Definition 2). Moreover, for each such rule, one of the literals of the corresponding clause must be present in an activation set. From these literals, an interpretation I can be constructed satisfying all clauses C_i, $i = 1, \ldots, j$ in analogy to the reverse direction before (note that this interpretation is partial, not all propositions need to occur in $X_1 \cup X_2$; however, the truth value of the remaining propositions is irrelevant and can be set arbitrary).

Finally, note that $\mathcal{B}_\Phi$ is of polynomial size wrt. Φ. This gives a polynomial-time reduction from Sat to Dec-QI, showing NP-hardness. □

So the verification of whether a rule base contains a potential issue is indeed intractable. Note that it is immediate to apply this result also for minimal potential issues (for membership, apply the same non-deterministic algorithm as before and apply a check via Lemma 2; for hardness, apply the same reduction as above and ask whether the constructed rules are in a minimal potential issue). Also, using a similar reduction as in [8], it is immediate to see that the problem of counting the number of potential issues is #P-complete.[3]

Proposition 2. *Counting the number of potential issues is #P-complete.*

Proof. For #P-membership, Lemma 1 already showed that checking whether a given tuple (R_1, X_1, R_2, X_2) is a potential issue can be decided in polynomial time. It follows that counting the number of potential issues is in #P.

For showing hardness, we reduce the problem #1-3-SAT to our problems that has been shown to be #P-complete in [10]. Given a formula Φ over $A = \{a_1, \ldots, a_m\}$ in 3-CNF, i. e., a formula of the form $\Phi = (l_{1,1} \vee l_{1,2} \vee l_{1,3}) \wedge \ldots \wedge (l_{n,1} \vee l_{n,2} \vee l_{n,3})$ (with exactly 3 literals per clause), we ask for the number of those interpretations $I : A \to \{\mathsf{T}, \mathsf{F}\}$ s. t. for all $i = 1, \ldots, n$ there is *exactly one* $k \in \{1, \ldots, n(i)\}$ with $I(l) = \mathsf{T}$ (if $l = l_{i,k}$ is an atom) or $I(l') = \mathsf{F}$ (if $\neg l' = l_{i,k}$ is a negated atom). We call an interpretation satisfying this condition *1-3-model* of Φ. On input Φ we construct a rule base $\mathcal{B}_\Phi$ as follows. For each clause C_i, $i = 1, \ldots, n$, we create two new atoms α_i, α_i' (both with the informal meaning that α_i/α_i' is derivable in $\mathcal{B}_\Phi$ if C_i is satisfied). Furthermore, let ρ_1 and ρ_2 be two new atoms. For each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ we construct six rules of the form

$$\begin{aligned}\mathcal{B}_i = \{&\alpha_i \leftarrow l_{i,1}, \overline{l_{i,2}}, \overline{l_{i,3}}, \rho_1;\\ &\alpha_i \leftarrow l_{i,1}, \overline{l_{i,2}}, \overline{l_{i,3}}, \rho_2;\\ &\alpha_i \leftarrow \overline{l_{i,1}}, l_{i,2}, \overline{l_{i,3}}, \rho_1;\\ &\alpha_i \leftarrow \overline{l_{i,1}}, l_{i,2}, \overline{l_{i,3}}, \rho_2;\\ &\alpha_i \leftarrow \overline{l_{i,1}}, \overline{l_{i,2}}, l_{i,3}, \rho_1;\\ &\alpha_i \leftarrow \overline{l_{i,1}}, \overline{l_{i,2}}, l_{i,3}, \rho_2\}\end{aligned}$$

Moreover, we create new atom $\delta_1, \ldots, \delta_m$ and construct the following rules

$$\mathcal{B}_A = \{\delta_i \leftarrow a_i; \quad \delta_i \leftarrow \neg a_i \mid i = 1, \ldots, m\}$$

[3] #P is the complexity class of counting problems where the problem of deciding whether a particular element has to be counted is in P.

Then we create yet another new atoms π and define $\mathcal{B}_\Phi$ to be composed of the above rules and two further rules:

$$\mathcal{B}_\Phi = \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_n \cup \mathcal{B}_A \cup \{\pi \leftarrow \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_m; \\ \neg\pi \leftarrow \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_m\}$$

We now claim that the number of 1-3-models of Φ is exactly the number of potential issues of $\mathcal{B}_\Phi$.

Let I be a 1-3-model if Φ. Define $(X \cup \{\rho_1\}, R_1, X \cup \{\rho_2\}, R_2)$ via

$$\begin{aligned}
X &= \{a \mid a \in \mathcal{A}, I(a) = \mathsf{T}\} \cup \{\neg a \mid a \in \mathcal{A}, I(a) = \mathsf{F}\} \\
R_A &= \{\delta_i \leftarrow a_i \mid a_i \in X\} \cup \{\delta_i \leftarrow \neg a_i \mid \neg a_i \in X\} \\
R_1 &= R_A \cup \{\pi \leftarrow \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_m\} \\
&\quad \cup \{\alpha_i \leftarrow l_{i,1}, \overline{l_{i,2}}, \overline{l_{i,3}}, \rho_1 \mid I(l_{i,1}) = \mathsf{T}, I(l_{i,2}) = \mathsf{F}, I(l_{i,3}) = \mathsf{F}, i = 1, \ldots, n\} \\
&\quad \cup \{\alpha_i \leftarrow \overline{l_{i,1}}, l_{i,2}, \overline{l_{i,3}}, \rho_1 \mid I(l_{i,1}) = \mathsf{F}, I(l_{i,2}) = \mathsf{T}, I(l_{i,3}) = \mathsf{F}, i = 1, \ldots, n\} \\
&\quad \cup \{\alpha_i \leftarrow \overline{l_{i,1}}, \overline{l_{i,2}}, l_{i,3}, \rho_1 \mid I(l_{i,1}) = \mathsf{F}, I(l_{i,2}) = \mathsf{F}, I(l_{i,3}) = \mathsf{T}, i = 1, \ldots, n\} \\
R_2 &= R_A \cup \{\neg\pi \leftarrow \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_m\} \\
&\quad \cup \{\alpha_i \leftarrow l_{i,1}, \overline{l_{i,2}}, \overline{l_{i,3}}, \rho_2 \mid I(l_{i,1}) = \mathsf{T}, I(l_{i,2}) = \mathsf{F}, I(l_{i,3}) = \mathsf{F}, i = 1, \ldots, n\} \\
&\quad \cup \{\alpha_i \leftarrow \overline{l_{i,1}}, l_{i,2}, \overline{l_{i,3}}, \rho_2 \mid I(l_{i,1}) = \mathsf{F}, I(l_{i,2}) = \mathsf{T}, I(l_{i,3}) = \mathsf{F}, i = 1, \ldots, n\} \\
&\quad \cup \{\alpha_i \leftarrow \overline{l_{i,1}}, \overline{l_{i,2}}, l_{i,3}, \rho_2 \mid I(l_{i,1}) = \mathsf{F}, I(l_{i,2}) = \mathsf{F}, I(l_{i,3}) = \mathsf{T}, i = 1, \ldots, n\}
\end{aligned}$$

Observe that for each clause C_i, $i = 1, \ldots, n$ both R_1 and R_2 contain exactly one rule with head α_i and that exact rule is activated by $X \cup \{\rho_1\}$ and $X \cup \{\rho_2\}$, respectively. It follows that $X \cup \{\rho_1\}$ minimally activates R_1 and $X \cup \{\rho_2\}$ minimally activates R_2. Furthermore, $X \cup \{\rho_1\}$ is R_1-consistent, $X \cup \{\rho_2\}$ is R_2-consistent, and $X \cup \{\rho_1, \rho_2\}$ is $R_1 \cup R_2$-inconsistent. It follows that $(X \cup \{\rho_1\}, R_1, X \cup \{\rho_2\}, R_2)$ is a potential issue of B_Φ.

Conversely, let (X_1, R_1, X_2, R_2) be any potential issue. As the only derivable conflict in R_Φ is between π and $\neg\pi$, the corresponding rules must be present in R_1 and R_2, respectively. Assume $\pi \leftarrow \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_m \in R_1$ then there has to be at least one rule from $\{\delta_i \leftarrow a_i; \quad \delta_i \leftarrow \neg a_i\}$ for each $i = 1, \ldots, m$ in R_1. As X_1 must be consistent, not both rules can be activated, so there is exactly one of these rules in R_1. It also follows that X_1 is exactly the union of the premises of that rules and either ρ_1 or ρ_2. Without loss of generality, assume $\rho_1 \in X_1$. In order to have $X_1 \setminus X_2 \neq \emptyset$ and $X_2 \setminus X_1 \neq \emptyset$, we must have $\rho_2 \notin X_1$ and $\rho_2 \in X_2$. For each α_i there must be at least one of the three rules with head α_i (where the body contains ρ_1) in R_1, otherwise α_i could not be derived. As at most one of these rules can be activated

by X_1, there is exactly one of the three rules in R_1 (which is also activated, otherwise (X_1, R_1, X_1, R_2) would not be a potential issue). The same applies to R_2 and the rules where the body contains ρ_2. Define now an interpretation I via $I(a) = \mathsf{T}$ if $a \in X_1$ and $I(a) = \mathsf{F}$ if $\neg a \in X_1$. As X_1 activates each rule with head α_i for $i = 1, \ldots, n$, I satisfies exactly one literal of each clause C_i. It follows that I is a 1-3-model of Φ.

It follows that each 1-3-model of Φ corresponds exactly to one potential issue of $\mathcal{B}_\Phi$. Therefore, their number is exactly the same. As $\mathcal{B}_\Phi$ is of polynomial size wrt. Φ we have shown that counting the number of potential issues is #P-hard. □

From these complexity results, we see that the intractability poses an obstacle for applying the notion of potential issues in practice, e.g., it may be unfeasible to compute all potential issues. To counteract this problem, we propose an approach to compute the top-k potential issues (intuitively, a set of k most severe potential issues), which we present in the following.

5 Application in Practice: Computing the Top-k Potential Issues

As motivated in the introduction, analyzing a rule base for potential issues is an important task during business rule management, as it is hard, if not impossible, for human modellers to consider all rules and their interrelations for any unwanted interactions. Assume a modeller needs to analyze a rule base for any potential issues. In a naive setting, one could compute the set of all minimal potential issues via Definition 2 and present the results to the user. However, with this naive approach, we see two fundamental problems:

1. As shown in Section 4, the computation may be unfeasible.
2. Even if one could compute the set of all potential issues, a list of all such potential issues may likely be too large for humans to process. This may severely impede the ability of the human to actually leverage the presented results for concrete actions, e.g., how to re-model the rule base.

From the two identified problems, it seems apparent that "simply" computing the set of all potential issues may not be a viable solution in practice. Instead, only a feasible number of results, e. g., a prioritized list, should be presented to the user. Therefore, we present an approach to compute the *top-k*, i.e., "most severe", potential issues. Importantly, we compute this limited set in an ad-hoc/online fashion, i. e., without the need to pre-compute the entire set of potential issues in advance

(which, as discussed, may be intractable). We build on the so-called Apriori approach from the field of associative rule learning (in relational databases) [1] for the problem of retrieving highly problematic potential issues in rule bases. The Apriori algorithm can be used to identify "frequent item sets" relative to certain quantitative measures and therefore seems a good candidate for identifying a set of most severe potential issues in our setting. Regarding the notion of a *severity* of a potential issue, we argue this is closely linked to the probability of certain facts, respectively, the occurrence of specific fact combinations: The combined probability of facts ocurring together determines the probability of whether a potential issue will become an actual inconsistency during run-time. From this perspective, if a fact combination that would activate a potential issue is impossible or highly unlikely, the severity of that potential issue also decreases; vice versa, a potential issue "activated" more often can be seen as more severe.

Importanly, this notion of linking the severity of a potential issue to the probability of the corresponding activation set (= fact set) is *one* possible dimension of severity. There might well exist other dimensions, for example, even if a potential issue is unlikely, it might be considered as severe. We will further comment on this in Section 6. However, the viewpoint we take here can cleary be motivated by empirical evidence (we will introduce our evaluation and real-life datasets in Section 5.2). The *Business Process Intelligence Challenge*[4] provides real-life datasets containing actual customer cases. From this data, the actual distribution of fact occurrences over all cases can be computed. Figure 1 shows the distribution of fact occurrences for the BPI 2020 dataset. As can be seen, there are only a few facts that occur often, and many facts that occur seldomly. In this setting, we argue the combined probability of certain fact sets can be used properly to assess the severity of potential issues, e. g., issues associated with an activation set that has a very low probability may be unproblematic. As we will show in Section 5.2 (Evaluation), the distributions as shown in Figure 1 are very common in real-life settings and thus facilitate a prioritization of potential issues as proposed in this work.

An observation that is in order is that we do not assume any form of domain knowledge about dependencies between facts. Hence we assume we can actually observe arbitrary fact sets stemming from customer cases without any further hidden constraints, i.e., probabilistic independence can be assumed for the occurrence of these facts. Note however that if domain knowledge is available about certain relationships, these (known) fact combination can also be provided as input to our approach in order to check for potential issues pertaining to this input combination. As we will show in our empirical evaluation (see below), the facts we observed in

[4] https://bit.ly/3AMBpra

Activity	Frequency	Relative frequency
Declaration SUBMITTED by EMPLOYEE	11,531	20.43 %
Declaration FINAL_APPROVED by SUPERVISOR	10,131	17.95 %
Payment Handled	10,044	17.8 %
Request Payment	10,040	17.79 %
Declaration APPROVED by ADMINISTRATION	8,202	14.53 %
Declaration APPROVED by BUDGET OWNER	2,820	5 %
Declaration REJECTED by EMPLOYEE	1,365	2.42 %
Declaration REJECTED by ADMINISTRATION	952	1.69 %
Declaration APPROVED by PRE_APPROVER	685	1.21 %
Declaration REJECTED by SUPERVISOR	293	0.52 %
Declaration SAVED by EMPLOYEE	135	0.24 %
Declaration REJECTED by MISSING	91	0.16 %
Declaration REJECTED by PRE_APPROVER	86	0.15 %
Declaration REJECTED by BUDGET OWNER	59	0.1 %
Declaration FOR_APPROVAL by SUPERVISOR	1	0 %
Declaration FOR_APPROVAL by PRE_APPROVER	1	0 %
Declaration FOR_APPROVAL by ADMINISTRATION	1	0 %

Figure 1: Distribution of fact occurrences over all cases of the BPI 2020_1 dataset.

real-life datasets were probabilisticly independent and we could not discover such problems, so we argue there will be domains where this assumption is well justified.

We now continue to present and evaluate our approach.

5.1 Proposed Approach

Given a rule base $\mathcal{B}$ and the task to analyze the contained potential issues, we propose to compute only the *top-k* potential issues, where the parameter k represents a minimal threshold of the probability that the facts corresponding to the activation set of the potential issue will actually occur together (i. e., that the potential issue will actually become an inconsistency at run-time). In the following, we assume that a distribution of the probability of occurrence for individual facts is known in advance, either based on historic process data or based on expert assessment.

We begin by pairwise considering all fact sets $\subseteq \mathcal{L}$. The motivation of considering *pairs* of fact sets is that the individual sets of this pair are meant to correspond to the *two* individual activation sets of a potential issue. In particular, we compute those pairs of fact sets $\subseteq \mathcal{L}$ that have a combined probability[5] $> k$, where k is a user-defined parameter. Note that in the worst case, if all facts have a probability of 1, this would scale exponentially, as it would require to compare all subsets of available facts. However, based on the presented empirical evidence (cf. the above discussion), the actual number of fact sets that need to be considered will likely be much lower.

For all such pairs of fact sets that will occur with a probability $> k$, we store this pair along with the corresponding probability. As we lean on the Apriori algorithm principle, we refer to such a triple of fact set, fact set and probability as an *apriori*

[5]Recall that probabilistic indepence is assumed in our setting. Then, the combined probability is the product of the individual probabilities.

pair. Consequently, for all apriori pairs, denoting the two individual fact sets of that pair as X_1, X_2, we verify whether there exists a tuple $t = (R_1, X_1, R_2, X_2)$ with $R_1, R_2 \subseteq \mathcal{B}$ and t being a minimal potential issue of $\mathcal{B}$. If this is the case, the tuple t belongs to the top-k potential issues. The described verification task of checking if there exists a potential issue triggered by a given apriori pair can be easily computed by representing the rule base as a directed graph $G_{\mathcal{B}} = (\mathcal{B}, E_{\mathcal{B}})$ – where for every $r \in \mathcal{B}$ we add an edge ($body(r)$,$head(r)$) to $E_{\mathcal{B}}$ – and then checking if there exists individual paths from X_1 and X_2 to two nodes corresponding to two complementary literals (e. g. $a, \neg a$). Here, every path can be interpreted as a sequence of rules, c.f. also Definition 1 on rule set activation. We refer the reader to [6] for a detailed example.

As a result, we are able to identify the top-k potential issues for a rule base $\mathcal{B}$ w.r.t. a threshold k. The approach proposed in this section is summarized in Algorithm 1.

Algorithm 1: Algorithm for computing the top-k potential issues of a rule base $\mathcal{B}$

Input : Rule base $\mathcal{B}$, List⟨String,Double⟩ *factOccurrenceDistribution* (facts and their probabilities), *threshold* (minimal top-k threshold)
Output: Set of all top-k potential issues in $\mathcal{B}$

```
1 aprioriPairs ← construct all apriori pairs with a minimal probability k //cf. above discussion
2 topKPotentialIssues = {}
3 for a : aprioriPairs do
4     A1 ← a[0]
5     A2 ← a[1]
6     if ∃R1, R2 ⊆ B s.t. (R1, A1, R2, A2) is a min. pot. issue then
7         topKPotentialIssues ← topKPotentialIssues ∪ (R1, A1, R2, A2)
8 return topKPotentialIssues
```

Example 5. *Consider the business rule base $\mathcal{B}_7$, with*

$$\begin{aligned}\mathcal{B}_7 = \{&creditworthy \leftarrow platinumCustomer;\\ &\neg creditworthy \leftarrow hasDebt;\\ &\neg creditworthy \leftarrow blackListed;\\ &platinumCustomer \leftarrow honorCircle\}.\end{aligned}$$

Then we have the following four potential issues in $\mathcal{B}_7$

$$\begin{aligned}
t_1 =&(\{creditworthy \leftarrow platinumCustomer\}, \{platinumCustomer\},\\
&\{\neg creditworthy \leftarrow hasDebt\}, \{hasDebt\})\\
t_2 =&(\{creditworthy \leftarrow platinumCustomer\}, \{platinumCustomer\},\\
&\{\neg creditworthy \leftarrow blackListed\}, \{blackListed\})\\
t_3 =&(\{creditworthy \leftarrow platinumCustomer;\\
&\quad platinumCustomer \leftarrow honorCircle\}, \{honorCircle\},\\
&\{\neg creditworthy \leftarrow blackListed\}, \{blackListed\})\\
t_4 =&(\{creditworthy \leftarrow platinumCustomer;\\
&\quad platinumCustomer \leftarrow honorCircle\}, \{honorCircle\},\\
&\{\neg creditworthy \leftarrow hasDebt\}, \{hasDebt\})
\end{aligned}$$

Assume the following probability of occurrence was observed from historic process data:

$$\begin{aligned}
platinumCustomer &: 0.9 \qquad & hasDebt &: 0.1\\
blackListed &: 0.5 \qquad & honorCircle &: 0.3\\
&... &&
\end{aligned}$$

(e.g., the combined probability of platinumCustomer and hasDebt is $0.9 * 0.1 = 9\%$*). Consider a minimum threshold* k *of 40%. This yields only one apriori pair with a probability* $> k$*:*

$$\langle platinumCustomer, blackListed, 0.45\rangle$$

So via line 6 in Algorithm 1, we can correctly return $\{t_2\}$ *as the set of top-k potential issues with* $k = 40\%$*, effectively filtering out* 75% *of all potential issues that have to be presented to the user.*

Theorem 1. *For a given probability* k*, Algorithm 1 is correct for the problem of computing the set of all potential issues that have an activation set with a combined probability* $> k$ *(i.e. the set of top-k potential issues).*

Proof. Following [9], we consider an algorithm to be correct if it satisfies the properties of *soundness* and *completeness*. We address these in turn.

Regarding soundness, assume Algorithm 1 is not sound. Then the algorithm would either 1) not return all potential issues with a probability $> k$, or 2) return a potential issue with a probability $<= k$. Regarding case 1, this is not the case as

we iterate through all apriori pairs (line 3) and verify if a corresponding potential issue can be found. Note that we assume the algorithm assesses this verification of min. potential issues by checking the conditions in Definition 2, i.e., the detected tuples are min. potential issues by definition. Regarding case 2, this is implictly assumed via the generation of all apriori pairs, i.e., a fact combination is only added to the set of all apriori pairs if the combined probability exceeds k. So Algorithm 5 is sound by contradiction.

Regarding completeness, note that we assume the map of facts (and occurrences) to be finite. In this case, note that any rule base without any potential issues, no issues with a probability $> k$, or no facts has an empty set of top-k potential issues. So there does not exist a case where failure should be reported as via the above assumption. The set of top-k potential issues is initialized in line 2, and relevant issues are added to this set (line 7). The set is then always returned in line 8. So Algorithm 5 is complete by invariance. □

5.2 Evaluation

To evaluate the benefits of our proposed approach for real-life settings, we conducted an initial evaluation with real-life datasets. The goal of this evaluation is to compare the *runtimes* and the *number of potential issues* for a) computing *all* potential issues, and b) computing only the *top-k* potential issues (for a predefined k). In the following, we present our evaluation results, which clearly show that our approach can immensely reduce the number of potential issues that need to be presented to the human expert. We begin by briefly introducing our dataset.

5.2.1 Description of Data-Set

We performed our experiments with the real-life datasets of the Business Process Intelligence Challenge (BPIC). The BPIC is an international scientific challenge series and provides real-life case data from various domains such as healthcare, government or industry. We selected these data sets as they are real-life data sets that stem from domains where correct decision-making is critical (for example, the data set of the BPIC 2016 contains decision rules for the treatment of Sepsis - a life threatening disease), thus, investigating potential issues here is an important task in these domains.

Importantly, various tools exist that allow to *mine* rule sets from this case data. Here, we applied the state-of-the-art tool MINERful [3]. With this tool, it is possible to obtain a set of rules that define an *if-then* relation between atomic propositions. Our rule mining approach was as follows: Importantly, we did NOT mine temporal

rules (as we focus on structural rules in this work). So the process instances in the datasets were "flattened" s.t. we analyzed only the co-occurrence of different facts in individual instances. Here, a rule $a \Rightarrow b$ denotes that the occurrence of a implies the occurrence of b. In the MINERful tool, such a rule $a \Rightarrow b$ is denoted as a rule of type "CoExistence". The inverse is "NotCoExistence", which denotes that a reaction b should not follow from an activation a in the instance. These rules as mined from the datasets were transferred to the general rule form in (1) as used in this work. Note that the presented implementation is not limited to rules with only one premise. Also observe that there can be some other rule types beyond the scope of this work, e. g., regarding cardinalities, which were therefore filtered out. In result, using the MINERful tool, we can obtain rule bases of the general form as considered in this work (see below for link to the obtained rule bases).

For our evaluation, we used the available BPIC datasets of the last 10 years.[6] In Table 1, we show an overview of the analyzed datasets, including the domain, the number of (customer) cases contained in the dataset, as well as the number of distinct facts (i. e., activations and reactions) that can appear as literals in the respective rule bases. As can be seen, the considered datasets reflect real-life case data of industrial complexity, with up to 250.000 cases. As mining parameters, we selected standard mining parameters as suggested in [15], namely a support factor of 75% (minimum number of cases a rule has to be fulfilled in), as well as confidence and interest factors of 12.5% (support scaled by the ratio of cases in which the activation occurs, resp. support scaled by the ratio of cases both the activation and reaction occur). All obtained rule sets can be found online[7]. Table 1 also indicates the size of these obtained rule sets. As shown, the sizes of the considered rule bases range up to 1000 rules.

[6]Note that public case data was not provided for the BPIC 2014, so this year is omitted. In some years, the BPIC data sets contain multiple log files, which are marked accordingly.

[7]https://cloud.uni-koblenz.de/s/XLofLbCx82axtx5

	dataset	domain	# of cases	# of facts	# of rules
1	BPI2012	Financial Industry (Loan process)	13 087	24	75
2	BPI2013_1	Car manufacturing process (Volvo)	819	4	3
3	BPI2013_2	-	1 487	4	1
4	BPI2013_3	-	7 554	3	2
5	BPI2015_1	Government Sector (Permit process)	937	398	615
6	BPI2015_2	-	645	410	1013
7	BPI2015_3	-	1 087	383	594
8	BPI2015_4	-	787	356	721
9	BPI2015_5	-	892	389	922
10	BPI2016	Healthcare (Sepsis treatment process)	1 050	16	81
11	BPI2017	Financial Industry	31 509	26	144
12	BPI2018	Government (EU fund distribution process)	43 809	41	30
13	BPI2019	Industry (Purchase process)	251 734	42	25
14	BPI2020_1	Government (Travel reimbursement process)	6 886	17	20
15	BPI2020_2	-	7 065	34	84
16	BPI2020_3	-	2 099	51	51
17	BPI2020_4	-	6 449	29	33
18	BPI2020_5	-	10 500	19	23

Table 1: Overview of the considered datasets including industrial domain, number of cases, number of facts, and size of the rule-base obtained via the described mining tool.

We now continue to present our experiment results.

5.2.2 Experiment Results

For the considered datasets, we computed a) *all* potential issues, and b) only the *top-k* potential issues (with $k = 50\%, 60\%, ..., 90\%$). The experiments were run on Mac OS with 3GHz i7 processor and 16 GB RAM. As we tested on fixed data-sets and not random ones, re-running the tests yielded very similar results in our setting, so we ommit a discussion of standard deviations etc. in the following.

As already discussed, the number of returned top-k potential issues relative to a threshold k will strongly depend on the actual distribution of fact occurrences. Interestingly, for all considered datasets, the occurrence distribution was very similar, with only a few facts that occured many times, and many facts that occured seldomly (cf. also the example in Figure 1). In our opinion, this re-affirms our motivation of using the notion of probability to rank potential issues by their severity (i. e., there will be many potential issues which have a very low probability of being actually "activated" at run-time). The fact occurrence distributions for all consid-

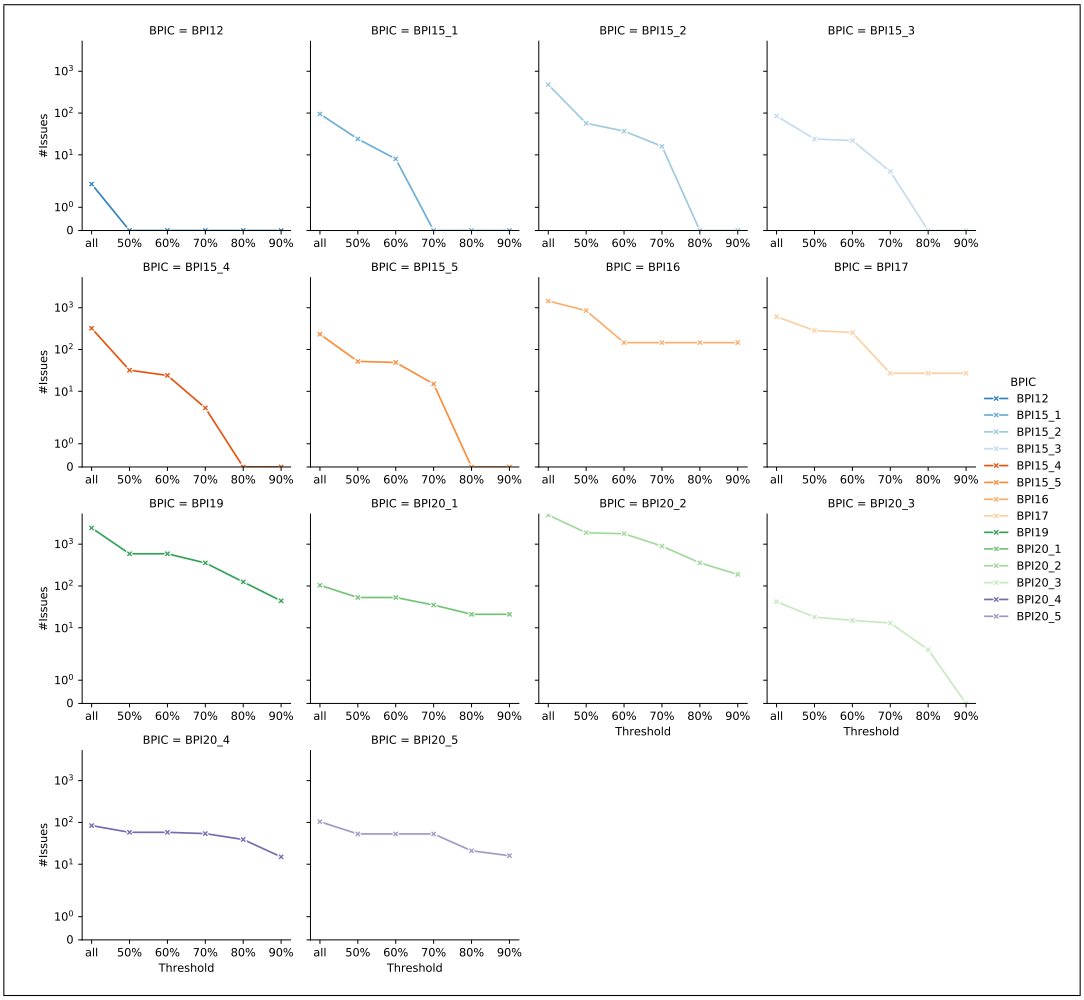


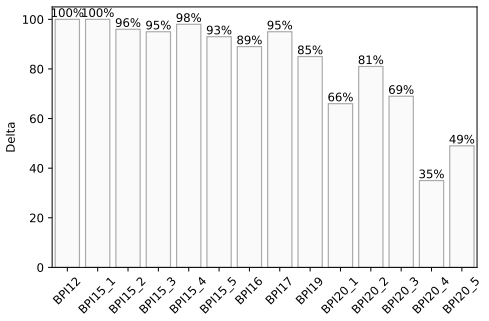
Figure 2: Total number of potential issues (y-axis) for different thresholds k (x-axis; where x includes: all issues; top-k issues for $k = 50\%, ..., 90\%$) for the considered datasets.

ered datasets can be found in the appendix of this work. The reader can easily see that those occurrences behave very similar to the one shown in Figure 1.

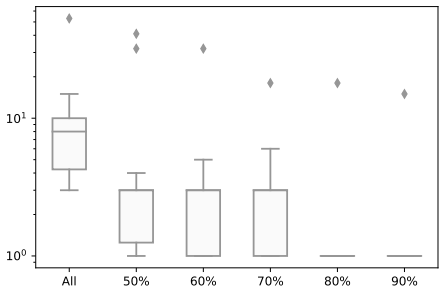
Continuing, the results of our evaluation are as shown in Figure 2. For every dataset, the corresponding plot shows the total number of potential issues that exist in the rule set (all pot. issues), as well as the concrete number of top-k potential

issues w.r.t. a threshold k (with $k = 50\%, ..., 90\%$).[8] The y-axis represents the found number of issues and the x-axis represents the discrete different settings for k.

For all considered datasets, there was a clear drop of the returned number of potential issues when selecting a parameter of $k \geq 50\%$. This shows that setting a minimal probability threshold (as to the probability of a potential issue being actually activated at run-time) can clearly reduce the number of potential issues that have to be analyzed. For a minimal threshold of $k = 80\%$, there were many datasets where the number of returned potential issues dropped to zero. Based on these results, it seems promising to use our approach in order to reduce the load for humans as many potential issues with very small probabilities can be "filtered out". This can be seen in Figure 3 (a), which shows the percental savings of potential issues that do not have to be screened when the minimal threshold is set to $k = 50\%$, (vs. the case that *all* potential issues have to be screened).



(a) Percentage of potential issues that can be filtered out for the considered datasets when setting a threshold of $k = 50\%$.



(b) Distribution of runtimes (ms) needed to compute all potential issues, as well as only the top-k potential isses (for $k = 50\%$-90%) for the considered datasets.

Figure 3: Overview of savings (percentage of issues that can be filtered for k = 50% (a); runtimes (b)) when computing only the top-k potential issues.

As can be seen, setting the minimal threshold to 50% saved on average 82% of potential issues (much higher for higher ks), with many of the savings ranging upwards of 90%. We do not dispose over the domain knowledge to assess which minimal threshold can be plausibly set in the considered domains. However, as part of this initial evaluation, we can show that setting the minimal threshold to 50% can save on average 82% of potential issues that have to be analyzed (across the

[8]The datasets for BPI13 and BPI18 did not conatin any potential issues and were therefore omitted.

considered datasets), which we believe could be a valuable benefit for companies. Not surprisingly, runtimes also drop off nicely when a minimal threshold is set, cf. Figure 3 (b). We do acknowledge that computation was very feasible for the considered datasets, even for computing all potential issues (within the milliseconds). So for the analyzed datasets, this factor may be discarded - we see a much higher advantage in the fact that far less potential issues have to be presented to the human when setting a threshold. Note however that computing potential issues is unfeasible from the perspective of computational complexity, so the factor that runtime can be reduced by setting a minimal threshold should not be neglected in general. In future works, it would be interesting to analyze further datasets, if available.

Intuitively, how to select the parameters is highly dependent on the use-case and has high impact on the algorithm performance. The underlying idea of our "top-k" approach however remains to "live with" certain potential problems if their severity (e.g., based on probability) is below an acceptable threshold, which is in line with many ideas and approaches for paraconsistent reasoning techniques [11].

6 Conclusion

In this paper, we introduced the notion of potential issues in rule bases. Our results allow experts to identify those sets of rules that, if activated together, will cause inconsistencies during run-time. Importantly, it is usually not clear which rules might become activated, so deciding how to handle potential issues requires a careful trade-off of how likely certain rules might become activated together. To this end, we introduced the notion of a severity of a potential issue based on fact probabilities, which allows to present a prioritized list of potential issues. Here, our proposed approach allows to compute top-k potential issues in an online manner, i.e., without the need to pre-compute the entire set of potential issues (which might be intractable).

Our approach is applicable for many real-life applications such as rule modeling or, in particular, rule mining. The developed implementation can directly be integrated into state of the art rule mining tools, e.g. [3], to facilitate a post-processing for rule mining algorithms. It is widely acknowledged that state of the art mining approaches can suffer from yielding inconsistent rule sets (cf. [15]), so integrating means for handling potential issues seems in line here.

A limitation of our proposed approach for computing the top-k potential issues is that the notion of severity is based on how likely an issue is to be activated. Intuitively, there could be other notions of severity, e.g., even unlikely potential issues can be "severe" if the affected rules are business-critical. In future works, it

would be useful to support modellers also with other forms of potential issue analysis, possibly also investigating postulates that confine their "severity" for specific use-cases. Notions such as inconsistency cost could be used as a basis towards this direction. Also, the approach proposed in this work operates on a syntactical level, so it might be promising to also consider the semantics in future works.

In regard to knowledge engineering, the work at hand also sheds light on methodological aspects related to (collaborative) rule modelling and knowledge lifecycles. At the core, the problem of potential issues can arise due to unforseen interactions between different pieces of knowledge. A naive strategy to counteract this problem would be to model dedicated rules for *all* possible combination of fact inputs (e. g., as exceptions). However, such approaches seem highly unfeasible in practice, as this could lead to an exponential explosion of required rules needed to capture all cases, i.e., the rule base would not be maintainable anymore due to a lack of oversight. In fact, recent works acknowledge this problem of exponential explosions when wanting to capture all possible input combinations and explicitly recommend rule modelling methodologies based on having *incomplete* rule bases [18, 28]. Here, the notion of potential issues introduced in this work supports such recent knowledge engineering methods by allowing experts to analyze (incomplete) knowledge bases for any potential issues. In this context, the introduced notion of a *severity* of a potential issue also supports companies to select a cut-off point as to which issues need to be handled through explicit re-modelling, and which issues can be ignored in favour of lower rule base complexity. In the scope of verication and validation of rule-based systems [30], the work at hand can be related to works on *a priori* revision [12], with the goal of preventing issues by adding further constraints. However in this setting, a set of rules is assumed to be given, which is then further processes. For future works, it might be useful to investigate how rule modelling methodologies can be extended to facilitate proactively preventing potential issues directly during modelling.

References

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *International Conference on Management of Data*, pages 207–216, 1993.

[2] Carlos E Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The journal of symbolic logic*, 50(2):510–530, 1985.

[3] Anti Alman, Claudio Di Ciccio, Dominik Haas, Fabrizio Maria Maggi, and Alexander Nolte. Rule mining with rum. In *2nd International Conference on Process Mining*

(ICPM 2020), pages 121–128, 2020.

[4] Philippe Besnard. A logical analysis of rule inconsistency. *International Journal of Semantic Computing*, 5(03):271–280, 2011.

[5] Magnus Boman, Janis A Bubenko Jr, Paul Johannesson, and Benkt Wangler. *Conceptual modelling*. Prentice-Hall, Inc., 1997.

[6] Carl Corea and Patrick Delfmann. Quasi-inconsistency in declarative process models. In *International Conference on Business Process Management (BPM Forum 2019)*, pages 20–35, 2019.

[7] Carl Corea and Patrick Delfmann. A taxonomy of business rule organizing approaches in regard to business process compliance. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 15:4–1, 2020.

[8] Carl Corea and Matthias Thimm. On quasi-inconsistency and its complexity. *Artifial Intelligence*, 284:1–15, 2020.

[9] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[10] Nadia Creignou and Miki Hermann. On #p-completeness of some counting problems. Technical report, INRIA, 1993.

[11] Johan De Kleer. An assumption-based tms. *Artificial intelligence*, 28(2):127–162, 1986.

[12] Florence Dupin de Saint-Cyr, Béatrice Duval, and Stéphane Loiseau. A priori revision. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 488–497. Springer, 2001.

[13] Cristhian A. D. Deagustini, M. Vanina Martinez, Marcelo A. Falappa, and Guillermo R. Simari. How does incoherence affect inconsistency-tolerant semantics for datalog±? *Annals of Mathematics and Artificial Intelligence*, 82:43–68, 2018.

[14] Cristhian Ariel D Deagustini, Maria Vanina Martinez, Marcelo A Falappa, and Guillermo R Simari. Datalog+-ontology consolidation. *Journal of Artificial Intelligence Research*, 56:613–656, 2016.

[15] Claudio Di Ciccio, Fabrizio Maria Maggi, Marco Montali, and Jan Mendling. Resolving inconsistencies and redundancies in declarative process models. *Information Systems*, 64:425–446, 2017.

[16] Dragan Doder and Srdjan Vesic. How to decrease and resolve inconsistency of a knowledge base? In *International Conference on Agents and Artificial Intelligence*, volume 2, pages 27–37. SCITEPRESS, 2015.

[17] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A Reijers, et al. *Fundamentals of business process management*, volume 1. Springer, 2013.

[18] Alan N Fish. *Knowledge automation: how to implement decision management in business processes*, volume 595. John Wiley & Sons, 2012.

[19] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *National Conference on Artificial Intelligence*, pages 1295–1300. AAAI Press, 2006.

[20] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive

databases. *New Generation Computing*, 9:365–385, 1991.

[21] Ian Graham. *Business rules management and service oriented architecture: a pattern language*. John wiley & sons, 2007.

[22] John Grant and Anthony Hunter. Measuring consistency gain and information loss in stepwise inconsistency resolution. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 362–373. Springer, 2011.

[23] Sven Ove Hansson. A survey of non-prioritized belief revision. *Erkenntnis*, 50(2):413–427, 1999.

[24] John McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial intelligence*, 28(1):89–116, 1986.

[25] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[26] Jana-Rebecca Rehse and Peter Fettke. Process mining crimes–a threat to the validity of process discovery evaluations. In *International Conference on Business Process Management*, pages 3–19. Springer, 2018.

[27] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[28] Bruce Silver. *DMN method and style: The practitioner's guide to decision modeling with business rules*. Cody-Cassidy Press, 2016.

[29] Matthias Thimm. Inconsistency measurement. In *Proceedings of the 13th International Conference on Scalable Uncertainty Management (SUM'19)*, pages 9–23, December 2019.

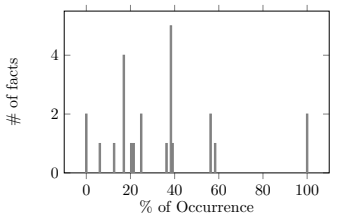
[30] Jan Vanthienen, Christophe Mues, and Ann Aerts. An illustration of verification and validation in the modelling phase of kbs development. *Data & Knowledge Engineering*, 27(3):337–352, 1998.

[31] Mathias Weske. *Business process management architectures*. Springer, 2007.

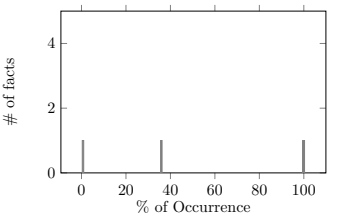
[32] Du Zhang. Inconsistency: the good, the bad, and the ugly. In *2009 IEEE International Conference on Information Reuse & Integration*, pages 182–187. IEEE, 2009.

Received 15 January 2024

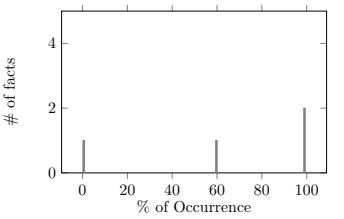
Appendix: Fact Occurrence Distributions for the BPI Data-Sets



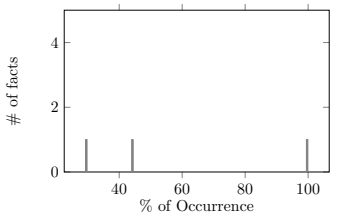
(a) BPI 2012



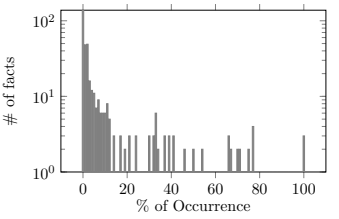
(b) BPI 2013_1



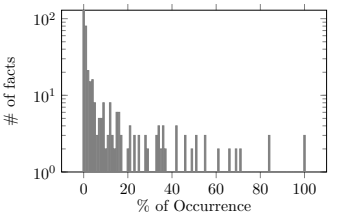
(c) BPI 2013_2



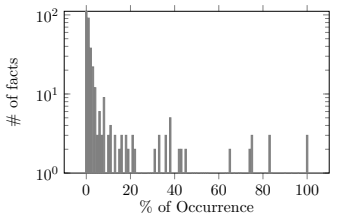
(d) BPI 2013_3



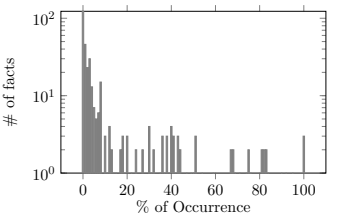
(e) BPI 2015_1



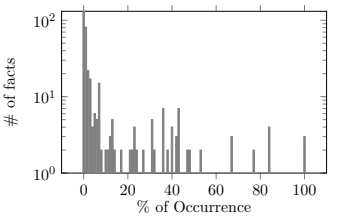
(f) BPI 2015_2



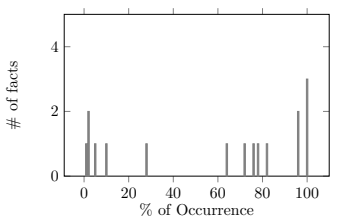
(g) BPI 2015_3



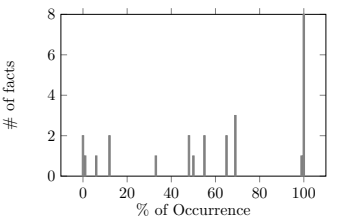
(h) BPI 2015_4



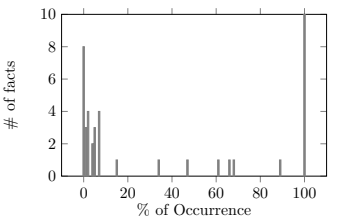
(i) BPI 2015_5



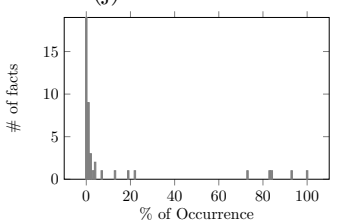
(j) BPI 2016



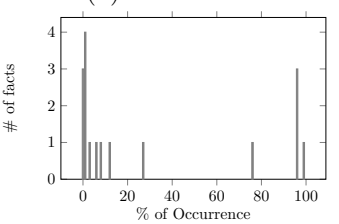
(k) BPI 2017



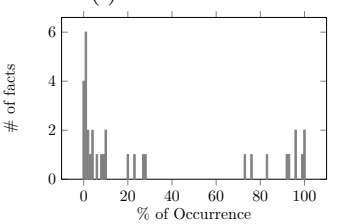
(l) BPI 2018



(m) BPI 2019



(n) BPI 2020_1



(o) BPI 2020_2

Received 15 January 2024