Checking Process Compliance against Natural Language Specifications using Behavioral Spaces

Han van der Aa[a,*], Henrik Leopold[a], Hajo A. Reijers[a,b]

[a]Department of Computer Sciences, VU University Amsterdam, Faculty of Sciences,
De Boelelaan 1081, 1081HV Amsterdam, The Netherlands
[b]Department of Mathematics and Computer Science, Eindhoven University of Technology,
PO Box 513, 5600MB Eindhoven, The Netherlands

Abstract

Textual process descriptions are widely used in organizations since they can be created and understood by virtually everyone. Because of their widespread use, they also provide a valuable source for process analysis, such as compliance checking. However, the inherent ambiguity of natural language impedes the automated analysis of textual process descriptions. While human readers can use their context knowledge to correctly understand statements with multiple possible interpretations, automated tools currently have to make assumptions about their correct meaning. As a result, compliance-checking techniques are prone to draw incorrect conclusions about the proper execution of a process. To provide a comprehensive solution to these reasoning problems, we use this paper to introduce the concept of a *behavioral space* as a means to deal with behavioral ambiguity in textual process descriptions. A behavioral space captures all possible interpretations of a textual process description in a systematic manner. Thus, it avoids the problem of focusing on a single, possibly incorrect interpretation. We use a quantitative evaluation with a set of of 47 textual process descriptions to demonstrate the usefulness of a behavioral space for compliance checking in the context of ambiguous texts.

Keywords: business process analysis, compliance checking, natural language processing, ambiguity

[*]Corresponding author. *Phone number:* +31 20 59 87788
Email addresses: j.h.vander.aa@vu.nl (Han van der Aa), h.leopold@vu.nl (Henrik Leopold), h.a.reijers@vu.nl (Hajo A. Reijers)

1. Introduction

Non-compliance represents a risk for many organizations. According to a recent study by Thomson Reuters, non-compliance may even represent a possible cause of bankruptcy, also for the so-called "behemoths" in the financial sector [1]. Recognizing the risk that is associated with non-compliance, organizations in a wide range of domains are stepping up their spending in order to ensure their compliance with laws, regulations, and procedures. In this context, automated compliance checking techniques play a crucial role thanks to their ability to automatically identify compliance violations [2, 3]. For this reason, numerous approaches have been developed to perform this task (cf. [4, 5, 6, 7]). What these compliance-checking techniques have in common is that they rely on a structured specification of allowed behavior, for example in the form of *process models* or *business rules*. As a result, these techniques ignore the wealth of information that is contained in less structured forms of process documentation, such as textual process descriptions [8].

While the relevance and widespread use of text documents as a source for process analysis has been emphasized in various contexts [9, 10, 11, 12], the inherent ambiguity of natural language presents a considerable challenge to compliance-checking techniques. For example, a simple natural language statement such as "*in parallel to the latter steps*" leaves room for interpretation. Due to this statement's ambiguity, it is generally impossible to infer with certainty whether "*latter*" refers to the preceding two, three, or even more activities mentioned in the textual description. In prior work, text-to-process model generation techniques have circumvented this problem by introducing interpretation heuristics [9, 13, 14]. In this way, these techniques obtain a single process-oriented interpretation of the text, in spite of the presence of ambiguous sentences. This interpretation, however, contains assumptions on the correct interpretation of essentially undecidable ambiguity issues. So, there is always the risk that the derived interpretation conflicts with the proper way to execute the process. As a result, the focus on a single, assumed interpretation can lead to incorrect and, thus, untrustworthy compliance-checking results.

To provide a rigorous solution for the reasoning problems caused by ambiguous natural language statements, we introduce a novel concept which we refer to as a *behavioral space*. A behavioral space precisely captures all possible behavioral interpretations of a textual process description. The behavioral space clearly defines which behavior is within and which behavior is outside any reasonable bounds of interpretation. By using behavioral spaces for compliance checking, we avoid the need to impose assumptions on the correct interpretations of ambiguous natural language texts. Therefore, compliance checks based on behavioral spaces provide trustworthy results: They avoid the risks associated with the selection of incorrect interpretations.

The remainder of the paper is structured as follows. Section 2 motivates the problem of reasoning under behavioral ambiguity in textual process descriptions. In Section 3, we introduce the notion of a behavioral space to capture behavioral ambiguity. Section 4 describes how a behavioral space can be generated from a textual process description. We show how to perform compliance checks using a behavioral space in Section 5. Then, Section 6 introduces a semi-automated pruning technique that can be used to effectively reduce the uncertainty in compliance-checking results. In Section 7, we demonstrate the usefulness of behavioral spaces and our proposed pruning technique through a quantitative evaluation using real-world data. Section 8 discusses streams of related work. Finally, we conclude the paper and discuss directions for future research in Section 9.

2. Behavioral Ambiguity in Textual Process Descriptions

In this section, we illustrate the problem associated with compliance checking of process behavior against textual process descriptions. The key challenge in this context is the inherent ambiguity of natural language. *Ambiguity* in natural language refers to a type of uncertainty in which several interpretations of the same text are plausible. For example, the sentence "*I saw a man on the hill with a telescope*" can have at least five plausible interpretations. These interpretations vary, among others, on who is on the hill (*I* or the *man*) and on the possessor or the location of the telescope (*I*, the *man*, or it is *on the hill*). In certain situations, the correct interpretation of an ambiguous statement can be clear from the context in which it is used, whereas in other situations even context cannot help to resolve ambiguity.

The goal of compliance checking is to determine if some observed behavior (i.e. a sequence of performed activities) conforms to the allowed behavior described by a process specification, i.e. a textual process description. Therefore, in a compliance-checking context we are particularly concerned with ambiguity related to the allowed process behavior described in a text, which we shall refer to as *behavioral ambiguity*. Behavioral ambiguity occurs when statements about the relations that exist between process steps can

After a claim is received, a claim officer reviews the request and records the claim information. The claim officer then validates the claim documents before writing a settlement recommendation. A senior officer then checks this recommendation. The senior officer can request further information from the claimant, or reject or accept the claim. In the former case, the previous steps must be repeated once the requested information arrives. If a claim is rejected, the claim is archived and the process finishes. If a claim is accepted, the claim officer calculates the payable amount. Afterwards, the claims officer records the settlement information and archives the claim. In the meantime, the financial department takes care of the payment.

Figure 1: Exemplary description of a claims handling process.

be interpreted in different ways. We illustrate the problem of behavioral ambiguity through the simplified description of a claims handling process, as presented in Figure 1. The description uses typical patterns to describe ordering relations, as observed in process descriptions obtained from practice and research [9].

At first glance, the description from Figure 1 may appear to be clear. However, on closer inspection, it turns out that the description does not provide conclusive answers to several questions regarding the proper execution of the described process. For instance:

Q1. Is it allowed that the claims officer records the claim information *before* reviewing the request?

Q2. Does it suffice for the claim officer to rewrite the settlement recommendation in case additional information has been requested?

Q3. Can the financial department start paying the claimant while the settlement information is still being recorded?

Based on the information provided in the textual description, these questions are not clearly decidable. This lack of decidability results from two forms of behavioral ambiguity: type ambiguity and scope ambiguity. *Type ambiguity* occurs when a textual description does not clearly specify the type of order relationship between two activities. For instance, the relation between the "*review request*" and "*record claim information*" activities in the first sentence is unclear. The term "*and*" simply does not allow us to determine whether these activities must be executed sequentially or whether they can be executed in an arbitrary order (Q1). *Scope ambiguity* occurs when statements in a textual description underspecify to which activity or activities they precisely refer. This type of ambiguity particularly relates to repetitions and parallelism. For instance, the statement "*the previous steps must be repeated*" does not clearly specify which activities must be performed again (Q2). Similarly, the expression "*in the meantime*" does not define when the financial department can start performing its activities (Q3).

As a result of such ambiguities, there are different views on how to properly carry out the described process. When deriving a single structured interpretation from a textual process description, as done by process model generation techniques (cf. [9, 13, 14]), there is always the risk that a derived interpretation conflicts with the proper way to execute the process. The focus on a single interpretation can, therefore, lead to wrong conclusions when reasoning about a business process. This can, for instance, result in a loss of efficiency by not allowing for parallel execution where possible (Q3). Furthermore, it can result in non-compliance with regulations, for example, by failing to impose necessary ordering restrictions (Q1) or by not repeating all of the required steps when dealing with the receipt of new claim information (Q2).

To avoid the problems associated with using an assumed interpretation, automated reasoning techniques should take into account all reasonable interpretations of a textual process description. Therefore, we use this paper to introduce the concept of a *behavioral space*. A behavioral space allows us to capture the full range of possible semantics that can be conveyed by textual descriptions in a structured manner. As such, it provides the basis to correctly reason about compliance to described processes.

3. Capturing Behavioral Ambiguity using Behavioral Spaces

In this section, we introduce and define the concept of a *behavioral space*. This concept provides the foundation to reason about behavioral compliance for ambiguous process descriptions. The general idea underlying behavioral spaces is to represent the causes and effects of behavioral ambiguity in a structured manner. In Section 3.1, we first consider how to capture the various possible interpretations of a single ambiguous behavioral statement. Then, Section 3.2 defines the concept of a behavioral space as a means to capture all possible ways to interpret an entire textual process description.

3.1. Behavioral Statement Interpretations

Textual process descriptions consist of statements, i.e. sentences or parts of sentences, that describe ordering relations between activities. We shall refer to such statements as *behavioral statements*. Each behavioral statement describes a single type of relation that holds between two or more activities. For example, the statement "*After a claim is received, a claims officer reviews the request*" describes a sequential relation between "*receive claim*" and "*review request*". These relations described in behavioral statements can be translated into a structured notation. In this paper, we employ the *behavioral profile relations* from [5] for this purpose. This choice is based on two main criteria. First, behavioral profile relations allow for an intuitive representation of the concepts we introduce in the remainder of this paper. Second, the relations can be used for computationally efficient compliance checks. This feature is important because the computational complexity of compliance checks increases with the number of interpretations of a textual process description.

The behavioral profile relations capture the ordering restrictions that are in effect between pairs of activities. Four different behavioral profile relations can exist for an activity pair (a_i, a_j). The *strict order* relation $a_i \rightsquigarrow a_j$ is used to express that activity a_i cannot be executed after the execution of activity a_j. The *reverse strict order relation* $a_i \rightsquigarrow^{-1} a_j$ indicates the opposite restriction, namely that a_i cannot be executed after the execution of a_i.[1] The *exclusiveness* relation $a_i + a_j$ denotes that either activity a_i or activity a_j can be executed in a single process instance. Finally, the *interleaving order* relation $a_i \parallel a_j$ states that a_i and a_j can be executed in an arbitrary order. Using the activity identifiers specified in Table 1, this results in the relation $a_1 \rightsquigarrow a_2$ as a relation that holds between *receive claim* and *review request* .

Table 1: Activities in the running example

ID	**Activity**	**ID**	**Activity**
a_1	Receive claim	a_8	Reject claim
a_2	Review request	a_9	Accept claim
a_3	Record claim information	a_{10}	Receive requested information
a_4	Validate documents	a_{11}	Calculate payable amount
a_5	Write settlement recommendation	a_{12}	Record settlement information
a_6	Check recommendation	a_{13}	Archive claim
a_7	Request further information	a_{14}	Arrange payment

Behavioral ambiguity in a textual process description occurs when behavioral statements can be interpreted in different manners. We refer to these statements as *ambiguous behavioral statements*. These statements result in conflicting activity relations. For instance, the statement "*a claim officer reviews the request and records the claim information*" results in two interpretations. It is unclear whether this statement implies a strict order or an interleaving order between the two described activities. This leads to two

[1] Note that the reverse strict order relation $a_i \rightsquigarrow^{-1} a_j$ can only exist if and only if $a_j \rightsquigarrow a_i$.

different possible behavioral relations, namely $a_2 \rightsquigarrow a_3$ and $a_2 \parallel a_3$.

In the remainder, we use the term *statement interpretation* to refer to the various behavioral relations that are associated with a single interpretation of a behavioral statement. Unambiguous behavioral statements result in a single statement interpretation, whereas ambiguous statements lead to multiple interpretations. Definition 1 capture this notion.

Definition 1 (Statement Interpretations)**.** *Let s be a behavioral statement in the set of behavioral statements S_T of a textual process description T, A_T the set of activities described in T, and $\mathcal{R} = \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \|\}$ the set of behavioral profile relations. We define Γ_s as a set consisting of one or more statement interpretations for the statement s. Each* statement interpretation *$\gamma \in \Gamma_s$ captures the behavioral relations that follow a possible interpretation of s, which is defined as a partial function $\gamma : A_T \times A_T \nrightarrow \mathcal{R}$ that assigns a behavioral profile relation from $\mathcal{R}$ to a pair of activities from A_T, if any.*

Next, we discuss how these statement interpretations serve as a basis for the establishment of behavioral spaces, which consist of interpretations of an entire textual process description.

3.2. Behavioral Spaces

Using the interpretations of individual statements as a basis, we can construct views on the full process behavior described in a textual description. We refer to a specific view as a *process interpretation*. A process interpretation for a text T follows from the selection of a single *statement interpretation* for each statement s in the set of behavioral statements S_T. We define a process interpretation as given in Definition 2.

Definition 2 (Process Interpretation)**.** *Let T be a textual process description, A_T the set of activities described in T, S_T the set of behavioral statements in T with Γ_s the set of statement interpretations of a statement $s \in S_T$, and $\mathcal{R} = \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \|\}$ the set of behavioral profile relations. We then define a* process interpretation *as a tuple $P = (\mathcal{I}, BP)$, with:*

- *$\mathcal{I}$: a complete set of interpretations consisting of a single statement interpretation $\gamma \in \Gamma_s$ for each statement $s \in S_T$, formally the following constraint holds: $\forall s \in S_T : |\mathcal{I} \cap \Gamma_s| = 1$;*
- *$BP : A_T \times A_T \rightarrow \mathcal{R}$ a function that assigns a behavioral profile relation from $\mathcal{R}$ to each pair of activities from A_T.*

Recognize that in Definition 2 any activity relation that follows from a function $\gamma \in \mathcal{I}$ is part of BP, but that the reverse does not necessarily hold. BP defines a complete behavioral profile based on the interpretations included in $\mathcal{I}$, which includes relations that follow from the transitivity of the strict order and interleaving order relations [15]. For instance let $\gamma_1 \in \mathcal{I}$ be a statement interpretation that establishes the relation $a \rightsquigarrow b$ and let $\gamma_2 \in \mathcal{I}$ be a statement interpretation that establishes $b \rightsquigarrow c$. Then, BP will include the relation $a \rightsquigarrow c$ that follows due to transitivity[2], even though this relation is not part of any statement interpretation in $\mathcal{I}$. Despite the overlap between them, we include $\mathcal{I}$ alongside BP in Definition 2. This is done in order to preserve traceability between the obtained process behavior described by BP and the statement interpretation in $\mathcal{I}$ that provided the foundation for BP. As such, we can use this traceability to provide more useful diagnostic results when performing compliance checks.

A textual process description without ambiguity has exactly one process interpretation. For a textual process description T with behavioral ambiguity, the set of possible process interpretations follows naturally as the set of possible combinations of statement interpretations for the ambiguous statements. For example, a text with two ambiguous behavioral statements s_i and s_j, with, respectively, two and three different statement interpretations, will yield a set of $2 \times 3 = 6$ process interpretations. The concept of a behavioral space captures this spectrum of possible interpretations for a single process. Figure 2 visualizes this, by depicting a three-dimensional view on the behavioral relations that exist between the activities in an ambiguous textual description. Definition 3 provides the formal definition of a behavioral space.

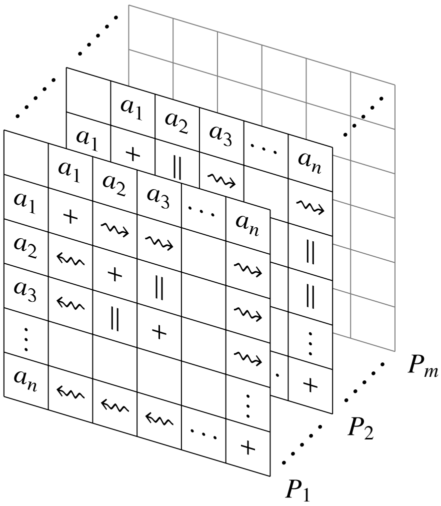
Definition 3 (Behavioral Space)**.** *Let T be a textual process description, A_T the set of activities described in T, and S_T the set of behavioral statements in T. We define a* behavioral space $\mathcal{BS}_\mathcal{T}$ *as a set of process interpretations of the textual process description T.*

In Section 4, we consider how to automatically generate behavioral spaces from textual process descriptions.

4. Constructing Behavioral Spaces

This section describes an approach to automatically generate a behavioral space from a textual process description. The approach, visualized in Figure 3, consists of three steps. First, a textual process description T is parsed in order to identify and analyze the set of behavioral statements S_T. Second, the proposed

[2] Note that this only applies if no interpretation $\gamma \in \mathcal{I}$ denotes a (different) relation between activities a and c.

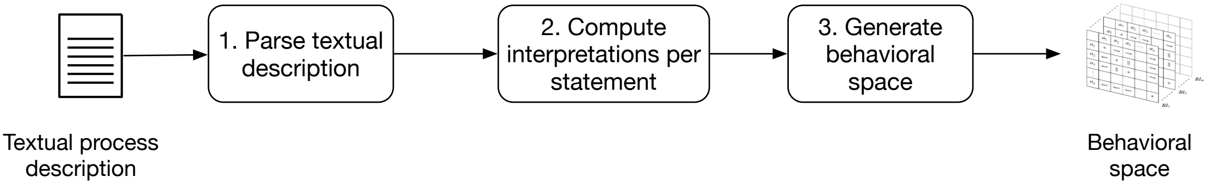


	a_1	a_2	a_3	$\dots$	a_n
a_1	+	⇝	⇝		⇝
a_2	⇜	+	∥		⇝
a_3	⇜	∥	+		⇝
$\vdots$					$\vdots$
a_n	⇜	⇜	⇜	$\dots$	+

Figure 2: A behavioral space as a collection of *m* process interpretations

approach generates behavioral interpretations for each statement in S_T. Third and lastly, the different statement interpretations are combined into a collection of process interpretations that, together, comprise the behavioral space $\mathcal{BS}$.

In the remainder of this section, we present the details on each of these three steps. Given the focus on behavioral ambiguity of this paper, we mainly describe those aspects of the generation procedure that are specific to the consideration of ambiguity. Existing text-to-process model generation approaches, cf. [9], already address most challenges related to the parsing of textual descriptions (step 1) and related to the extraction of behavioral relations for *unambiguous* behavioral statements (part of step 2).



1. Parse textual description
2. Compute interpretations per statement
3. Generate behavioral space
Textual process description
Behavioral space

Figure 3: Steps involved to construct a behavioral space from a textual description

4.1. Parsing Textual Process Descriptions

The first step in the approach is to parse a textual process description T. The goal of this parsing step is to identify the set of behavioral statements S_T and to extract behavioral information from these statements.

Each behavioral statement in a text describes a single type of relation that holds between a number of activities. Therefore, a sentence in a textual description can contain more than a single behavioral statement. For example, the sentence "*After a claim is received, a claim officer reviews the request and records the claim information*", contains two separate behavioral statements. One statement describes the strict order relation between "*claim is received*" and "*reviews the request*". The other describes an ambiguous relation between "*reviews the request*" and "*records the claim information*".

For each behavioral statement in S_T, the parsing step extracts the parts of the statement that refer to process concepts and their inter-relations. As such, the parser identifies references to three *semantic components* that together comprise a behavioral relation of the form: <*source*, *relation type*, *target*>. These components are a *source* reference, a *relation type* reference, and a *target* reference. We speak about *references*, rather than *process concepts* here, because textual descriptions often use different ways to refer to the same concept. For instance, "*previous activity*" provides a reference to some aforementioned activity; it is not a process concept itself. To clarify the parsing results obtained in this manner, we present the parsing results for three statements in Table 2. We will use these examples in the remainder of this section to further illustrate the algorithm that generates statement interpretations.

Table 2: Exemplary outcomes of behavioral statement parsing

ID	**Statement**	**Source ref.**	**Relation ref.**	**Target ref.**
s_1	After a claim is received, a claim officer reviews the request	*claim is received* (a_1)	*after*	*reviews the request* (a_2)
s_2	a claim officer reviews the request and records the claim information	*reviews the request* (a_2)	*and*	*records the claim information* (a_3)
s_3	In the meantime, the financial department takes care of the payment	ϵ	*meantime*	*takes care of payment* (a_{14})

The techniques necessary to perform this parsing step are equal to the parsing used by existing text-to-model generations approaches. These approaches use a combination of standard natural language processing (NLP) tools and *heuristic-based* techniques. NLP tools, such as the *Stanford Parser* [16], are used to identify the grammatical structure of sentences. This structure is important to extract the business object on which an action is being performed and to identify the actor who is performing the action. For example, the "*claims officer*" (actor) "*reviews*" (action)"*the request*" (business object). While the NLP tools mainly provide general techniques to analyze individual sentences and, thereby, extract activities, more tailored techniques are necessary to extract the ordering relations that exist between activities. For this purpose, model generation approaches employ heuristic-based techniques that recognize typical patterns used to express ordering relations. Such patterns include sequences like ”After *activity i*, do *activity j*” or choices like “If *condition*, then *activity i*, else *activity j*”. Since these techniques are extensively described in related work, we do not elaborate on them. The interested reader may consult the work by Friedrich et al. [9] for a detailed description of a state-of-the-art parsing technique.

4.2. Computing Statement Interpretations

In the second step, the proposed approach constructs interpretations for each behavioral statement in S_T. Algorithm 1 formalizes this part of the approach. The algorithm takes as input a parsed behavioral statement $s \in S_T$, in which the semantic components have been identified. Then, it generates one or more statement interpretations, depending on the presence and the type of ambiguity in S_T. Given a statement $s \in S_T$, we distinguish three cases for this: (i) s is unambiguous, (ii) s contains type ambiguity, or (iii) s contains scope ambiguity. To which of these three cases a statement belongs depends on the ability to resolve the textual references with certainty. The following sections describe each of these cases in detail.

Algorithm 1 Computing interpretations for a behavioral statement

```
function computeStatementInterpretations(ParsedStatement s)
    Set interpretations = new Set();
    if s.isUnambiguous() then                                   ▷ Statement is not ambiguous
        Set sourceActivities = s.getSourceActivities();
        Set targetActivities = s.getTargetActivities();
        RelationType r = s.getRelationType();
        interpretations.add(createInterpretation(sourceActivities, r, targetActivities));
    if s.hasTypeAmbiguity() then                                ▷ Relation type is ambiguous
        Set sourceActivities = s.getSourceActivities();
        Set targetActivities = s.getTargetActivities();
        for RelationType r ∈ RelationTypes.get(s.getRelationReference()) do
            interpretations.add(createInterpretation(sourceActivities, r, targetActivities));
    if s.hasScopeAmbiguity() then                 ▷ Scope of source reference is ambiguous
        Set targetActivities = s.getTargetActivities();
        RelationType r = s.getRelationType();
        Set sourceActivities1 = getPrecedingActivitiesWithSameResource(s);
        interpretations.add(createInterpretation(sourceActivities1, r, targetActivities));
        Set sourceActivities2 = getPrecedingActivitiesWithSameObject(s);
        interpretations.add(createInterpretation(sourceActivities2, r, targetActivities));
        Set sourceActivities3 = getActivitiesFromPrecedingControlFlowBlock(s);
        interpretations.add(createInterpretation(sourceActivities3, r, targetActivities));
return interpretations;
```

4.2.1. Unambiguous Behavioral Statements

Generating a process interpretation for an unambiguous statement is relatively straightforward: all references in the statement can be resolved in a single and unambiguous manner. For these cases, we can generate a single statement interpretation by simply resolving the references and constructing a behavioral relation. For example, for statement s_1, we can directly generate the relation $a_1 \rightsquigarrow a_2$. Algorithm 1 describes in lines 3–7 the creation of an interpretation for unambiguous statements. Since there is no ambiguity, we can automatically extract the set of source activities, target activities, and the relation type from the parsed statement.

4.2.2. Statements with Type Ambiguity

A behavioral statement with type ambiguity describes a relation among a specific set of activities, but does not clearly define the type of relationship. Such statements can be identified because they have an ambiguous way to refer to the relation type, i.e. the relation reference is ambiguous. In line 8, Algorithm 1 checks if statement s has type ambiguity by determining if the relation reference (*relationRef*) is a known ambiguous type indicator. In the context of this work, we focus on two ambiguous type indicators, namely the terms *and* and *or*. It is important to distinguish between cases where *relationRef* = "*and*" and cases where *relationRef* = "*and, then*" or similar. In the latter cases, the relation type is not ambiguous, since a sequential relation is clearly specified. Although these statements with unclear relation types are ambiguous, we can generate a set of statement interpretations that accurately capture the different possible interpretations.

If a statement indeed suffers from type ambiguity, the algorithm continues by resolving the unambiguous *source* and *target* activity references (lines 9–10). Afterwards, the algorithm generates a single statement interpretation for each applicable relation type (lines 11–12). For example, for statement s_2 from Table 2, we generate one statement interpretation that contains a sequential relation and one that contains an interleaving order relation between activities a_2 and a_3.

4.2.3. Statements with Scope Ambiguity

Dealing with behavioral statements with scope ambiguity represents the most complex case of the three. These statements describe the existence of a relation, but do not specify between which activities this relationship holds. In practice, such ambiguity occurs with respect to the set of *source* activities to which a behavioral statement refers. For example, the statement s_3 "*In the meantime, the financial department takes care of the payment*", does not specify to what activities "*meantime*" refers. Consequently, the source reference is empty (ϵ), as indicated in Table 2. Statements with scope ambiguity typically occur in the context of *parallelism* and *loops* because such behavioral patterns can be associated with textual references to groups of activities.

To automatically identify cases with scope ambiguity, we analyzed how existing techniques for the generation of process models from textual descriptions handle parallelism and loops. These techniques generally use heuristics to identify and analyze behavioral statements in a text. They build on predefined sets of indicators that pinpoint the different types of relations, e.g. "*while*" and "*in the meantime*" for parallel or interleaving order relations and "*is repeated*" for backward loops. Specifically, we analyzed the set of indicators used by the state-of-the-art technique from [9]. In this analysis, we isolated a subset of the parallelism indicators employed by the technique that typically result in statements with scope ambiguity. These indicators are presented in Table 3. What these indicators have in common is that their usage does not allow for the specification of a scope of the statement, i.e. these indicators cannot be associated with a source reference at all. Consider, for example, the sentence "*In the meantime, the financial department takes care of the payment*". It is syntactically not possible to specify to which set of activities a statement with *in the meantime* refers. By contrast, if we replace this construct with a non-ambiguous indicator, from the second set in Table 3, this problem can be avoided. For example, the indicator "*while*" can be used to specify the scope of the statement, e.g. "*While the claim is being archived*". By observing the usage of such ambiguous indicators, we can identify statements with scope ambiguity (line 13 of Algorithm 1) and generate interpretations for them (lines 14–21).

Table 3: Parallel indicators used in [9] and their classification

Class	**Contents**
Unambiguous	*while*, *as well as*, *in parallel to*
Ambiguous	*meanwhile*, *concurrently*, *meantime*, *in the meantime*

Although statements with scope ambiguity are highly problematic since they refer to an unknown set of activities, we can still identify a set of possible meanings for them. In particular, we can utilize the knowledge that statements such as "*the previous steps*" and "*in the meantime*" refer to distinct parts of a process. This means that the set of activities to which these statements refer *cannot* be any arbitrary combination of activities. The activities in the set must rather have a certain *commonality*, such as a set of activities that are all executed by the same person.

For this reason, we generate interpretations for statements with scope ambiguity based on sets of activities that have a certain common attribute. In particular, given a textual process description, we can identify sets of subsequently described activities that are (i) performed by the same resource, (ii) performed on or with the same (business) object, or (iii) are part of the same control-flow construct (e.g. a choice in the process). Based on this, we obtain different interpretations for the statement s_3. In this statement, "*in the meantime*" can refer to different moments when the financial department can start taking care of the payment. This results in three possible statement interpretations, each with a different commonality:

1. *Common resource:* While the *claims officer*, the last described resource, is performing its tasks, after a senior claims officer has accepted the claim, i.e. $\{a_{11}, a_{12}, a_{13}\}$;
2. *Common object:* While activities are being performed on the last mentioned business object (the *claim*), i.e. $\{a_{13}\}$;
3. *Last control-flow construct:* While the last mentioned activity before the statement is being executed, i.e. $\{a_{13}\}$.

These three possibilities result in three sets of relations that can follow from the same behavioral statement. Lines 14–21 describe the generation of a statement interpretation for each of the three sets of activities.

4.3. Generating Behavioral Interpretations

Based on the statement interpretations extracted from ambiguous and unambiguous behavioral statements, we can generate a set of process interpretations, i.e. the behavioral space, for the entire textual description. Algorithm 2 formalizes this generation step.

The set of process interpretations included in a behavioral space should contain all possible combinations of statement interpretations for the behavioral statements. Lines 3–14 in Algorithm 2 describe the creation of these combinations. The underlying idea is that all existing text interpretations, starting from

Algorithm 2 Generate a behavioral space from a textual process description

```
 1: function constructBehavioralSpace(Text text)
 2:     BehavioralSpace behavioralSpace = new BehavioralSpace();
 3:     List processInterpretations = new List();
 4:     processInterpretations.add(new ProcessInterpretation());
 5:     List newProcessInterpretations = new List();
 6:     for Statement s ∈ text.getStatements() do
 7:         List interpretations = computeStatementInterpretations(s);
 8:         for Interpretation i ∈ interpretations do
 9:             for ProcessInterpretation pi ∈ processInterpretations do
10:                 Interpretation newPI = pi.copy();
11:                 newPI.add(i);
12:                 newProcessInterpretations.add(newPI)
13:         interpretations = newInterpretations.copy();
14:         newInterpretations.clear();
15:     for ProcessInterpretation pi ∈ interpretations do
16:         computeFullBehavioralProfile(pi)
17:         if pi.getBehavioralProfile().isConsistent() then
18:             behavioralSpace.add(pi);
19:     return behavioralSpace;
```

an empty list (line 3), are incrementally extended with a single statement interpretation (lines 8–11). This ensures that each possible combination of statement interpretations is included in the list. For example, as considered in the previous section, the claims handling process contains three ambiguous statements with, respectively, two, three, and two possible interpretations. This results in a total number of 12 (2 × 3 × 2) possible combinations of the statement interpretations and, thus, of 12 interpretations in a behavioral space $\mathcal{BS}_T$. After all combinations have been generated, we compute a full behavioral profile for each of the interpretations and add them to the behavioral space (lines 15–17).

To compute the complete behavioral profile for a process interpretation (line 16), we exploit the transitivity of the strict order and interleaving order relations [15]. In this way, we can obtain relations beyond those pair-wise relations that we extracted from a textual description. For example, if a text specifies that activity a_i is followed by a_j and a_j is followed by a_k, i.e. $a_i \rightsquigarrow a_j$ and $a_j \rightsquigarrow a_k$, then a_i is also followed by a_k, i.e. $a_i \rightsquigarrow a_k$. After constructing this profile, the algorithm checks the internal consistency of the behavioral profile with a technique defined in [15]. We only include mutually consistent process interpretations to the behavioral space (lines 17–18). For example, we exclude obviously incorrect interpretations in which one statement interpretation yields the relation $a_i \rightsquigarrow a_j$ and another the relation $a_i + a_j$.

Once the process interpretations have been obtained in this manner, the construction of the behavioral space is complete. In Section 5, we illustrate the usefulness of these spaces for compliance checking.

5. Compliance Checking using Behavioral Spaces

By capturing behavioral ambiguity in a structured manner, behavioral spaces allow us to reason about process compliance without the need to arbitrarily settle ambiguity. In this section, we demonstrate this by showing how to perform a compliance check of an execution trace versus a behavioral space generated from a textual process description. The goal of *compliance checking* is to determine whether the behavior captured in an execution trace is allowed by the behavioral specification of a business process [4]. The key difference between traditional compliance checking and compliance checking using behavioral spaces lies in the potential outcomes of a check. In traditional compliance checking, a trace is either *compliant* or it is *non-compliant* with a business process. Due to the behavioral ambiguity captured in behavioral spaces, a trace can be compliant or non-compliant, but also *potentially compliant* with a behavioral space. The latter outcome occurs for traces that comply with one or more process interpretations in a behavioral space, but not with all of them.

5.1. Process Interpretation Compliance

Compliance checking of a trace t against a behavioral space $\mathcal{BS}_T$ builds on the compliance checking of t against individual process interpretations of the behavioral space. This is similar to the compliance check of a trace and a behavioral profile, as obtained from a process model (see [17]). This check builds on a comparison of the behavioral profile of a trace BP_t to the behavioral profile relations of a process interpretation $P = (\mathcal{I}, BP)$.

The behavioral profile BP_t captures the strict order, exclusiveness, and interleaving order relations for the set of activities A_t in a trace t. Given an activity pair $(a_i, a_j) \in (A_t \times A_t)$, BP_t contains the strict order relation $a_i \rightsquigarrow_t a_j$ iff at least one occurrence of activity a_i precedes an occurrence of activity a_j in t, and no occurrence of a_j precedes an occurrence of a_i in t. BP_t contains the interleaving order relation $a_i \parallel a_j$ iff at least one occurrence of a_i precedes an occurrence of a_j in t, and at least one occurrence of a_j precedes an occurrence of a_i in t.

Given a behavioral profile of a trace BP_t and a process interpretation $P \in \mathcal{BS}_T$, we can determine if t is compliant with $P = (\mathcal{I}, BP)$ by checking if the relations in BP_t do not violate the behavioral relations in BP.

Specifically, t is compliant with P if all relations in BP_t are *subsumed* by the relations in BP. A behavioral profile relation $R \in \mathcal{R}$ is subsumed by relation $R' \in \mathcal{R}$ if the relations are equal, i.e. $R = R'$, or if R' restricts less behavior than R. Definition 4 formally defines the notion of subsumption according to [17].

Definition 4 (Subsumption Predicate)**.** *Given two behavioral relations* $R, R' \in \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \|\}$*, the* subsumption predicate $\mathcal{S}(R, R')$ *is satisfied, iff* $R \in \{\rightsquigarrow, \rightsquigarrow^{-1}\} \wedge R' = +$ *or* $R = R'$ *or* $R = \|$.

Based on the notion of subsumption, we define compliance between a trace and a process interpretation in Definition 5.

Definition 5 (Trace to Process Interpretation Compliance)**.** *Let t be an event trace with an activity set A_t and $P = (\mathcal{I}, BP)$ a process interpretation in the behavioral space $\mathcal{BS}_\mathcal{T}$ with an activity set A_T, such that $A_t \subseteq A_T$. Then, the compliance predicate $compl(t, P)$ is satisfied if for each activity pair $(x, y) \in (A_t \times A_t)$ the relation R_t in BP_t of the pair (x, y) is subsumed by the relation R_P of the pair (x, y) in BP, i.e. $\mathcal{S}(R_P, R_t)$.*

Next, we describe how to determine the compliance of a trace to a behavioral spaced based on compliance checks between a trace and the space's individual process interpretations.

5.2. Behavioral Space Compliance

To determine the level of compliance between a trace and a behavioral space, we consider the number of process interpretations with which a trace complies. In particular, we quantify the *support* of a behavioral space $\mathcal{BS}_T$ for a trace t as the ratio between the number of interpretations to which t is compliant and the total number of interpretations in $\mathcal{BS}_T$:

$$\text{supp}(t, \mathcal{BS}_T) = \frac{|\{P \in \mathcal{BS}_T \mid compl(t, P)\}|}{|\mathcal{BS}_T|} \tag{1}$$

The support metric quantifies the fraction of interpretations that allow for a trace to occur. A support value of 1.0 indicates that a trace is without any doubt compliant with the behavioral space, i.e. independent of the chosen interpretation. A support of 0.0 shows that there is no interpretation under which a trace complies with the behavioral space. Therefore, it can be said with certainty that the trace is non-compliant with $\mathcal{BS}_T$. Finally, any trace t with a support value $0.0 < \text{supp}(t, \mathcal{BS}_T) < 1.0$ is potentially compliant with $\mathcal{BS}_T$. This implies that there are certain interpretations of the textual description to which the trace complies. To illustrate the usefulness of the support metric, consider the following three partial execution traces of the running example:

- Trace $t_1 =< a_1, a_2, a_3, a_4, a_5 >$;
- Trace $t_2 =< a_1, a_3, a_2, a_4, a_5 >$;
- Trace $t_3 =< a_{11}, a_{14}, a_{12}, a_{13} >$.

The difference between the traces t_1 and t_2 is that, in t_1, activity a_2 occurs before a_3, whereas these are executed in reverse order in t_2, i.e. $a_2 \rightsquigarrow a_3 \in BP_{t_1}$ and $a_3 \rightsquigarrow a_2 \in BP_{t_2}$. Furthermore, recall that the behavioral relation between these two activities is given by the ambiguous behavioral statement s_2. Depending on the interpretation of s_2, there either exists a strict order or an interleaving order relation between a_2 and a_3, i.e. $R(a_2, a_3) = \{\rightsquigarrow, \|\}$. The relation $a_2 \rightsquigarrow_{t_1} a_3$ from t_1 is subsumed by both possible interpretations included in the behavioral space, since sub($\rightsquigarrow, \rightsquigarrow$) and sub($\rightsquigarrow, \|$) are both satisfied. Therefore, t_1 is compliant with all interpretations in the behavioral space and, thus, has a support value of 1.0. For trace t_2 we observe a different situation. While $a_3 \rightsquigarrow_{t_2} a_2$ in trace t_2 is subsumed by relation $a_2 \| a_3$, this relation is not subsumed by $a_2 \rightsquigarrow a_3$. Therefore, t_2 does not comply with half of the process interpretations in the behavioral space. This results in supp$(t_2, \mathcal{BS}_T) = 0.5$.

Aside from providing information on the process interpretations with which a trace complies, behavioral spaces allow us to obtain further diagnostic information from this compliance check. In particular, we can gain insights into the conditions under which a trace is compliant with a process description. For example, we can learn under which interpretations of the statement s_3, "*In the meantime, the financial department takes care of the payment*", trace t_3 is compliant. In t_3, the financial department pays the settlement amount (a_{14}) before the claims officer records the settlement information (a_{12}). This complies with one of the two interpretations of statement s_3 and, therefore, results in a support value of 0.5. Furthermore, we know that this trace is compliant, if and only if "*in the meantime*" means "*while the claims officer is performing its tasks*" and not "*while the claims officer is archiving the claim*". Such diagnostic information can be useful when interpreting the support values for a trace or when aiming to resolve the ambiguity contained in a textual description.

6. Pruning Behavioral Spaces based on Information Gain

This section demonstrates how the behavioral space of a textual process description can be reduced to provide more accurate compliance-checking results. We refer to this act as *pruning* a behavioral space. In particular, we present a technique that supports users in resolving behavioral ambiguity in an efficient manner. The technique identifies those ambiguous behavioral statements in a textual description that lead to the most uncertainty in compliance-checking results. To achieve this, we define a metric that captures the *information gain* that can be achieved by resolving the ambiguity in behavioral statements. This metric quantifies the amount of uncertainty in compliance-checking results caused by a particular ambiguous behavioral statement. Therefore, information gain illustrates how much compliance-checking uncertainty can be removed by resolving the ambiguity in a particular statement. It serves a similar purpose as the information-gain metric used in the context of decision trees to quantify reductions in *information entropy* (cf. [18, 19]). Before introducing the information-gain metric, we first consider how compliance-checking uncertainty can be reduced by resolving ambiguous statements and, thereby, pruning a behavioral space.

6.1. Compliance-Checking Uncertainty

In Section 5, we demonstrated that behavioral spaces allow for reasoning about compliance without the need to resolve behavioral ambiguity. This is achieved by introducing the notion of *potential compliance*, which captures situations where a trace is compliant to some process interpretations, but non-compliant to others. Such a classification provides valuable information, especially in the context of the diagnostic information that can be associated with it, i.e. for which statement interpretations a trace is compliant or non-compliant. Still, such cases represent a form of unclarity in the compliance-checking results, because it is not known if a trace is actually compliant or not. We will refer to this state as *compliance-checking uncertainty*.

The accuracy of compliance-checking results can be improved by reducing the level of compliance-checking uncertainty. This can be achieved by resolving the cause of ambiguity in ambiguous statements. For instance, by replacing an ambiguous type indicator such as “*and*” with either “*and, then*” or with “*and, meanwhile*”. In these cases, there are less potentially compliant traces and more traces for which it can be stated with certainty that they are compliant or not. Behavioral spaces represent a powerful tool to support users in this endeavor. First, behavioral spaces support the improvement of compliance checking accuracy by providing insights into the *causes* (i.e. the ambiguous statements) and the *effects* (i.e. the different interpretations) of behavioral ambiguity. For instance, the behavioral space shows us that statement s_2 from the running example is ambiguous. Therefore, it is clear that if a user decides to resolve this ambiguity by selecting the correct interpretation of s_2, we reduce the overall compliance-checking uncertainty. Second, behavioral spaces can support users even further by letting them focus their resolution efforts on the ambiguous statements that are the greatest causes of compliance-checking uncertainty. We achieve this with the information-gain metric that we introduce next.

6.2. Information Gain

We introduce *information gain* (IG) as a metric that describes how much compliance-checking uncertainty can be resolved by selecting a single interpretation for an ambiguous statement. A proper quantification for this gain is to consider for how many traces the interpretations of a single ambiguous statement disagree about their compliance. By resolving the ambiguity in statements of which the interpretations disagree about the largest number of traces, a maximum of compliance-checking uncertainty can be removed. We define IG for a set of statement interpretations Γ and a set of traces (i.e. a log) L in Equation 2.

$$IG(\Gamma, L) = |\bigcup_{\gamma \in \Gamma} C_L(\gamma) - \bigcap_{\gamma \in \Gamma} C_L(\gamma)| \tag{2}$$

In this equation, we use $C_L(\gamma)$ to refer to the set of traces from L that are compliant to the behavioral relations that comprise a statement interpretation γ. $IG(\Gamma, L)$ specifies the size of the set of traces that are compliant to at least one interpretation, but also non-compliant to at least one interpretation. This is computed by taking all traces that are allowed according to at least one interpretation in Γ, i.e. the union of all sets $C_L(\gamma)$ for $\gamma \in \Gamma$, minus those traces that are allowed by all interpretations in Γ, i.e. the intersection of these sets.

The metric IG can be applied in two different ways, depending on the availability of an event log. If an event log is not available, a log L_G can be generated that contains all traces that are potentially compliant to the behavioral space $\mathcal{BS}$. In this case, $IG(\Gamma, L_G)$ can be used to identify those phrases that lead to the biggest potential reduction in ambiguity. However, if an event log L_R related to the process is already available, the information-gain metric can be used to compute the information gain in the context of truly observed behavior. In this case, $IG(\Gamma, L_R)$ represents the gain in ambiguity specific to the event log L_R.

To illustrate the usage of IG, consider the statements s_2 and s_3 used throughout this paper (introduced in Table 2) and a log L with (partial) execution traces:

- $t_1 =< a_2, a_3, a_{12}, a_{13}, a_{14} >$
- $t_2 =< a_2, a_3, a_{14}, a_{12}, a_{13} >$
- $t_3 =< a_2, a_3, a_{14}, a_{12}, a_{13} >$
- $t_4 =< a_3, a_2, a_{14}, a_{13}, a_{12} >$

Recall that statement s_2 has two interpretations, with the following sets of behavioral relations: $\{a_2 \rightsquigarrow a_3\}$ and $\{a_2 \parallel a_3\}$. It can be easily observed that these statements disagree about any trace in which a_3 occurs before a_2, i.e. of which the behavioral profile contains $a_3 \rightsquigarrow_t a_2$. Trace t_4 is the only trace in L for which this is the case. Therefore, $IG(\Gamma_{s_2}, L) = 1$. To compute IG for statement s_3, it suffices to consider the most restrictive and most flexible interpretations of the statement. The most *restrictive* interpretation states that a_{14} can only be executed in parallel, i.e. possibly before, activity a_{13}. By contrast, the most *flexible* interpretation of s_3 specifies that a_{14} is in an interleaving order with a_{11}, a_{12}, and a_{13}. This means that from log L, only trace t_1 is compliant to the former interpretation, whereas all four traces are compliant to the latter interpretation of s_3. Therefore, three traces in L are in dispute by the interpretations in Γ_{s_3}, i.e. $IG(\Gamma_{s_3}, L) = 3$. From this, it can be concluded that the resolution of ambiguity in statement s_3 has a greater impact on the ambiguity in log L than the resolution of s_2.

By computing IG for all ambiguous statements in a behavioral space and resolving the statements with the highest information gain, users can efficiently reduce the level of compliance-checking uncertainty. As such, the behavioral space will be pruned by removing process interpretations that are not compliant with the resolved ambiguity. This greatly enhances the efficient usage of the notion of behavioral spaces for compliance checking.

7. Evaluation

In this section, we evaluate the usefulness of behavioral spaces for compliance checking in the context of ambiguous textual process descriptions. For this purpose, we conduct a two-stage evaluation. First, we assess the impact that the consideration of behavioral spaces has on compliance-checking results. In particular, we compare the compliance-checking results obtained by using behavioral spaces to two alternative ways of dealing with behavioral ambiguity. Second, we demonstrate the effectiveness of the proposed IG metric for the reduction of uncertainty in compliance-checking results.

In the remainder, Section 7.1 first introduces the test collection used in both parts of the evaluation. Then, Sections 7.2 and 7.3 respectively describe the evaluation of the compliance-checking results and of the IG metric. Finally, we discuss limitations of the evaluation results in Section 7.4.

7.1. Test Collection

To perform our evaluation, we reuse the collection of textual process descriptions from the text-to-model generation approach by Friedrich et al [9]. The collection contains 47 process descriptions from various industrial and scholarly sources. Table 4 gives an overview of the characteristics of the test collection.

Table 4: Overview of the test collection

ID	Source	Type	PD	S	L
1	HU Berlin	Academic	4	10.0	18.1
2	TU Berlin	Academic	2	34.0	21.2
3	QUT	Academic	8	6.1	18.3
4	TU Eindhoven	Academic	1	40.0	18.5
5	Vendor Tutorials	Industry	4	9.0	18.2
6	inubit AG	Industry	4	11.5	18.4
7	BPM Practitioners	Industry	1	7	9.7
8	BPMN Practice Handbook	Textbook	3	4.7	17.0
9	BPMN Guide	Textbook	6	7.0	20.8
10	Federal Network Agency	Public Sector	14	6.4	20.0
	Total		**47**	**9.2**	**17.2**

Legend: PD = Number of process descriptions per source, S = Average number of sentences, L = Average number of words per sentence

The data from Table 4 illustrate that the included process descriptions differ greatly in size. The average number of sentences ranges from 4.7 to 34.0. The longest process description contains a total of 40 sentences. Furthermore, the descriptions differ in the average length of the sentences. While the BPM Practitioners source contains process descriptions with rather short sentences (9.7 words), the process descriptions from the TU Berlin source contain relatively long sentences (21.2 words). Lastly, the process descriptions differ in terms of how explicitly and unambiguously they describe the process behavior. Among others, this results from the variety of authors that created the textual descriptions. Hence, we believe that the collection is well-suited for achieving a reasonably high external validity of the results.

7.2. Compliance Evaluation

To demonstrate the usefulness of behavioral spaces for compliance checking, we compare the compliance-checking results obtained by using behavioral spaces to two alternative ways of dealing with behavioral ambiguity. These two alternatives are: (i) imposing assumptions on the correct interpretation of behavioral statements, and (ii) ignoring ambiguous statements because they cannot be resolved. The goal of this part of the evaluation is to show that behavioral spaces provide a much more reasonable view on the process behavior allowed by a textual process description, when compared to the two alternatives. Section 7.2.1 describes the details of the setup used for this step. In Section 7.2.2, we present and discuss the results.

7.2.1. Setup

To conduct the evaluation, we implemented a prototype to generate behavioral spaces from textual process descriptions. To achieve this, we build on the state-of-the-art text-to-process model generation approach by Friedrich et al. [9]. In particular, our Java prototype uses a library that is part of the RefMod-Miner[3], which implements the process model generation approach in a stand-alone tool. We use the library to automatically identify activities and extract behavioral profile relations that exist between activities.

We compare the compliance-checking results obtained by using behavioral spaces to two alternative ways of dealing with behavioral ambiguity. The first alternative reflects the possibility to deal with behavioral ambiguity by imposing assumptions on the correct interpretation of a text, i.e. by selecting a single interpretation for each ambiguous behavioral statement. Second, it is possible to deal with behavioral ambiguity by ignoring all ambiguous statements and, thus, only focusing on the behavioral relations that can be extracted with certainty from a text. Given these alternatives to behavioral spaces, we generate three behavioral models (BMs) for each of the 47 textual process descriptions as follows:

1. **Fully interpreted behavioral profile ($\mathbf{BP^{full}}$):** This behavioral model reflects an approach that imposes assumptions on the correct interpretation of ambiguous statements. To obtain BP^{full}, we generate a process model by using the text-to-model generation approach from [9] and, subsequently, extracting a behavioral profile from this model;

[3] http://refmod-miner.dfki.de

2. **Minimally restricted behavioral profile ($\mathbf{BP^{min}}$):** This behavioral model reflects an approach in which ambiguous statements are fully ignored. The resulting behavioral profile only captures the behavioral relations that can be extracted with certainty from the textual process description. To obtain BP^{min}, we remove all behavioral profile relations from BP^{full} that were extracted from the analysis of ambiguous behavioral statements;

3. **Behavioral space ($\mathcal{BS}$):** The behavioral space generated for the textual description in accordance with the interpretation generation method described in Section 4.

We conduct our evaluation by comparing the sizes of the sets of traces that are (potentially) compliant with the three behavioral models, in accordance with the definitions provided in Section 5.[4] Using $C(BM)$ to refer to the collection of traces that are compliant or potentially compliant to a behavioral model BM, we quantify the size differences using the following two metrics:

$$R_1 = \frac{\mid C(\mathcal{BS}) \mid}{\mid C(BP^{min}) \mid} \qquad (3) \qquad\qquad R_2 = \frac{\mid C(BP^{full})) \mid}{\mid C(\mathcal{BS}) \mid} \qquad (4)$$

R_1 quantifies the ratio between the number of traces allowed by a behavioral space and a minimally restricted behavioral profile. This measure reflects how much behavior that certainly does not comply with t is allowed when ambiguous statements are ignored. R_2 quantifies the ratio between the number of traces allowed by a behavioral space and those allowed by a fully interpreted behavioral profile. This measure reflects how much behavior that is possibly compliant to t is marked as noncompliant by an approach that imposes assumptions on ambiguous statements.

7.2.2. Results

Table 5 summarizes the evaluation results. The first interesting thing to note is how common textual process descriptions with behavioral ambiguity are. In total, 32 of the 47 textual process descriptions (70%) contained one or more ambiguous phrases. The majority of these cases, 28 in total, included just phrases with type ambiguity. Four cases contain statements with scope ambiguity, 3 of which also contain behavioral statements with type ambiguity.

[4] For processes that contain loops, we only include traces with at most one repetition.

Table 5: Evaluation results

Collection	**P**	**S$_{\text{type}}$**	**S$_{\text{scope}}$**	**A**	**|PI|**	**R$_1$**	**R$_2$**
Only type ambiguity	28	64	0	19.6	11.0	100.0%	37.8%
With scope ambiguity	4	13	4	24.0	76.5	16.4%	0.5%
Total	32	77	4	20.2	19.1	89.5%	33.7%

Legend: P = number of processes, S_{type} = statements with type ambiguity, S_{scope} = statements with scope ambiguity, A = extracted activities per process (avg.), $|\mathcal{PI}|$ = interpretations per behavioral space.

For processes with just type ambiguity in their descriptions, there is a clear difference between the behavior allowed by fully interpreted behavioral profiles $C(BP^{full})$ and the behavior allowed by behavioral spaces $C(\mathcal{BS})$. As indicated by metric R_2, the fully interpreted behavioral profiles allow for only 37.8% of the behavior allowed by the behavioral space. For the remaining 62.2% of the traces, we *cannot* state with certainty that they do not comply with the process described in the text. This difference results from ordering restrictions that the text-to-model generation algorithm imposes on activities, even when these ordering restrictions may not exist. Behavioral spaces do not impose such restrictions and, thus, mark traces that exhibit such execution flexibility as potentially compliant. This consideration of the cases with type ambiguity already illustrates the impact of assumptions on compliance checking. Nevertheless, this impact is much more severe for textual process descriptions that also contain statements with scope ambiguity.

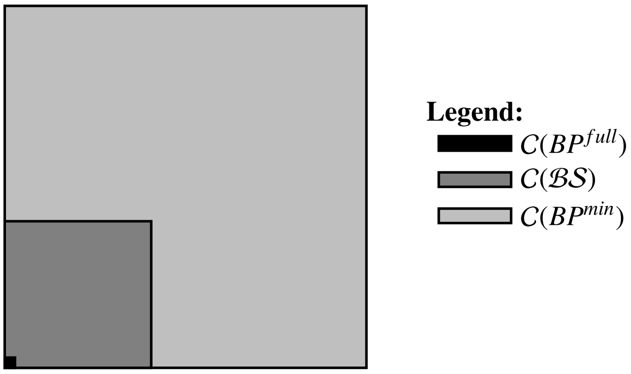


Figure 4: Visualization of three sets of compliant traces for cases with scope ambiguity.

The behavioral models for the 4 cases with scope ambiguity show much larger differences among the behavior they allow. We visualize the relative sizes of the three sets of compliant traces in Figure 4. The light-gray area denotes the set of traces compliant with BP^{min}, i.e. the set of traces that remain when treating ambiguous statements as undecidable. The behavior allowed by the behavioral space, represented by the dark-gray area, is considerably smaller, as also indicated by the R_1 score of 16.4%. This number reveals that 83.6% of the traces in $C(BP^{min})$ are not compliant with any reasonable interpretation of the statements with scope ambiguity. Figure 4 also shows the considerable impact that the usage of single interpretations has on the number of compliant traces. The tiny size of the black area in the figure and the R_2 score of 0.5% both indicate that, for the cases with scope ambiguity, the fully interpreted behavior profiles allow for only a very small fraction of the behavior that is (potentially) compliant to a behavioral space. Again, the remaining 99.5% represent traces that do not necessarily conflict with behavior specified in a textual process description.

The evaluation results show the impact both of ignoring ambiguous statements and of imposing single interpretations on them. As visualized by Figure 4, behavioral spaces provide a balance between these loosely restricted and too restricted behavioral models. In summary, behavioral spaces exclude a large number of nonsensical traces, which can be excluded by generating proper interpretations for ambiguous statements. Still, they allow for much more traces than the restricted models that are obtained by imposing assumptions on the ambiguous statements in textual descriptions.

7.3. Pruning Evaluation

In the second part of the evaluation, we set out to demonstrate how effective the proposed pruning technique is at reducing uncertainty in compliance-checking results. Specifically, we assess how quickly compliance uncertainty can be reduced when we employ the *IG* metric, introduced in Section 6, to select ambiguous phrases. As a benchmark, we compare the results obtained in this manner to a random selection mechanism. Section 7.3.1 describes the setup of this part of the evaluation, followed by a presentation of the results in Section 7.3.2.

7.3.1. Setup

To evaluate the effectiveness of the *IG* metric, we make use of the behavioral spaces generated for the textual descriptions in the previous part of the evaluation. Specifically, we select the behavioral spaces for the 17 textual descriptions with more than one ambiguous behavioral statement. For these cases it is relevant to determine which ambiguous phrase should be resolved first. To compute values for the IG metric, we generate a log L_t for each textual description T that contains all traces that are potentially compliant to the behavioral space $\mathcal{BS}$.

In this evaluation, we are interested in how much we can reduce compliance-checking uncertainty by using IG to select the ambiguous phrases we resolve first. We quantify this reduction by comparing the number of *potentially compliant* traces that remain after resolving the ambiguity in a statement to the original number of potentially compliant traces. Again, we use $C(\mathcal{BS})$ to denote the set of potentially compliant traces for a given behavioral space $\mathcal{BS}$. Then, we compute the fraction of compliance uncertainty that remains after resolving k ambiguous statements as given by Equation 5. Here, $\mathcal{BS}_k$ represents the behavioral space that remains after resolving k ambiguous statements. $\mathcal{BS}_0$ represents the behavioral space for which no ambiguity is resolved.

$$U(\mathcal{BS}, k) = \frac{|C(\mathcal{BS}_k)|}{|C(\mathcal{BS}_0)|} \tag{5}$$

We compute the value $U(\mathcal{BS}, k)$ for each $k \in 1, \ldots, n$, where n is the number of ambiguous statements in $\mathcal{BS}$. As a benchmark, we compare these values against the uncertainty that remains after randomly selecting ambiguous statements to resolve. In particular, we compute the average value of $U(\mathcal{BS}, k)$ for each of the $n!$ different orders in which ambiguous statements can potentially be selected. As such, this benchmark mimics the situation in which people blindly select ambiguous statements to resolve.

7.3.2. Results

Figure 5 visualizes the evaluation results. The curves represent the average reduction in uncertainty over the 17 relevant cases. The figure shows considerable differences between the reduction obtained by using the information gain metric versus the reductions obtained through random selection.

When interpreting these results, it is important to recognize that the minimum uncertainty reduction per statement observed in the test collection is 50.0%. This quantity represents the amount of uncertainty that is removed when resolving the most simple form of an ambiguous statement: a statement with type ambiguity between just two activities. Therefore, the reduction of 63.7% that is obtained by randomly resolving a single ambiguous statement appears is only 13.7% higher than the minimum improvement per statement. In

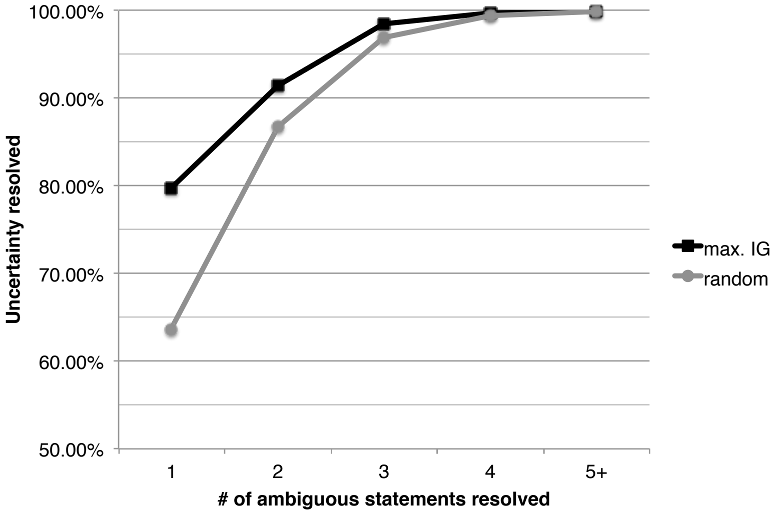


Figure 5: Visualization of three sets of compliant traces for cases with scope ambiguity.

comparison, usage of the IG metric results in a removal of 79.7% of the compliance-checking uncertainty in the first step. This reflects an improvement of 29.7 percentage points above the minimum improvement per statement. Therefore, the information gain metric leads to an improvement that is more than twice as high as random selection.

The 79.7% reduction in uncertainty demonstrates that, in many textual descriptions, a single statement has a much bigger impact on the ambiguity than the others. These statements are typically the ones with scope ambiguity. This is the case because scope ambiguity results in a high number of potentially compliant traces, usually caused by interpretations with a considerable number of activities with interleaving order relations. This is, for instance, shown for statement s_3 considered in the running example of this paper. Similarly, statements with type ambiguity among more than two activities result in an increased number of interleaving order relations compared to statements with ambiguity between just two activities. Therefore, these statements result in a high number of potentially compliant traces. By resolving these ambiguous statements first, the information gain metric can be used to quickly resolve the vast majority of compliance-checking uncertainty.

7.4. Limitations

Our evaluation demonstrates the usefulness of behavioral spaces for reasoning about process compliance in the context of textual process descriptions. However, the evaluation results should be considered against the background of some limitations. In particular, we are able to identify limitations related to the behavioral space generation approach, to the compliance-checking technique, and limitations related to the evaluation.

7.4.1. Generation Approach

Limitations related to the *generation approach* concern two facets. First, it has to be considered that the natural language processing techniques on which we build our approach are not fully accurate. The model generation approach from [9] is heuristics-based, which means it does not cover all possible linguistic patterns that can be used to express behavioral relations. For instance, the approach cannot handle constructs corresponding to OR-gateways in process models, such as "*At least two of the following three activities should be performed*". However, since this approach represents the state of the art, it does provide an accurate reflection of the quality of model generation approaches. Furthermore, it is important to stress that our generation approach is largely independent of the underlying model generation approach. As long as this approach can provide information regarding source, relation type, and target references, behavioral spaces can be generated according to the algorithms detailed in Section 4.1.

Second, it has to be taken into account that we generate interpretations for statements with scope ambiguity based on three commonalities. While these commonalities represent the plausible options in the context of process descriptions, it is possible that, in certain situations, other properties are necessary to capture the set of activities to which a statement refers. Still, the behavioral space will identify such statements as ambiguous and help users to analyze the impact of this ambiguity on compliance checks.

7.4.2. Compliance Checking

A point of consideration regarding our compliance-checking technique is that we perform compliance checks based on behavioral profiles. The expressive power of these relations has been shown to be less than that of process modeling notations such as Petri nets, which are used by certain other compliance-checking techniques [4]. Because of this lack of expressive power, behavioral profiles abstract from certain process behavior. As a result, compliance checking based on behavioral profiles is less restrictive than compliance checking techniques based on Petri nets. [20] provides a detailed overview of the restrictions on expressiveness. Nevertheless, we have selected behavioral profiles because they enable highly efficient compliance checks, which is an important prerequisite given the numerous possible process interpretations that can exist in behavioral spaces. Furthermore, by using behavioral predicates to express behavioral relations, we are able to combine the implications of different statement interpretations in an intuitive and safe manner. Lastly, it is important to stress that the notion of a behavioral space as a basis for compliance checking is independent of the underlying compliance measures. As long as behavioral spaces are used to capture the various interpretations of ambiguous process descriptions, the same reasoning can be directly transferred to other notions of process compliance.

7.4.3. Evaluation

As for limitations related to the evaluation, we would like to point out that the presented quantitative results are bound to the specifics of the textual process descriptions included in the test collection. The employed data collection does not form a statistically representative sample. In fact, the creation of such a sample is hardly feasible since natural language offers such a high degree of freedom. However, the data set is composed of several heterogeneous sources and contains a considerable degree of ambiguity. Furthermore, this data set has been previously used to evaluate the process model generation approach of Friedrich et al. [9]. Therefore, we are confident that our evaluation shows a realistic picture of the impact of behavioral ambiguity in practical settings.

8. Related Work

The work presented in this paper primarily relates to three major research streams: compliance checking, the analysis of textual process descriptions, and the representation of data uncertainty. In the remainder of this section, we describe each of these related areas.

8.1. Compliance Checking

Process compliance checking or conformance checking involves the comparison of different behavioral specifications. These techniques are applied in various application scenarios, including process querying [6], legal compliance [21], and auditing [2]. Most compliance checking techniques focus on the comparison of observed process behavior, as captured in *execution traces*, to a process specification. A plethora of techniques exist for this purpose (cf. [4, 22, 23, 24]). In this paper, we used techniques that perform compliance checks against behavioral profiles, introduced in [5]. The advantage of these techniques is their high computational efficiency. Therefore, they represent an attractive choice for compliance checking against behavioral spaces, which can contain a vast number of different behavioral interpretations. However, other commonly used techniques compare execution traces to process models based on so-called *alignments*. These techniques (cf. [4, 23]) provide different diagnostic information than compliance checks based on behavioral profiles. Furthermore, the compliance checks can be considered to be more accurate in some situations, because behavioral profile relations abstract from certain details of process behavior. It is important to emphasize at this point that the notion of a behavioral space, which consists of different process interpretations, can be easily translated to other compliance checking notions, such as alignment-based checks.

The aforementioned compliance checking techniques all rely on a *structured* specification of the behavior allowed for a process, e.g. in the form of process models. In earlier work, we introduced the first compliance checking approach that works with *textual* process descriptions [25], a less structured representation format. The current manuscript extends the earlier conference paper in three ways. First, we provided detailed descriptions of the algorithms and dictionaries used to automatically generate behavioral spaces from textual process descriptions. Second, we introduced a method to reduce the level of uncertainty in compliance checking results by supporting users in the efficient resolution of ambiguous parts of a textual description, a so-called *pruning* method. Lastly, we have extended the evaluation to demonstrate the usefulness of our pruning method.

8.2. Analysis of Textual Process Descriptions

The majority of works that consider the analysis of textual artifacts related to business processes focus on the automated derivation of process models from them. Respective techniques have been designed for textual process descriptions [9, 26], group stories [13], use case descriptions [14], and textual methodologies [27]. From these, the text-to-model generation technique from Friedrich et al. [9] is typically recognized as the state of the art [28]. Therefore, we used it as a basis for our own prototype and as a benchmark for our evaluation. Though none of these existing works mentions the problem of behavioral ambiguity explicitly, all techniques impose assumptions on the interpretation of ambiguous behavioral statements. This results in a single interpretation, i.e. a process model, for any given text. However, this comes with the great disadvantage that the behavior allowed by this representation is much stricter than the behavior specified in the textual description. Our earlier works on the comparison of textual process descriptions to process models [29, 30] face similar issues when reasoning about the consistency of the two artifacts.

8.3. Representing Data Uncertainty

Similar to behavioral ambiguity in textual process descriptions, ambiguous or uncertain data is also present in other application contexts. In these cases, uncertainty can be caused by, among others, data randomness, incompleteness, and limitations of measuring equipment [31]. This has created a need for algorithms and applications for uncertain data management [32]. As a result, the modeling of uncertain data has been studied extensively (cf. [33, 34, 35, 36]). Our notion of a behavioral space builds on concepts related to those used in uncertain data models. For instance, similar to the behavioral interpretations captured in a behavioral space, the model presented by Das Sarma et al. [36] uses a set of *possible instances* to represent the spectrum of possible interpretations for an uncertain relation. Furthermore, the model described in [33] uses conditions to capture dependencies between uncertain values. This notion has the same result as the sets of behavioral relations that we derive from uncertain behavioral statements and convert into different behavioral interpretations. Still, the technical aspects and application contexts of these uncertain data models, which mostly relate to querying and data integration [32], differ considerably from the process-oriented view of behavioral spaces.

9. Conclusions

In this paper, we introduced the concept of a behavioral space to deal with the ambiguity in textual process descriptions. A behavioral space captures all possible interpretations of a textual process description. In this way, it avoids the issue of focusing on a single process-oriented interpretation of a text. We demonstrated that a behavioral space is a useful concept for reasoning about a process described by a text. In particular, we used a quantitative evaluation with a set of 47 textual process descriptions to illustrate that a behavioral space strikes a reasonable balance between ignoring ambiguous statements and imposing fixed interpretations on them. Furthermore, we demonstrated the usefulness of a semi-automated pruning technique to quickly reduce the level of uncertainty remaining in compliance-checking results.

While we defined the behavioral space concept based on textual process descriptions, we would like to point out that its use is not limited to pure text. A behavioral space can also help to capture the full behavior of other types of ambiguous process representations. Consider, for instance, process models containing activities that describe several streams of actions by using ambiguous behavioral statements such as "*and*". It has been found that such *non-atomic* activities can result in different interpretations of how to properly execute the process [37]. A behavioral space would also be useful for application scenarios beyond compliance checking. In future work, we set out to explore these usage scenarios of behavioral spaces in more detail.

Bibliography

[1] S. English, S. Hammond, The rising costs of non-compliance: from the end of a career to the end of a firm, Thomson Reuters, 2014.

[2] R. Accorsi, T. Stocker, On the exploitation of process mining for security audits: the conformance checking case, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, ACM, 2012, pp. 1709–1716.

[3] W. M. van Aalst, K. M. van Hee, J. M. van Werf, M. Verdonk, Auditing 2.0: using process mining to support tomorrow's auditor, Computer 43 (3) (2010) 90–93.

[4] W. Van der Aalst, A. Adriansyah, B. van Dongen, Replaying history on process models for conformance checking and performance analysis, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2 (2) (2012) 182–192.

[5] M. Weidlich, J. Mendling, M. Weske, Efficient consistency measurement based on behavioral profiles of process models, IEEE Transactions on Software Engineering 37 (3) (2011) 410–429.

[6] A. Awad, G. Decker, M. Weske, Efficient compliance checking using bpmn-q and temporal logic, in: International Conference on Business Process Management, Springer, 2008, pp. 326–341.

[7] F. Chesani, P. Mello, M. Montali, F. Riguzzi, M. Sebastianis, S. Storari, Checking compliance of execution traces to business rules, in: International Conference on Business Process Management, Springer, 2008, pp. 134–145.

[8] H. Van der Aa, H. Leopold, F. Mannhardt, H. A. Reijers, On the fragmentation of process information: Challenges, solutions, and outlook, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2015, pp. 3–18.

[9] F. Friedrich, J. Mendling, F. Puhlmann, Process model generation from natural language text, in: Advanced Information Systems Engineering, Springer, 2011, pp. 482–496.

[10] H. Leopold, J. Mendling, A. Polyvyanyy, Supporting process model validation through natural language generation, IEEE Transactions on Software Engineering 40 (8) (2014) 818–840.

[11] M. Selway, G. Grossmann, W. Mayer, M. Stumptner, Formalising natural language specifications using a cognitive linguistic/configuration based approach, Information Systems 54 (2015) 191–208.

[12] H. Leopold, H. van der Aa, F. Pittke, M. Raffel, J. Mendling, H. A. Reijers, Integrating textual and model-based process descriptions for comprehensive process search, in: International Workshop on Business Process Modeling, Development and Support, Springer International Publishing, 2016, pp. 51–65.

[13] J. C. de Gonçalves, F. M. Santoro, F. A. Baiao, Business process mining from group stories, in: Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on, IEEE, 2009, pp. 161–166.

[14] A. Sinha, A. Paradkar, Use cases to process specifications in Business Process Modeling Notation, in: IEEE International Conference on Web Services, 2010, pp. 473–480.

[15] S. Smirnov, M. Weidlich, J. Mendling, Business process model abstraction based on behavioral profiles, in: Service-Oriented Computing, Springer, 2010, pp. 1–16.

[16] D. Klein, C. D. Manning, Accurate unlexicalized parsing, in: Proceedings of the 41st Annual Meeting of the ACL-Volume 1, ACL, 2003, pp. 423–430.

[17] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, M. Weske, Process compliance analysis based on behavioural profiles, Information Systems 36 (7) (2011) 1009–1025.

[18] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1) (1986) 81–106.

[19] S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Systems, Man, & Cybernetics.

[20] A. Polyvyanyy, A. Armas-Cervantes, M. Dumas, L. García-Bañuelos, On the expressive power of behavioral profiles, Formal Aspects of Computing 28 (4) (2016) 597–613.

[21] S. Sadiq, G. Governatori, K. Namiri, Modeling control objectives for business process compliance, in: International conference on business process management, Springer, 2007, pp. 149–164.

[22] E. Ramezani, D. Fahland, W. M. P. van der Aalst, Where did i misbehave? diagnostic information in compliance checking, in: International conference on business process management, Springer, 2012, pp. 262–278.

[23] A. Adriansyah, B. van Dongen, W. van der Aalst, Conformance checking using cost-based fitness analysis, in: Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, IEEE, 2011, pp. 55–64.

[24] J. Munoz-Gama, J. Carmona, W. M. P. Van Der Aalst, Single-entry single-exit decomposed conformance checking, Information Systems 46 (2014) 102–122.

[25] H. van der Aa, H. Leopold, H. A. Reijers, Dealing with behavioral ambiguity in textual process descriptions, in: International Conference on Business Process Management, Springer International Publishing, 2016, pp. 271–288.

[26] A. Ghose, G. Koliadis, A. Chueng, Process discovery from model and text artefacts, in: Services, 2007 IEEE Congress on, IEEE, 2007, pp. 167–174.

[27] E. Viorica Epure, P. Martin-Rodilla, C. Hug, R. Deneckere, C. Salinesi, Automatic process model discovery from textual methodologies, in: Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on, IEEE, 2015, pp. 19–30.

[28] M. Riefer, S. F. Ternis, T. Thaler, Mining process models from natural language text: A state-of-the-art analysis, in: Multikonferenz Wirtschaftsinformatik (MKWI-16), March 9-11, Illmenau, Germany, Universität Illmenau, 2016.

[29] H. Van der Aa, H. Leopold, H. A. Reijers, Detecting inconsistencies between process models and textual descriptions, in: Business Process Management, Springer, 2015, pp. 90–105.

[30] H. van der Aa, H. Leopold, H. A. Reijers, Comparing textual descriptions to process models – the automatic detection of inconsistencies, Information Systems, 2016 (in press).

[31] J. Pei, B. Jiang, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data, in: Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 15–26.

[32] C. C. Aggarwal, P. S. Yu, A survey of uncertain data algorithms and applications, Knowledge and Data Engineering, IEEE Transactions on 21 (5) (2009) 609–623.

[33] S. Abiteboul, P. Kanellakis, G. Grahne, On the representation and querying of sets of possible worlds, Vol. 16, ACM, 1987.

[34] T. Imieliński, W. Lipski Jr, Incomplete information in relational databases, Journal of the ACM (JACM) 31 (4) (1984) 761–791.

[35] L. Peng, Y. Diao, Supporting data uncertainty in array databases, in: ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 545–560.

[36] A. D. Sarma, O. Benjelloun, A. Halevy, J. Widom, Working models for uncertain data, in: 22nd International Conference on Data Engineering, IEEE, 2006, pp. 7–7.

[37] F. Pittke, H. Leopold, J. Mendling, When language meets language: Anti patterns resulting from mixing natural and modeling language, in: Business Process Management Workshops, Springer, 2014, pp. 118–129.