Softw Syst Model (2015) 14:1105–1125
DOI 10.1007/s10270-013-0389-6

SPECIAL SECTION PAPER

Querying process models by behavior inclusion

Matthias Kunze · Matthias Weidlich · Mathias Weske

Received: 23 September 2012 / Revised: 27 May 2013 / Accepted: 11 October 2013 / Published online: 3 December 2013

Abstract Business processes are vital to managing organizations as they sustain a company's competitiveness. Consequently, these organizations maintain collections of hundreds or thousands of process models for streamlining working procedures and facilitating process implementation. Yet, the management of large process model collections requires effective searching capabilities. Recent research focused on similarity search of process models, but querying process models is still a largely open topic. This article presents an approach to querying process models that takes a process example as input and discovers all models that allow replaying the behavior of the query. To this end, we provide a notion of behavioral inclusion that is based on trace semantics and abstraction. Additional to deciding a match, a closeness score is provided that describes how well the behavior of the query is represented in the model and can be used for ranking. The article introduces the formal foundations of the approach and shows how they are applied to querying large process model collections. An experimental evaluation has been conducted that confirms the suitability of the solution as well as its applicability and scalability in practice.

Keywords Process model search · Behavioral querying · Trace inclusion · Process model repositories

Communicated by Dr. Selmin Nurcan.

M. Kunze · M. Weske
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
e-mail: matthias.kunze@hpi.uni-potsdam.de

M. Weske
e-mail: mathias.weske@hpi.uni-potsdam.de

M. Weidlich (✉)
Technion—Israel Institute of Technology, Haifa, Israel
e-mail: weidlich@tx.technion.ac.il

1 Introduction

Business process models explicitly capture the knowledge to carry out operations and services of an organization. Hence, process models are essential knowledge assets and are stored and maintained in process model repositories for reference, e.g., for documentation, automation, and certification, and to enable reuse, which leads to more efficient and consistent process design [2]. As business process management has gained influence toward sustaining a company's competitiveness, many organizations maintain hundreds or thousands of business process models [2,60]. Such collections comprise, for instance, reference processes. Examples provided in [24] include the SAP reference model collection (∼600 models), a collection of reference models for Dutch municipalities (∼600 models), and a collection of business services toward insurance management provided by IBM (∼250 models). An Australian insurance company reportedly maintains about 6,000 process models [24] and a Chinese railway car manufacturer even several hundred thousand [65].

Effective use of this knowledge asset requires, in particular, powerful capabilities to access the information and search for process models. Recent work has largely addressed similarity search, which generally addresses duplicate detection [25], e.g., to find similar process models in the model collections of two companies after a merger or acquisition. The general property of these similarity measures is a comparison of the number of shared features. If two models share only few features relative to their size, they will exhibit little similarity, even if one model is contained in another, significantly larger one. Hence, similarity search is not well suited to find process models of which only few yet important aspects are considered a priori. Process model querying addresses such cases where a query is formulated by a user to discover process models that comply with this query [52]. It is use-

ful to avoid the creation of duplicates, as it allows searching for processes before modeling. By querying, existing process logic can be discovered, reused, and revised, which shortens the design time and increases the quality of newly created process models.

Typically, querying requires a specific language to express certain requirements of the query and map them to features exposed by candidate models. In this article, we propose a novel querying approach that takes a regular process model as input and discovers all process models that comply with the example in their behavior. It has been inspired by Zloof, who proposed querying by example for relational databases, where *'the user basically formulates his query […] with an example of a possible answer'* [79]. Quite similar, a querying approach is presented, where a process model that consists only of few activities and control flow relations between them acts as an example and any possible answer must include the example's behavior.

Our contribution is twofold. First, we introduce a notion for behavioral inclusion between query and candidate process models, called abstract trace inclusion, and a measure for ranking matching models. A trace is a sequence of activities that can be executed in a certain order by a business process. Informally, abstract trace inclusion requires that for every trace of the query, a matching model must be able to produce a trace that contains all activities of the query trace in the same order. In other words, a matching model must be able to replay the behavior of the query. To rank matching models, we propose an asymmetric closeness measure that quantifies the amount of abstraction required to achieve abstract trace inclusion. Abstract trace inclusion and closeness can be decided for process models with a finite state space. Our second contribution is the characterization of abstract trace inclusion and closeness based on successor relations. For process models given as sound free-choice workflow-systems, deciding inclusion and computation of closeness can be done efficiently. Successor relations also form the basis for an index data structure that enables fast and scalable process model querying.

This article builds upon the ideas of our previous work [44], which introduced behavior inclusion based on behavioral relations. Here, we take a different approach by first defining behavior inclusion and closeness for trace semantics, whereas behavioral relations are only used in a second step to achieve efficient reasoning. Based on this, we illustrate the design of an efficient querying system and evaluate it toward its effectiveness, i.e., how well it aligns with human assessment of matching, and its efficiency, i.e., search performance and scalability in a practical setting.

The article is structured as follows. The next section briefly introduces the background of process model querying by a motivating example. Formal preliminaries, namely net systems, successor relations, and process alignments, are given in Sect. 3, before we present the concepts to match and rank candidate process models against a query model in Sect. 4. We illustrate how these concepts are applied for process model querying and how they can be used for efficient querying of process models in Sect. 5. The organization and results of our evaluation are presented in Sect. 6. We discuss related work in Sect. 7, before Sect. 8 concludes the article.

2 Motivating example

We illustrate the topic addressed in this article and the difference between behavioral querying and similarity with an example that provides the basis for later discussions. Figure 1 shows various business processes modeled in BPMN that handle customer requests regarding the mending of previously sold devices, due to service notifications or customer complaints. All processes include the activity *manage cost* that covers actions with regard to evaluating costs and accounting the correct cost center. Spare parts must be *ordered* and *delivered* to the customer, and the warehouse *stock* must be updated. Also, an *invoice is sent* to the customer for the provided service. Please note that these processes may be initialized by different triggers, i.e., start events, and that the process in Fig. 1c either manages the cost or sends an invoice, depending on an earlier decision whether the device is under warranty or not. We refer to the events and activities by their annotation, i.e., A–F for activities and X and Y for start events, hereafter.

Process model querying is a fundamental function of business process repositories and targets at the discovery of process models. In the following, the term *query* denotes the specification of the search question. We refer to models from a repository, against which the query is evaluated, as a *candidate*. If the candidate satisfies the requirements of a query, we call it a *match*. Querying essentially targets at discovering knowledge for reuse. For instance, before modeling a business process in the first place, a user may want to search for existing process models that contain the required behavior. In that case, the user could extract the fragment of a match and reuse it in their own model. Alternatively, the user may search for completions for a process stub [41,53]. In such cases, queries would consist of only few elements that are requested, i.e., a set of activities and behavioral relations between them.

Example queries that outline the expressiveness of our approach are depicted in Fig. 2. Figure 2a presents a purely sequential query, which expresses that all three activities shall be carried out by a matching process in the same order as in the query. This is obviously in line with process models in Fig. 1b, c. The process in Fig. 1a matches this query too, because the parallel gateway allows for the interleaving exe-

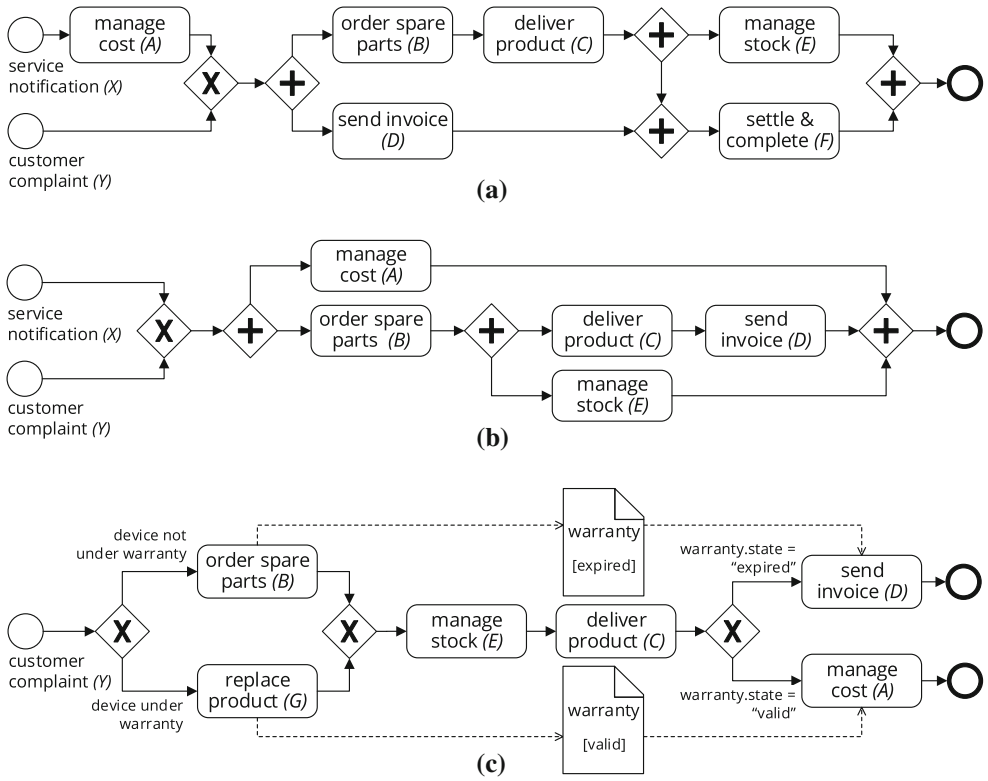


Fig. 1 Example process models

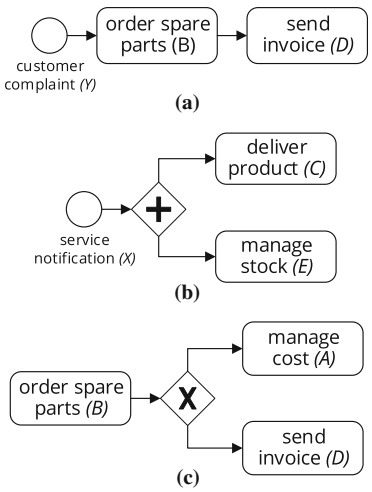


Fig. 2 Example queries

cution of actions B and D, which includes that B can be executed before D. This coincides with the requirement of the query although they are structurally distinct from each other, i.e., the sequence flow edge between B and D of the query has no counterpart in the match. The query in Fig. 2b requests a process that allows delivering the product and manage the stock in parallel or interleaving order, i.e., their actual order should not be restricted by a matching process, and Fig. 2c requests models that provide a choice between cost management and sending an invoice after ordering spare parts. We will discuss these examples in more detail, in Sect. 4.

A comprehensive body of work addresses the computation of similarity between pairs of process models, both by structural and behavioral aspects [7]. However, similarity search of process models [25] is not a promising solution to discover models in our setting, as similarity evaluates how much two process models resemble each other. While either of the example queries has at least one precise match among the candidates, queries and candidates are not similar at all, in particular, since the queries are much smaller than the candidates. Instead, querying aims at retrieving all models that precisely match a given query, even if the match is a significant extension of the query's features [44].

3 Preliminaries

This section defines preliminaries for our work. We first discuss notions and notations for net systems. Then, we turn to alignments between net systems and introduce different successor relations.

3.1 Net systems

We first clarify notations for sequences. A *sequence* over a set S of length $n \in \mathbb{N}$ is a function $\sigma : \{1, \ldots, n\} \to S$. If $\sigma(i) = s_i$ for $i \in \{1, \ldots, n\}$, we write $\sigma = \langle s_1, \ldots, s_n \rangle$ with $\langle\rangle$ as the *empty sequence*. The length of σ is denoted by $|\sigma|$. The set of all finite sequences over S is denoted by S^*. *Concatenation* of a sequence $\nu \in S^*$ and $s \in S$, denoted by $\sigma = \nu; s$ is defined as $\sigma : \{1, \ldots, |\nu| + 1\} \to S$, such that for $1 \leq i \leq |\nu|$: $\sigma(i) = \nu(i)$ and $\sigma(|\nu| + 1) = s$.

Process models may be captured as workflow (WF-) systems [69], a dedicated class of Petri nets. For many common process description languages, such as the business process model and notation (BPMN), event driven process chains (EPC), and the business process execution language (BPEL), net-based formalizations have been presented, see [49] for an overview. Based on [59], we recall basic notions of nets and net systems.

Definition 1 (*Net*) A *net* is a tuple $N = (P, T, F)$ with P and T as finite disjoint sets of places and transitions, and a flow relation $F \subseteq (P \times T) \cup (T \times P)$.

We write $X = P \cup T$ for all nodes and denote the irreflexive transitive closure of F by F^+. For $x \in X$, $\bullet x := \{y \in X \mid (y, x) \in F\}$ is the pre-set, $x\bullet := \{y \in X \mid (x, y) \in F\}$ is the post-set, and $\bullet(x\bullet) := \{z \in X \mid y \in X \wedge (x, y) \in F \wedge (z, y) \in F\}$. A net N is *free-choice*, iff $\forall\ p \in P$ with $|p\bullet| > 1$ holds $\bullet(p\bullet) = \{p\}$ [19]. As we will see later, free-choice nets are characterized by a close relation between their structure and their semantics. WF-nets have been defined in [69] as follows.

Definition 2 (*WF-net*) A net $N = (P, T, F)$ is a *WF-net*, iff N has an *initial* place $i \in P$ with $\bullet i = \emptyset$, a *final* place $o \in P$ with $o\bullet = \emptyset$, and the *short-circuit* net $N' = (P, T \cup \{t_c\}, F \cup \{(o, t_c), (t_c, i)\})$, $t_c \notin T$, of N is strongly connected.

We define semantics of nets as follows. Let $N = (P, T, F)$ be a net. $M : P \mapsto \mathbb{N}$ is a *marking* of N, $\mathbb{M}$ denotes all markings of N. $M(p)$ returns the number of *tokens* in place p. $[p]$ denotes the marking where place p contains just one token and all other places contain no tokens. For any transition $t \in T$ and any marking $M \in \mathbb{M}$, t is *enabled* in M, denoted by $(N, M)[t\rangle$, iff $\forall\ p \in \bullet t\ [M(p) \geq 1]$. Marking M' is reachable from M by *firing* of t, denoted by $(N, M)[t\rangle(N, M')$ such that $M' = M - \bullet t + t\bullet$, i.e., one token is taken from each input place of t and one token is added to each output place of t.

A sequence of transitions $\sigma = \langle t_1, \ldots, t_n \rangle$, $n \in \mathbb{N}$, is a *firing sequence*, if and only if there exist markings $M_0, \ldots, M_n \in \mathbb{M}$, such that for all $i \in \mathbb{N}$, $1 \leq i \leq n$ it holds $(N, M_{i-1})[t_i\rangle(N, M_i)$. We say that σ is enabled in M_0, denoted by $(N, M_0)[\sigma\rangle$. For any two markings $M, M' \in \mathbb{M}$, M' is reachable from M in N, denoted by $M' \in [N, M\rangle$, if there exists a firing sequence σ leading from M to M'. Firing of σ in M is denoted by $(N, M)[\sigma\rangle(N, M')$.

A *net system*, or *system*, is a pair $S = (N, M_i)$, where N is a net and M_i is the *initial marking* of N. A *WF-system* is a pair $S = (N, M_i)$, where N is a WF-net with initial place i and $M_i = [i]$. System S is free-choice, iff N is free-choice.

The exemplary business process models and queries captured in BPMN in Figs. 1 and 2 can be mapped to net systems following the formalization approaches presented by Dijkman et al. [23] for the control flow and by Awad et al. [3] for the data perspective. The latter is needed for representing the data artifacts and their influence on the control flow routing for the model in Fig. 1c. Figures 3 and 4 show the net systems obtained for our examples. The initial marking of a net is derived from the entry points of the business process model [18].

Formalizations of business process models that employ basic control flow routing, such as AND-gateways and XOR-gateways, can be mapped to free-choice net systems. As discussed in [49], however, advanced control flow concepts such

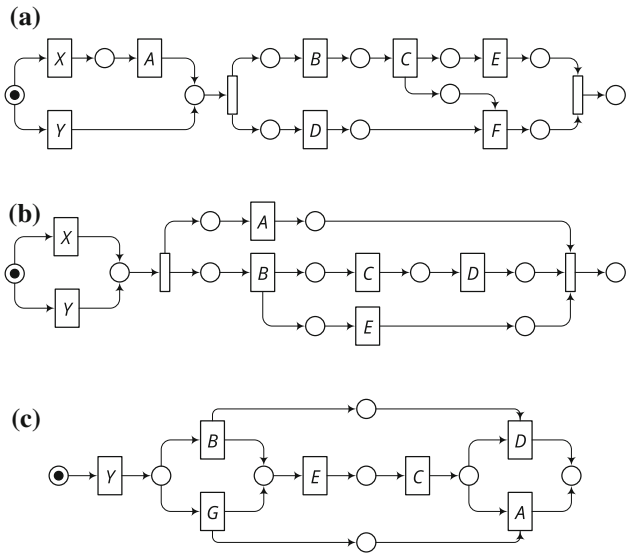


Fig. 3 Net systems for example processes. **a** Net system S_1 of Fig. 1a. **b** Net system S_2 of Fig. 1b. **c** Net system S_3 of Fig. 1c

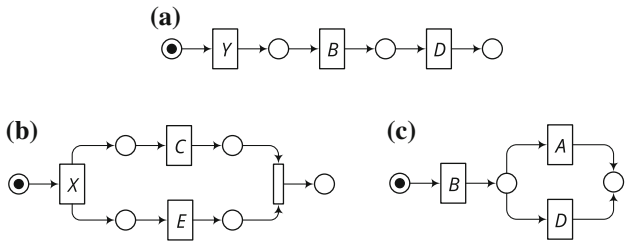


Fig. 4 Net systems for queries. **a** Net system Q_1 of Fig. 2a. **b** Net system Q_2 of Fig. 2b. **c** Net system Q_3 of Fig. 2c

as OR-gateways and exception handling can typically not be represented with free-choice net systems. Also, capturing the data perspective may result in non-free-choice net systems. The latter is illustrated by our examples since all net systems except for the one formalizing also the data perspective of a business process model, i.e., the net system in Fig. 3c, are indeed free-choice. In order to obtain Petri nets that follows a WF-net structure, it may also be needed to apply refactoring techniques to normalize business process models given in other languages [70].

In this article, we assume trace semantics for net systems. Traces are sequences of activity occurrences of a business process. Trace semantics of process behavior are purely sequential and concurrently enabled transitions are captured by interleaved firing [33]. Hence, the set of traces of a system $S = (N, M_i)$ is defined by $\mathcal{T}(N, M_i) = \{\sigma \in T^* \mid \exists\, M \in \mathbb{M}\,[(N, M_i)[\sigma\rangle(N, M)]\}$.

Further, we recall the *soundness* property that was established as a generic correctness criterion for business process models [68]. It requires WF-systems (1) to always terminate, and (2) to have no dead transitions [68]. Both requirements imply proper termination, i.e., for all reachable markings holds that a token in the final place implies the absence of tokens for all other places. A system (N, M_i) is *live*, iff for every marking $M \in [N, M_i\rangle$ and $t \in T$, there exists a marking $M' \in [N, M\rangle$ such that $(N, M')[t\rangle$. System (N, M_i) is *bounded*, iff $[N, M_i\rangle$ is finite. It is *sound*, iff the short-circuit system (N', M_i) is live and bounded.

Finally, we recall that for sound free-choice net systems, there exists a close relation of syntax and semantics. That is, for each path in such a net, there exists a firing sequence that executes each transition on that path in the respective order [39].

3.2 Alignments

In order to evaluate whether a query matches a candidate process model, first and foremost, an alignment between corresponding elements in both models needs to be identified. Once net systems are used for modeling a business process, transitions represent the active parts, i.e., the activities and events of the process. Consequently, an alignment of two net systems is defined as a relation over their transitions, thereby capturing correspondences between the models.

Definition 3 (*Alignment*) Let $S = (N, M_i)$ and $S' = (N', M_i')$, $N = (P, T, F)$ and $N' = (P', T', F')$, be net systems. An *alignment* $\sim \subseteq T \times T'$ associates corresponding transitions of both systems to each other.

An alignment relation captures correspondences between single transitions. However, business process models may assume different levels of modeling granularity so that sets of activities, or sets of transitions of two net systems, need to be related to each other. Formally, such complex 1:n or even n:m correspondences can still be captured with the presented notion of an alignment by including the Cartesian product of the two transition sets in the relation.

Correspondences between process models can be defined manually [21]. Yet this is practically not applicable for the use case of search in large process collections. Our work therefore assumes that an alignment can be constructed without human intervention. Finding correspondences automatically relates to the matching problem known for data schemas and ontologies [28]. Recently, various works adapted techniques from these fields for identifying correspondences between process models, see [12,22,26,45,72]. To overcome terminological heterogeneity, the majority of approaches toward process model search employs string based methods, e.g., substring containment, or edit distances, such as the Levenshtein distance [46]. Cohen et al. [16] provides a comprehensive overview of such techniques. Also, techniques from natural language processing [51], e.g., stop word elimination, word-stemming, and the application of linguistic models, e.g., WordNet [56], have been used to identify correspondences, cf. [26,45].

However, resolving differences in model granularity is far from trivial and only partially addressed by existing matching techniques [72]. Since automatic matching is crucial for evaluating a query against a large set of candidate models, in the remainder of this paper, we assume that only single pairs of transitions of two models correspond to each other. Further, to keep notation concise, we abstract from the actual alignment relation that is constructed between the transitions of a query and a candidate model. That is, for two net systems $S = (N, M_i)$ and $S' = (N', M_i')$, $N = (P, T, F)$ and $N' = (P', T', F')$, we consider two transitions $t \in T$ and $t' \in T'$ to be equal, if and only if they are related by the constructed alignment relation, $t \sim t'$.

In our exemplary candidate models (Fig. 1) and query models (Fig. 2), we observe a rather homogeneous vocabulary so that construction of an alignment is straightforward. For the respective net systems given in Figs. 3 and 4, we already abstracted from the alignment relation. That is, equal labels indeed hint at equal transitions.

3.3 Successor relations

Our goal is not only to evaluate queries against candidate process models, but also to provide a ranking for matching models. To this end, we need a measure for behavioral distance between two models. Since queries typically refer only to a few transitions of the candidate models, this measure needs to be fine-granular and be defined on the level of single transitions. Against this background, our work builds on k-successor relations as defined in [75]. They capture the minimal distance between the occurrences of two transitions

Table 1 Minimal k-successor relations for example net systems from Fig. 3

	X	Y	A	B	C	D	E	F
X	·	·	1	3	4	3	5	6
Y	·	·	·	2	3	2	4	5
A	·	·	·	2	3	2	4	5
B	·	·	·	·	1	1	2	2
C	·	·	·	·	·	1	1	1
D	·	·	·	1	1	·	1	1
E	·	·	·	·	·	1	·	1
F	·	·	·	·	·	·	1	·

(a) $\trianglerighteq_k^{S_1}$

	X	Y	A	B	C	D	E
X	·	·	2	2	3	4	3
Y	·	·	2	2	3	4	3
A	·	·	·	1	1	1	1
B	·	·	1	·	1	2	1
C	·	·	1	·	·	1	1
D	·	·	1	·	·	·	1
E	·	·	1	·	1	1	·

(b) $\trianglerighteq_k^{S_2}$

	Y	A	B	C	D	E	G
Y	·	4	1	3	4	2	1
A	·	·	·	·	·	·	·
B	·	·	·	2	3	1	·
C	·	1	·	·	1	·	·
D	·	·	·	·	·	·	·
E	·	2	·	1	2	·	·
G	·	3	·	2	·	1	·

(c) $\trianglerighteq_k^{S_3}$

in a dedicated trace, or any trace of a system. As such, they allow for measuring a behavioral distance between models by comparing the minimal occurrence distances for transitions that are part of either model.

Definition 4 (*k-Successor relation*) Let $S = (N, M_i)$ be a system, $N = (P, T, F)$, and $k \in \mathbb{N}$. The *k-successor relation* $\triangleright_k^\sigma \subseteq T \times T$ for a trace $\sigma \in \mathcal{T}(N, M_i)$ is defined by:

$$x \triangleright_k^\sigma y \Leftrightarrow \exists 1 \leq i \leq |\sigma| \, [\, \sigma(i) = x \wedge \sigma(i+k) = y]$$

The *k-successor relation* $\triangleright_k^S \subseteq T \times T$ for system S is defined by:

$$x \triangleright_k^S y \Leftrightarrow \exists\, \sigma \in \mathcal{T}(N, M_i) \left[\, x \triangleright_k^\sigma y\right]$$

Both successor relations are the basis for the definition of an up-to-k-successor relation and a minimal k-successor relation, either for a trace or for a system. Let $S = (N, M_i)$ be a system and $k \in \mathbb{N}$. Then, the *up-to-k-successor relation* $>_k^\sigma \subseteq T \times T$ for $\sigma \in \mathcal{T}(N, M_i)$ is defined by $x >_k^\sigma y \Leftrightarrow \exists\, 1 \leq l \leq k \, [\, x \triangleright_l^\sigma y \,]$, whereas the equivalent for S, $>_k^S \subseteq T \times T$, is defined by $x >_k^S y \Leftrightarrow \exists\, 1 \leq l \leq k \, [\, x \triangleright_l^S y \,]$. The *minimal k-successor relation* $\trianglerighteq_k^\sigma \subseteq T \times T$ for $\sigma \in \mathcal{T}(N, M_i)$ is defined by $x \trianglerighteq_k^\sigma y \Leftrightarrow x \triangleright_k^\sigma y \wedge (x, y) \notin >_{k-1}^\sigma$, whereas the equivalent for S, $\trianglerighteq_k^S \subseteq T \times T$, is defined by $x \trianglerighteq_k^S y \Leftrightarrow x \triangleright_k^S y \wedge (x, y) \notin >_{k-1}^S$.

In [75] it was shown that for each system $S = (N, M_i)$, $N = (P, T, F)$, there exists a unique *successor bound* b_S such that all up-to-k-successor relations with $b_S \leq k$ are equal. Further, if $S = (N, M_i)$ is a sound free-choice WF-system, the 1-successor relation along with the set of transitions enabled in M_i provides a complete characterization of traces [75].

Table 1 shows the minimal k-successor relations of the net systems in Fig. 3. For each pair of transitions, a value in the matrix denotes the k of the minimal k-successor relation, which holds for that pair. For the sake of clarity, we disregarded transitions that have no label in Table 2. Further, the up-to-1-successor relations for the net systems in Fig. 4 are illustrated in 2 (the symbol x means that the respective pair is part of the relation).

Table 2 Up-to-1-successor relations for query net systems from Fig. 4

	Y	B	D
Y	·	x	·
B	·	·	x
D	·	·	·

(a) $>_1^{Q_1}$

	X	C	E
X	·	x	x
C	·	·	x
E	·	x	·

(b) $>_1^{Q_2}$

	A	B	D
A	·	·	·
B	x	·	x
D	·	·	·

(c) $>_1^{Q_3}$

4 Fundamentals of behavior inclusion

Our approach assumes a regular process model as a search query, i.e., an example of what the process models sought from a repository can do. This relies on a notion of behavior inclusion: Whatever behavior the query allows for, shall be supported by a matching model. A match may, nevertheless, extend the behavior of the query. In order to rank the results, i.e., promote more relevant matches to the user, we introduce the notion of closeness that quantifies how much additional behavior a match offers, compared to a query. A model that extends a query less than another model is behaviorally closer and therefore considered more relevant to the user's search intention.

This section presents the formal definition of our approach to behavioral querying. First, Sect. 4.1 presents a notion of abstract trace inclusion that is used to characterize behavioral matches. In Sect. 4.2, we show how these notions are decided for bounded net systems and sound free-choice WF-systems. We elaborate on the expressiveness of the proposed query notion in Sect. 4.3. Abstract trace inclusion relies on a notion of trace abstraction. For net systems that show abstract trace inclusion, closeness evaluates the degree of trace abstraction to achieve inclusion required for ranking. The computation for bounded net systems and sound free-choice WF-systems is presented in Sect. 4.4.

4.1 The notion of abstract trace inclusion

In this work, we focus on trace semantics of process models in terms of net systems. Traces have been proposed as a means to capture and compare the behavior of process models [33], which has since seen wide application in the field of process model search [7].

Behavioral inclusion of a query in a matching model implies that each trace a query provides shall be contained in a trace of the matching model, modulo other transition occurrences. However, traces of a query may contain transition occurrences that can be misleading. Let $\overline{T}$ be the subset of the transitions of a net system that represents activities in the original business process model, i.e., transitions $T \setminus \overline{T}$ have been introduced during the transformation of a business process into a net system to preserve the execution seman-

tics of the process. In our example net systems, these transitions are depicted without a label. Obviously, these transitions should not be considered for a query. Therefore, we define projected traces that discard all occurrences of transitions that have no business semantics.

Definition 5 (*Projected trace*) Let $S = (N, M_i)$, $N = (P, T, F)$, be a system, and $\overline{T} \subseteq T$. We define a short-hand notation for transition occurrences of a trace $\sigma \subseteq T^*$ up to index j, such that $\overline{T}^*_{\sigma|j} = \{t_x \in \sigma | x < j \wedge t_x \in \overline{T}\}$. Then, the projection of trace σ by $\overline{T}$ is defined by $\overline{\sigma} = \bigcup_{i=0}^{|\overline{T}^*_{\sigma|n}|} (i, t_i)$ with $t_i \in \overline{T}$, such that $\exists\, j \in \mathbb{N}\, [(j, t_i) \in \sigma \wedge i = |\overline{T}^*_{\sigma|j}|]$.

Consider Q_2 of Fig. 4b, where the final transition has been introduced to obtain a workflow net from the query model. Since this transition conveys no business semantics it shall be disregarded in the traces of Q_2. Hence, the set of projected traces of a net system is defined by $\overline{\mathcal{T}}(N, M_i) = \{\overline{\sigma} \in \overline{T}^* \mid \exists\, \sigma \in T^*, M \in \mathbb{M}\, [(N, M_i)[\sigma\rangle(N, M)]\}$.

To actually decide behavior inclusion, we start by defining the notion of loose abstraction for traces. Intuitively speaking, a sequence of transitions is a loose abstraction of a trace, if it is obtained from the trace by removing some occurrences of transitions. This is different from the notion of abstraction known in behavior inheritance [6], which abstracts all occurrences of a transition. We highlight this aspect by referring to the following definition as loose abstraction.

Definition 6 (*Loose abstraction*) Let $S = (N, M_i)$, $N = (P, T, F)$ be a net system and $T' \subseteq T$ a set of transitions. Let $\sigma \in \mathcal{T}(N, M_i)$ be a trace and $\sigma' \in (T')^*$ a sequence of transitions. Sequence σ' is a *loose abstraction* of σ, denoted by $\sigma' = \tau(\sigma)$, iff there exists an injection $f : \sigma' \mapsto \sigma$ such that

- $f((i, x)) = (j, x)$ and $i \leq j$, and
- $f((i, x)) = (j, x)$ and $f((i + 1, y)) = (k, y)$ implies $j < k$.

Loose abstraction is the basis for abstract trace inclusion, which captures the intuition of behavioral containment of a query in a matching model. It requires each trace of one net system to correspond to a trace of a second system, once a certain set of transition occurrences has been abstracted.

Definition 7 (*Abstract trace inclusion*) Let $S = (N, M_i)$ and $S' = (N', M'_i)$, $N = (P, T, F)$ and $N' = (P', T', F')$, be net systems. S' shows abstract trace inclusion with S, denoted by $S' \sqsubseteq S$, iff for all projected traces $\overline{\sigma}' \in \overline{\mathcal{T}}(N', M'_i)$ there exists a trace $\sigma \in \mathcal{T}(N, M_i)$ and it holds $\overline{\sigma}' = \tau(\sigma)$.

For an example, consider the query model shown in Fig. 2a and its corresponding net system Q_1 in Fig. 4a, which can produce only the trace $\overline{\sigma}' = \langle Y, B, D\rangle$.

Net system S_1, depicted in Fig. 3a, can produce the trace $\sigma = \langle Y, \tau, B, C, E, D, F\rangle$, for which $\overline{\sigma}'$ is an abstraction, illustrated by the highlighted transition occurrences. In fact, each net system depicted in Fig. 3 provides at least one trace that can be abstracted to $\overline{\sigma}'$ and therefore shows abstract trace inclusion with Q_1. Query Q_3, depicted in Fig. 3c, can produce the traces $\langle B, D\rangle$ and $\langle B, A\rangle$. Whereas the first trace is an abstraction of above trace σ, too, S_1 cannot produce any trace where A is executed after B. Consequently, S_1 does not show abstract trace inclusion of Q_3.

Abstract trace inclusion induces a notion of a match between a query model and some process model. This match is complete in the sense that every trace of the query model is an abstraction of some trace of the matching process model. In contrast, we may speak of a partial match, if abstract trace inclusion holds only for certain traces of the query model. Without further restrictions, however, this notion of partial matching is of limited use. A trace is any firing sequence σ that is enabled in the initial marking M_i of a net system (N, M_i), cf. Sect. 3.1. Even if the empty trace would be excluded, traces may consist of only one occurrence of a transition enabled in M_i. Hence, any candidate model that shows at least a single trace containing one of the transitions enabled in the initial marking of the query would be considered to be a partial match. Hence, we resort to complete matches, hereafter.

4.2 Deciding abstract trace inclusion

From Definition 7 follows that a match is decided by comparing all projected traces of a query with potentially all traces of a candidate. This yields a sufficient condition for abstract trace inclusion. Yet, due to cyclic dependencies, the net system of a process model may have an infinite set of traces. Abstract trace inclusion can be decided by means of the state space for bounded net systems, but remains a computationally expensive task. We therefore present a necessary condition that can be evaluated more efficiently based on the successor relations introduced earlier. For sound free-choice WF-systems, the necessary and sufficient conditions coincide, i.e., abstract trace inclusion can be decided efficiently.

Bounded net systems Trace equivalence and inclusion are decidable for unlabeled net systems since they are reduced to the reachability problem [32]. Abstraction of traces, however, implies that equivalence and inclusion of traces cannot be reduced to a problem for unlabeled net systems. Hence, we are able to state decidability only for bounded net systems that have a finite state space.

Theorem 1 *Abstract trace inclusion is decidable for bounded net systems.*

Proof Since the net systems are bounded, their behavior can be represented by finite labeled transition systems. Each transition in this system may be subject to abstraction. This is incorporated by augmenting the transition system with an additional silent transition per non-silent transition that has the same source and target state. Then, the result follows from decidability of trace inclusion for finite labeled transition systems [67]. □

Deciding abstract trace inclusion for bounded net systems is computationally hard in the general case. In fact, the problem of deciding whether two labeled transition systems show trace inclusion is PSPACE-complete [67].

However, if the successor relations of query and candidate net systems are known, a necessary condition for abstract trace inclusion can be specified based on the containment of correspondences of query transitions in the candidate net system and the notion of successor inclusion. It holds between two net systems, if successorship of transitions in one system implies successorship of the transitions in the other system. The intuition behind is that a query model must not define successorship between two transitions that cannot be mirrored by a matching model even if abstraction is applied. Since there is a successor bound for k-successor relations, cf. Sect. 3.3, successor inclusion requires only the investigation of all relations up to this bound.

Definition 8 (*Successor inclusion*) Let $S = (N, M_i)$ and $S' = (N', M_i')$, $N = (P, T, F)$ and $N' = (P', T', F')$, be net systems, and $>_{b_{S'}}^{S'} \subseteq \overline{T}' \times \overline{T}'$ and $>_{b_S}^{S} \subseteq T \times T$ their successor relations, respectively.
S shows *successor inclusion* of S', denoted by $S' \trianglelefteq S$, iff $>_{b'_S}^{S'} \subseteq >_{b_S}^{S}$ and $\overline{T}' \subseteq T$.

As successor inclusion only requires comparing all successors of the query against those of a candidate, it can be computed in quadratic time to the number of transitions in the query net system if the successor relations are known, which is more efficient than computing abstract trace inclusion. Hence, candidate models can be excluded from an exhaustive investigation to decide abstract trace inclusion, if they do not include all successor relationships required by the query.

Theorem 2 *Let S and S' be WF-systems. Then,*

$$S' \not\trianglelefteq S \Rightarrow S' \not\sqsubseteq S.$$

Proof Follows from the definition of $\sqsubseteq$ and the definition of $x >_{b_S}^{S} y$: From $x \not>_{b_S}^{S} y$ follows that $\nexists\ \sigma \in \mathcal{T}(N, M_i)\ [x >_{b_S}^{\sigma} y]$, and thus $\nexists\ 1 \leq i \leq |\sigma|, 1 \leq k \leq b_S\ [\sigma(i) = x \wedge \sigma(i+k) = y]$. □

We clarify the necessary condition with the following example. Any up-to-k-successor relation of candidate model S_1 does not contain the pair $(B, A) \in >_{b_{Q_3}}^{Q_3}$, i.e., there is no trace such that B is executed before A in S_1. Consequently, S_1 cannot show abstract trace inclusion with Q_3.

Sound free-choice WF-systems For the distinguished class of sound free-choice WF-systems, the notion of successor inclusion is even sufficient to decide abstract trace inclusion.

Theorem 3 *Let S and S' be sound free-choice WF-systems. Then,*

$$S' \sqsubseteq S \Leftrightarrow S' \trianglelefteq S.$$

Proof ($\Rightarrow$) Follows from Theorem 2.
($\Leftarrow$) By induction on the length of a trace $\overline{\sigma}' \in \overline{T}(N', M_i')$.
(Base) Let $\overline{\sigma}' = \langle x \rangle$. Since S is sound, there is a trace $\sigma \in \mathcal{T}(N, M_i)$ with $\sigma(j) = x$ for $1 \leq j \leq |\sigma|$. Hence, it holds $\overline{\sigma}' = \tau(\sigma)$.
(Step) Let $\overline{\sigma}'; y \in \overline{T}(N', M_i')$, $\sigma \in \mathcal{T}(N, M_i)$, and $\overline{\sigma}' = \tau(\sigma)$. Let $\overline{\sigma}'(|\overline{\sigma}'|) = x$ and, without loss of generality assume that $\sigma(|\sigma|) = x$ either. Since $x >_1^{S'} y$ it holds $x >_{b_S}^{S} y$. Let M be the marking with $(N, M_i)[\sigma\rangle(N, M)$. Assume that x is not enabled in M. Then, $x >_{b_S}^{S} y$ implies either (1) $x F^+ y$ or (2) x and y are enabled concurrently in some marking $M_2 \in [N, M_i\rangle$, cf. Lemma 1 and Theorem 1 in [74]. If (1), then σ can be extended with transitions $t_1, \ldots, t_n \in T$ such that $\sigma; t_1; \ldots; t_n; y \in \mathcal{T}(N, M_i)$, cf. Theorem 1 [74]. Consider (2). Assume $M_2 \notin [N, M\rangle$ for all markings M_2 that enable x and y concurrently. Then, one of those markings M_2 must have been reached after firing $\sigma(j)$, $1 \leq j \leq |\sigma|$, when firing σ in M_i. Since $\sigma(|\sigma|) = x$, it holds $x >_{b_S}^{S} \sigma(j+1)$, a contradiction with $M_2 \notin [N, M\rangle$. Since $M_2 \in [N, M\rangle$, σ may be extended by firing all transitions $t_1, \ldots, t_n \in T$ to reach M_2, which again yields $\sigma; t_1; \ldots; t_n; y \in \mathcal{T}(N, M_i)$. □

Successor inclusion and thus abstract trace inclusion is decided efficiently.

Corollary 1 *Let S and S', $N = (P, T, F)$ and $N' = (P', T', F')$, be sound free-choice WF-systems. Then, $S' \sqsubseteq S$ is decided in $O(n^3)$ time with $n = \max(|P \cup T|, |P' \cup T'|)$.*

Proof Follows from the computation of k-successor relations of sound free-choice WF-system in $O(n^3)$ time with n as the number of nodes [75] and Theorem 3. □

4.3 Vocabulary and expressiveness of a query

The term vocabulary refers to the concepts of a language by which it is possible to express syntactically correct statements. In our case, the vocabulary used to express a query equals the vocabulary used to express a process model, because we assume a regular process model as query

that describes an example of the desired behavior. Evaluation of the query is conducted on an abstraction over the behavioral semantics expressed by this query, that is, execution traces. This abstraction allows searching among process models that have been created with different process modeling languages, and query and candidate model are not required to be expressed in the same modeling language given that Petri net formalizations for these languages exist. This is the case for the majority of common business process modeling languages such as BPMN, EPC, and BPEL [49].

The expressiveness of a query language is restricted by the provided vocabulary and the underlying matching approach. The reuse of common business process modeling languages to express queries and the concept of abstract trace inclusion determine that a query can request the presence of behavior in a match, but not the absence of activities or behavioral relationships between them. We argue that this restriction is in line with the use case we aim for: to enable non-expert users finding process models that provide desired behavior. Query languages that provide a richer expressiveness, such as expressing the absence of concepts, e.g., [8,15], are more complex and make it difficult to formulate even simple queries.

Abstract trace inclusion for bounded net systems and successor inclusion for sound free-choice WF-systems support the expression of occurrence of activities, their ordering, and the co-occurrence of several activities in a trace. Each activity contained in a query must be contained in a matching model. More precisely, each transition that appears in a trace of the query must also be contained in at least one trace of the match to hold abstract trace inclusion. The same holds for successorship relations. If a pair of transitions appears in any order in a trace of the query, the same order must be mirrored by the successor relation of the match, including repetition of activities. Co-occurrence refers to causal dependencies between actions, i.e., if two activities co-occur in a trace of the query, they must also occur together in a trace of the match. The compliance of these properties in all traces of a match can, however, not be enforced, which is in line with the query being an abstraction of the behavior of a matching model.

Based on this discussion, we investigate matching the example queries with the provided process models. For this purpose, we resort to the basic workflow patterns *sequence*, *parallel split*, and *exclusive choice* that are represented in the example queries in Fig. 2. The corresponding net systems are depicted in Fig. 4.

Sequence. A sequential path in a query model, e.g., Q_1 in Fig. 4a, requires a match to provide a trace that executes corresponding transitions in the same order. This may be satisfied, if the corresponding transitions are on a path in the candidate model, as it is the case for S_2 and S_3 in Fig. 3. But, since our approach considers the actual behavior, rather than graph structures, it may also be matched by models that do not show such a path. This becomes apparent in the successor relations, i.e., if a pair of transitions is contained in the successor relation, there exists a trace that contains occurrences of these transitions in the required order. For instance, the concurrent enabling of B and D in S_1 allows for interleaved firing, which includes the order $B >^{S_1}_{b_{S_1}} D$ requested by Q_1. In contrast, S_1 is not a match for Q_3, as it does not allow executing B before A, whereas this is required by the query.

A transition t that is part of a loop of the query is a successor to itself, i.e., $t >^{S}_{b_S} t$, and needs to be matched by candidates, for instance, by cyclic dependencies, too.

Parallel split. A parallel split allows for interleaved firing of the parallel activities, as they become enabled concurrently, cf. Fig. 4b. Consequently, the query can produce traces containing any order of execution and successor relation pairs of parallel transitions are symmetric, e.g., $C >^{Q_2}_{b_{Q_2}} E$ and $E >^{Q_2}_{b_{Q_2}} C$ in Table 2b. Consequently, the ordering of corresponding transitions must not be restricted in a matching model, i.e., the match must also offer symmetric relation pairs. Hence, models S_1 and S_3 of Fig. 3, where actions C and E are strictly ordered, are no matches for Q_2, whereas S_2 is, due to the parallel split. The interleaved ordering of transitions may also be matched by a cyclic structure in a candidate if it contains corresponding transitions [74].

Exclusive choice. As a query cannot require the absence of a behavioral relation, exclusive choice does not express the mutual exclusion of two activities in traces of a matching model. In comparison with sequence and parallel split that require co-occurrence of transitions in at least one trace, this is relaxed for transitions of a query that are exclusive. For instance, transitions A and D of Q_3 in Fig. 4c are not in a successor relation. While it is required that these transitions appear in traces of a match, they are not required to co-occur in a common trace. Nevertheless, co-occurrence is allowed in any order, including interleaving order. For our examples, this ensues that S_2 matches Q_3. Although the exclusive gateway is structurally matched in S_3, this candidate is not a match due to the implicit place between B and D, which renders the net, depicted in Fig. 4b, non-free-choice. Hence, the query trace $\langle B, A\rangle$ cannot be mirrored in S_3.

The triggering of a process, in terms of start events, can also be formulated in a query up to certain limitations. Since the transformation of process models to net systems [23,49] also maps events to transitions, they can be included in the query as well, as is shown for the queries in Fig. 2a, b. Hence,

events are treated similar to activities and can be used for querying as discussed above. Since queries cannot express the absence of behavior, processes that execute activities before these events cannot be excluded.

4.4 The notion of closeness

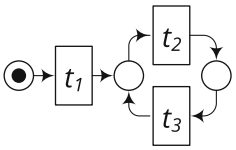
Once process models that match a query model in terms of abstract trace inclusion have been identified, we are interested in a ranking of these models, to assist users in determining which models are more relevant to their query than others. This ranking is guided by the amount of abstraction required to achieve abstract trace inclusion, where less abstraction leads to a better match, since the model is closer to the query. To obtain a ranking, this section presents a closeness measure.

The intuition behind the closeness measure can be summarized as follows. Abstract trace inclusion requires that any two transitions that follow each other in some trace of the query model, also follow each other in some trace of the matching model. Still, several transition occurrences may be abstracted in the trace of the matching model. For each pair of transitions in direct successorship in a trace of the query model, we determine the minimal number of transition occurrences that need to be abstracted in some trace of the matching model. This provides us with a measure how *close* the behavior of the match is to the behavior of the query in terms of single transition pairs. For an example, consider the trace $\sigma = \langle Y, \tau, B, C, E, D, F \rangle$ of S_1 in Fig. 3a and the query trace $\overline{\sigma}'_{Q_1} = \langle Y, B, D \rangle$ of Q_1 in Fig. 4a. For the successor pair (Y, B) of the query trace $\overline{\sigma}'_{Q_1}$, the occurrence of the unlabeled transition τ must be removed from σ; for the successor pair (B, D) of $\overline{\sigma}'_{Q_1}$, transition occurrences C and E must be removed from σ. However, S_1 also allows executing D directly after B. In this case, no transition occurrences need to be removed from σ to achieve abstract trace inclusion.

Lower closeness values mean that, for a transition pair of the query, more transition occurrences of traces of the match need to be abstracted to achieve abstract trace inclusion. Closeness is utilized as a ranking score so that it is applied only when abstract trace inclusion has already been checked. In the same fashion as the decision of abstract trace inclusion, also for the ranking, we first discuss bounded net systems and later turn to sound free-choice WF-systems, which offer the opportunity to compute closeness more efficiently. Finally, we discuss the case of trivial queries comprising a single activity.

Bounded net systems To quantify the amount of abstraction needed to achieve abstract trace inclusion, we first establish the grounding for such a measure by the set of shortest successor maximal traces. Here, the intuition is to obtain set of traces that is finite, but characterizes the essentials of the behavior of a net system. We interpret such behavioral essentials in terms of the 1-successor relation. Hence, we require all traces to be maximal regarding the 1-successor: any possible extension of a trace must not add information on a new successor dependency. Since there may be an infinite number of such traces, we consider only the shortest trace of those with equal 1-successors.

Fig. 5 Cyclic net system



Definition 9 (*Shortest successor maximal traces*) Let $\mathcal{T}(N, M_i)$, $N = (P, T, F)$ be a set of traces. Then, the set of *shortest successor maximal traces* is defined as $\mathcal{T}_{max}(N, M_i) \subseteq \mathcal{T}(N, M_i)$ such that $\sigma \in \mathcal{T}_{max}(N, M_i)$ iff

- σ is successor maximal, i.e., for all $x \in T$ holds that either $\sigma; x \notin \mathcal{T}(N, M_i)$ or if $\sigma; x \in \mathcal{T}(N, M_i)$ then $\sigma(j) = \sigma(|\sigma|)$ and $\sigma(j+1) = x$ for some $1 \leq j < |\sigma|$, and
- σ is the shortest of those with equal successors, i.e., $\forall \sigma' \in \mathcal{T}(N, M_i)$ such that $\sigma'(j) = x$ and $\sigma'(j+1) = y$ implies $\sigma(l) = x$ and $\sigma(l+1) = y$ for $1 \leq j < |\sigma'|$ and $1 \leq l < |\sigma|$, it holds $|\sigma| \leq |\sigma'|$.

Consider the net system depicted in Fig. 5 that comprises a loop. Due to the cyclic dependency between transitions t_1 and t_3, there exists an infinite set of traces, $\{\langle t_1 \rangle, \langle t_1, t_2 \rangle, \langle t_1, t_2, t_3 \rangle, \langle t_1, t_2, t_3, t_2 \rangle, \langle t_1, t_2, t_3, t_2, t_3 \rangle, \ldots\}$. Firing t_2 and t_3 adds new 1-successors to traces. However, after the first complete iteration, i.e., when t_2 has been executed the second time, no information is added to the direct successorship of any of the transitions. Therefore, the set of shortest successor maximal traces for this net is $\{\langle t_1, t_2, t_3, t_2 \rangle\}$.

Using this notion, we are ready to define the closeness measure for ranking models that match a query based on the amount of abstraction required to achieve abstract trace inclusion. Following the above argumentation, we base the measure on pairs of transitions that succeed each other in some query trace and proceed step-wise: First, closeness is defined for a transition pair, grounded on a query trace and a trace of the matching model that induces the smallest amount of abstraction. This is conceptually related to the topic of trace alignment, cf. [11], which seeks to find the optimal alignment between two traces, such that they resemble each other as closely as possible. The closeness between the net systems of a query and a match is derived by considering all transition pairs that succeed each other in some query trace.

Definition 10 (*Closeness*) Let $S = (N, M_i)$ and $S' = (N', M_i')$, $N = (P, T, F)$ and $N' = (P', T', F')$, be bounded net systems, $>_k^{S'} \subseteq \overline{T}' \times \overline{T}'$, $\trianglerighteq_k^{S'} \subseteq \overline{T}' \times \overline{T}'$, $>_k^{S} \subseteq T \times T$, and $\trianglerighteq_k^{S} \subseteq T \times T$ their up-to-k-successor and minimal k-successor relations with successor bounds b_S and b'_S, respectively.
For a pair of transitions $(x, y) \in >_1^{S'}$ the *closeness* in S is defined as

$$c(x, y) = \begin{cases} 0 & \text{if } (x, y) \notin >_{b_S}^{\sigma} \text{ for} \\ & \sigma \in \mathcal{T}_{max}(N, M_i), \\ & \overline{\sigma}' \in \overline{\mathcal{T}}_{max}(N', M_i'), \\ & \overline{\sigma}' = \tau(\sigma) \\ \min\limits_{\substack{\overline{\sigma}' \in \overline{\mathcal{T}}_{max}(N', M_i'), \\ \sigma \in \mathcal{T}_{max}(N, M_i), \\ \overline{\sigma}' = \tau(\sigma), (x,y) \in \trianglerighteq_k^{\sigma}}} & 1/\sqrt{k} \quad \text{otherwise} \end{cases}$$

The overall closeness of S' to S is defined as

$$\rho(S', S) = \sum_{(x,y) \in >_1^{S'}} \frac{c(x, y)}{\left| >_1^{S'} \right|}.$$

In our example, all candidate models depicted in Fig. 1 are matches for the query Q_1 of Fig. 2a, cf. Sect. 4.3. For S_1, $c(Y, B) = 1/\sqrt{2}$, due to the trace with the shortest distance between Y and B results in B being a 2-successor of Y. $c(B, D) = 1/\sqrt{1} = 1$, because, in S_1, D can be executed directly after B. For the closeness of example models with Q_1 we obtain the following values:

$$\rho(Q_1, S_1) = \frac{1/\sqrt{2}+1/\sqrt{1}}{2} \approx 0.85$$
$$\rho(Q_1, S_2) = \frac{1/\sqrt{2}+1/\sqrt{2}}{2} \approx 0.71$$
$$\rho(Q_1, S_3) = \frac{1/\sqrt{1}+1/\sqrt{3}}{2} \approx 0.79$$

From the net systems in Fig. 3, it becomes apparent that S_1 is in fact the closest model to Q_3 as only transition A needs to be abstracted to achieve abstract trace inclusion. Comparing S_2 and S_3 shows that closeness prefers models with close successors: Although the number of abstracted transitions to achieve abstract trace inclusion is 2 in both net systems, S_3 receives higher closeness due to the direct successorship of (Y, B).

Closeness relates to the amount of abstraction and yields a value in the interval $(0, 1]$. A value of 0 cannot be achieved as all successorships of a query are mirrored in a match, eventually. Closeness should equal 1, if all traces of a query model show trace inclusion with the matching model without any abstraction.

Property 1 Let $S = (N, M_i)$ and $S' = (N', M_i')$ be bounded net systems. Then,

$$\overline{\mathcal{T}}\left(N', M_i'\right) \subseteq \mathcal{T}\left(N, M_i\right) \Rightarrow \rho\left(S', S\right) = 1.$$

This property follows directly from the definition of $\rho(S', S)$. For bounded net systems without further restrictions, the reverse does not hold true, though. The measure is based on successorship of transitions so that, for instance, differences in how often a transition occurs may not be taken into account. Since we apply the measure only for ranking of models for which abstract trace inclusion has already been determined, however, this aspect is of minor importance.

Computation of closeness for a given pair of models imposes challenges. The construction of shortest successor maximal traces requires investigation of the complete state space of the model and therefore has to cope with the state explosion problem [67]. Note, however, that this set may be precomputed for models since it does not depend on the query model. While this adds to the required space to store process models in repositories, it saves time during search, which we consider more valuable.

Sound free-choice WF-systems Again, closeness can be characterized based on successor relations for sound free-choice WF-systems. We define successor closeness as follows.

Definition 11 (*Successor closeness*) Let $S = (N, M_i)$ and $S' = (N', M_i')$, $N = (P, T, F)$ and $N' = (P', T', F')$, be sound free-choice net systems, and $>_{b_{S'}}^{S'} \subseteq \overline{T}' \times \overline{T}'$ and $>_{b_S}^{S} \subseteq T \times T$ their successor relations, respectively.
For a pair of transitions $(x, y) \in >_1^{S'}$ the *successor closeness* in S is defined as

$$c_s(x, y) = \begin{cases} 0 & \text{if } (x, y) \notin >_{b_S}^{S} \\ 1/\sqrt{k} \text{ with } (x, y) \in \trianglerighteq_k^{S} & \text{otherwise} \end{cases}$$

The overall successor closeness of S' to S is defined as

$$\rho_s(S', S) = \sum_{(x,y) \in >_1^{S'}} \frac{c_s(x, y)}{\left| >_1^{S'} \right|}$$

Since the set of successors for a net system grows only with the square of the number of transitions in the net, it offers a significant improvement over precomputing and storing shortest successor maximal traces. Indeed, closeness coincides with successor closeness, for sound free-choice WF-systems.

Theorem 4 *Let S and S' be sound free-choice WF-systems. Then,*

$$\rho_s\left(S', S\right) = \rho\left(S', S\right).$$

Proof Let $S = (N, M_i)$, $N = (P, T, F)$, be a system. Let $\sigma \in \mathcal{T}(N, M_i)$ with $\sigma(|\sigma|) = x$ and $(x, y) \in \trianglerighteq_k^S$. Let M be the marking with $(N, M_i)[\sigma\rangle(N, M)$ and, without loss of generality, assume that no transition $t \in T$, $t \neq y$, $\bullet t \cap x\bullet = \emptyset$ is enabled in M. If there exist a transition sequence $\sigma_2 = \langle t_1, \ldots, t_{k-1}\rangle \in T^*$ such that $(N, M)[\sigma_2\rangle(N, M_2)$ and

$(N, M_i)[y\rangle$ independent of trace σ, then the closeness for a transition pair is independent of the relation between query and model traces. Indeed, the existence of a sequence σ_2 follows from Theorem 35 in [75] stating that $\trianglerighteq_1^S$ and, thus, $>_1^S$ provides a complete characterization of trace semantics of S once the set of transitions $\{t \in T \mid (N, M_i)[t\rangle\}$ enabled in M_i is known. □

Again, consider the example processes and query Q_1. Models S_1 and S_2 are sound free-choice WF-systems, and we can compute their closeness to Q_1 purely based on the successor relations. From Tables 1a and 2a we observe, for instance that $Y >_1^{Q_1} B$ and $Y \trianglerighteq_2^{S_1} B$, and and $B >_1^{Q_1} D$ and $B \trianglerighteq_1^{S_1} D$, respectively. For the sound free-choice net systems, we obtain following successor closeness: $\rho_s(Q_1, S_1) = \frac{1/\sqrt{2}+1/\sqrt{1}}{2} \approx 0.85$ and $\rho_s(Q_1, S_2) = \frac{1/\sqrt{2}+1/\sqrt{2}}{2} \approx 0.71$. The observation that $\rho_s(Q_1, S_1) = \rho(Q_1, S_1)$ along with $\rho_s(Q_1, S_2) = \rho(Q_1, S_2)$ is in line with Theorem 4.

In the same manner as for the case of deciding abstract trace inclusion, the utilization of successor relations has the advantage that it allows for efficient computation. Combined with Theorem 4, it is possible to decide a match and compute closeness efficiently, if query and candidate are sound free-choice WF-systems, and only resort to abstract trace inclusion and closeness when they are not. Indeed, for sound free-choice WF-systems, closeness is computed in low polynomial time to the size of the model.

Corollary 2 *Let S and S', $N = (P, T, F)$ and $N' = (P', T', F')$, be sound free-choice WF-systems. Then, $\rho_S(S', S)$ is computed in $O(n^3)$ time with $n = \max(|P \cup T|, |P' \cup T'|)$.*

Proof Follows from the computation of k-successor relations of sound free-choice WF-system in $O(n^3)$ time with n as the number of nodes [75] and Theorem 4. □

Trivial queries The aforementioned closeness measures quantify the amount of abstraction needed between pairs of activities or transitions, respectively. Consequently, they are not applicable for trivial queries that comprise a single activity and for which the successor relation is empty. For these cases, we argue that an activity that is closer to the beginning of a process or its end may be considered more important, as it represents a preliminary step or a result of a process. We incorporate this idea in a closeness measure for trivial queries as follows.

Definition 12 (*Closeness for trivial queries*) Let $S = (N, M_i)$ and $S' = (N', M_i')$, $N = (P, T, F)$ and $N' = (P', T', F')$, be bounded net systems, such that $T' = \{t'\}$. $>_k^S \subseteq T \times T$, and $\trianglerighteq_k^S \subseteq T \times T$ the up-to-k-successor and minimal k-successor relations of S with successor bound b_S. Let $T_i = \{t \in T \mid \exists\, \sigma \in \mathcal{T}(N, M_i) : \sigma(1) = t\}$ and $T_o = \{t \in T \mid \exists\, \sigma \in \mathcal{T}(N, M_i), M_i[\sigma\rangle M : \sigma(|\sigma|) = t \wedge \nexists\, t' \in T : M[t'\rangle\}$ be sets of initial and final transitions of S.

For a pair of transitions $(x, y) \in (T \cup T') \times (T \cup T')$, *closeness for trivial queries* is defined as

$$c_0(x, y) = \begin{cases} 0 & \text{if } (x, y) \notin >_{b_S}^S \\ 1 & \text{if } x = y \\ 1/\sqrt{k+1} & \text{otherwise} \\ \text{with } (x, y) \in \trianglerighteq_k^S & \end{cases}$$

The overall closeness of the trivial query S' to S is defined as

$$\rho_0(S', S) = \max\Big(\max_{t \in T_i, t' \in T'} c_0(t, t'), \max_{t \in T_o, t' \in T'} c_0(t', t)\Big)$$

Given a trivial query that consists only of activity *deliver product (C)*, represented by the net system $Q = ((\emptyset, \{C\}, \emptyset), \emptyset)$ it is obvious that all candidate models (Fig. 3) satisfy abstract trace inclusion. To rank these models, we compute $\rho_0(Q, S_1) = \max(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{4}}, \frac{1}{\sqrt{4}}) \approx 0.58$ ($T_i = \{X, Y\}, T_o = \{\tau\}$), $\rho_0(Q, S_2) = \max(\frac{1}{\sqrt{4}}, \frac{1}{\sqrt{4}}, \frac{1}{\sqrt{3}}) \approx 0.58$ ($T_i = \{X, Y\}, T_o = \{\tau\}$), $\rho_0(Q, S_3) = \max(\frac{1}{\sqrt{4}}, \frac{1}{\sqrt{2}}) \approx 0.71$ ($T_i = \{Y\}, T_o = \{D, A\}$).

5 Efficient querying based on behavior inclusion

Having discussed the conceptual background for querying based on behavior inclusion, we now turn the focus on the application of the presented concepts for efficient querying and also discuss limitations of the presented approach.

The most simplistic technique to find all models that match a given query is to compare the query with each model, decide a match, and compute its closeness as to provide a ranking of all models that satisfy the query. The search time of such an approach can be approximated by the product of the number of models in the collection and the average time to decide a match. Hence, it grows linearly to the size of the process model collection and cannot be considered very efficient.

In order to make search efficient, it is required to avoid exhaustively examining every process model in the collection [58], and exclude process models that cannot satisfy the matching criterion early. A typical solution to this problem is commonly referred to as the filter and verification paradigm known from graph databases, cf. [61], and consists of two phases. In the first phase, referred to as *filter*, all process models that cannot satisfy the given query are excluded by an inexpensive operation and a candidate set is produced that is significantly smaller than the original model collection. In a subsequent *verification* phase, each process model of the

Table 3 Example inverted index

Key	Models
(A, B)	$\{S_1, S_2\}$
(A, C)	$\{S_1, S_2\}$
…	…
$(\boldsymbol{B},\boldsymbol{A})$	$\{S_2\}$
(B, C)	$\{S_1, S_2, S_3\}$
$(\boldsymbol{B},\boldsymbol{D})$	$\{S_1, S_2, S_3\}$
…	…
(C, A)	$\{S_2, S_3\}$
(C, D)	$\{S_1, S_2, S_3\}$
..	…
(Y, A)	$\{S_2, S_3\}$
(Y, B)	$\{S_1, S_2, S_3\}$
..	…

candidate set is examined exhaustively to decide a match. Additional computation, e.g., the computation of closeness to provide a ranking, is only performed in the latter phase. Therefore, optimization of the search performance generally addresses to optimize the filter phase to be most effective, i.e., exclude as many process models as possible, while the used operations should be very fast.

In our case, we benefit from the necessary condition of trace inclusion, i.e., all successor relation pairs of the query Q must be included in the successor relation of a matching process model P, i.e., $>_{b_Q}^{Q} \subseteq >_{b_P}^{P}$, cf. Theorem 2. In order to save computation time, we can reduce this to matching only the 1-successor relation of a query, i.e., we require that $>_1^Q \subseteq >_{b_P}^P$, cf. Definition 11.

Based thereon, we utilize an inverted index to quickly find a candidate set. Inverted indexes store a set of attributes and for each attribute a list of records that show this attribute [40]. Here, such an attribute is a successor relation pair and the associated records are process models that comprise this relation.

To construct the inverted index, we examine all process models of a collection. For every process model, we extract all successor relation pairs $(x, y) \in >_{b_P}^P$ and add each as a key to the inverted index. If two or more models share a relation, only one key will be added to the index which points to a list of these models. By requesting a key, the index allows for quick identification of all process models that contain this particular successor relation. The index also allows for the iterative addition and removal of process models by adding models and keys or removing models from the list of corresponding keys; if a key does not point anymore to at least one process model, it is removed. Table 3 shows an excerpt of such an index for the example processes, shown as net systems in Fig. 3, derived from their successor relations, cf. Table 1.

Filter To discover the candidate set in the filter phase, we extract the 1-successor pairs from the query and use it to find models in the inverted index. That is, we request the conjunction of these relations, which returns a candidate set of models that include all 1-successors from the query and hence satisfy the necessary condition of a match.

Consider the example query Q_3 depicted in Fig. 4c. It comprises the following 1-successor relation, which are highlighted in the inverted index, cf. Table 3: $>_1^{Q_3} = \{(B, A), (B, D)\}$. It is easy to see that only model S_2 includes both relation pairs.

For trivial queries comprising only a single activity or transition, respectively, we observe that the successor relation is empty. Consequently, we use the aforementioned index in a slightly different way. That is, we search for models whose key contains the requested activity in either position of the index entry. Note that according to the notion of abstract trace inclusion, cf. Definition 7, answering such a trivial query requires that in a matching model the respective activity must be executable. This is a stronger statement than requiring that the activity must exist in the process model.

Verification For each model in the candidate set, we then verify the sufficient condition of a match, i.e., decide a match with the sufficient condition and compute its closeness to the query as a score for ranking several result models. Whenever possible, we rely on successor inclusion and the closeness definition ρ_s for sound free-choice WF-systems, as it allows deciding a match and computing closeness more efficiently than abstract trace inclusion and closeness ρ for bounded net systems. As the successor closeness ρ_s can be computed purely by the minimal k-successor relations, cf. Definition 11, we also store the minimal k-successor relations of sound free-choice WF-systems, when adding the respective process model to the index. For process models that are not traced back to sound free-choice WF-systems but bounded net systems, we need to exploit the state space of the respective models to decide abstract trace inclusion and compute the closeness toward the query. Finally, for trivial queries comprising a single activity, the closeness definition given in Definition 12 applies.

Limitations Turning to the applicability of the presented approach, we observe that its major limitation is the requirement of boundedness of process models. Process models that cannot be traced back to bounded net systems, i.e., that have an infinitely large state space, are not considered by our approach. For unbounded net systems abstract trace inclusion may not be decidable and closeness cannot be computed. However, unboundedness is commonly seen as a behavioral error of a process model [68] that should be resolved prior to making the model available for reuse in a process model repository.

6 Evaluation

To evaluate our approach toward effectiveness and efficiency, we conducted a series of experiments. This section first elaborates on the used data set and experimental setup, before we turn to the obtained results.

6.1 Data set

We used two experimental data sets based on the SAP reference model collection [17]. Published by business software vendor SAP in 1997, it comprises over 600 process models in EPC notation which represent reference processes implemented in the SAP R/3 system. Mendling [55] elaborates on a variety of characteristics of this model collection. The choice to use models from this collection is motivated by two aspects in particular. On the one hand, a variety of models in this collection show a functional overlap and thus qualify for experiments on querying process models. Yet, the models use a homogenous vocabulary, which makes the construction of an alignment straight-forward. Consequently, this collection allows us to focus on the evaluation of the actual query mechanism limiting the bias that is induced by the performance of a technique for matching the activities and events of two process models. On the other hand, these models have repeatedly been used for empirical research in this field, e.g., to evaluate the effectiveness of process model similarity measures with regards to human assessment [21,42].

Since we utilize execution semantics of process models to query by means of behavior inclusion, we had to exclude models that showed syntactic errors or ambiguous instantiation semantics [18]; both issues prohibit deriving correct execution semantics. The latter issue refers to a missing start join, i.e., due to multiple start events that cannot be joined in a single node, it is not possible to decide upfront which events instantiate the process, and consequently makes it impossible to derive an initial marking, cf. Sect. 3.1. Hence, for our experiments, we used a subset of the models in that collection.

6.2 Setup

Given the homogenous vocabulary of the SAP reference model collection, we leveraged the string edit distance [46] as a basis to compute the similarity of activity labels. That is, we established a correspondence between two activities, if the similarity of their labels exceeds a certain threshold. For the experiments presented in the remainder, we selected a similarity threshold of 0.65.

We implemented query matching and closeness computation, presented in Sect. 4, in Java using the jBPT library[1]. For the inverted index, we adopted Apache Lucene[2]. We extracted all eventual successor relation pairs of a process model, i.e., $>^P_{bp}$, and used these as index terms, referencing the process' net system and its minimal k-successor relation. In the filtering phase, we extracted all 1-successor relation pairs from a query, i.e., $>^Q_1$, and searched for a conjunction of these terms in the index.

In a first experiment, we tested the effectiveness of the proposed approach. We set up an experiment to measure how well querying by behavior inclusion correlates with human assessment. For this experiment, we chose 34 candidates of the SAP dataset. From the candidates, we manually generated 10 query models and paired each query model with 10 of the candidates. Seven process modeling experts were then asked to determine for each pair of query and candidate process model, whether the process model matches the query and to rank all matches according to their favor. To decide a match, they were given the guideline that "a matching process model shall be able to replay the behavior of a query". For each query, we computed the list of candidates that showed abstract trace inclusion ranked by their closeness and compared this list with the expert judgment.

We measure effectiveness in terms of precision (the ratio of relevant identified models and all identified models) and recall (the ratio of relevant identified models and all relevant models). The motivation for using these measures is that, in contrast to similarity search of process models, querying of process models induces a fixed and well-defined result set, i.e., set of relevant models for a query. We relied on macro-averaging so that precision and recall values are aggregated by the arithmetic mean over the results for all query models. To evaluate the effectiveness of the closeness scores for ranking query results, we assess the correlation between the closeness values and the ranking according to human judgment.

In a second experiment, we assessed the performance of our approach and its scalability as follows. For increasing sizes of a base model collection, we searched for 100 randomly chosen queries and measured the median search time over these search runs. The query models were chosen randomly from the model collection to account for various levels of complexity when processing queries. We conducted each search run twice with the same queries, respectively: first as a sequential search, where we compared the query with each model in the current collection and second as efficient search by means of the filter and verification approach using the inverted index, presented in Sect. 5.

All experiments have been conducted on a 2.8GHz CPU running a Java 1.7 virtual machine with 1GB of heap memory assigned, of which only 400MB were used. Matching has been implemented single threaded. For sound free-choice

[1] http://code.google.com/p/jbpt/.

[2] http://lucene.apache.org/core/.

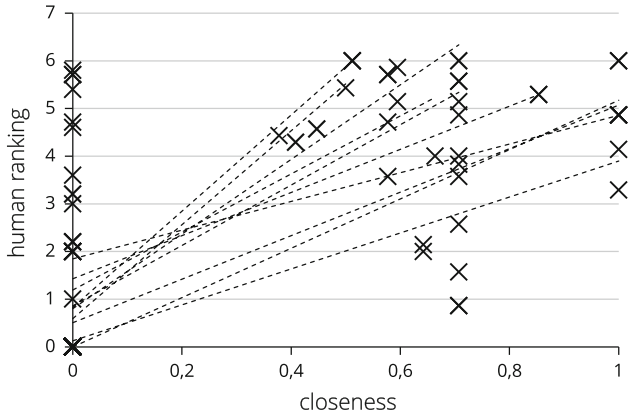


Fig. 6 The obtained closeness values relative to the average human ranking, *dashed lines* indicating the linear least-squares regressions for all ten different queries

WF-systems, minimal k-successor relations have been precomputed and cached in memory, to exclude times for successor computation and I/O operation during search. This aligns with the outlined indexing approach, where these relations are computed and stored when a model is added to the index.

6.3 Results

Testing the effectiveness of the presented approach to behavioral querying against the human assessment, we obtain a mean precision value of 0.983 and a mean recall value of 0.698. Clearly, the former indicates that identified models are indeed considered by the process modeling experts to be results for the query. Also, we consider the recall value to be acceptable since querying yields correct matches, whereas humans may be more forgiving when comparing process models. Further, we observe that the mean recall values have a standard deviation of 0.27 among all queries. We found the cause for this in one particular query that obtained a considerable low recall value of 0.167. This query features two activities that emerge from a parallel gateway. However, several subjects considered process models as a valid match, if they allow executing both activities in at least one of the possible orders. Hence, for some types of queries, partial match results appear to be also relevant, an aspect that shall be addressed in future work.

With respect to the closeness based ranking, Fig. 6 indeed shows that high closeness values are obtained for models that are also ranked high according to human judgment. Figure 6 also illustrates the linear least-squares regressions for all ten different queries (on average, the R^2 value of the regression models is 0.61). Those further underpin the trend of high closeness values being obtained for models that are also ranked high by humans. For our comparably small data set, however, these results turned out to be not statistically significant.

We conducted a detailed inspection of the models and their rankings to understand in which cases the closeness measure approximates the human ranking well. We illustrate our observations with the examples given in Fig. 7. Here, the query (a) consists of an activity *Delivery Processing (A)* followed by *Goods Issue Processing (B)*. All three candidates from the SAP reference model match this query; paths that provide traces complying with abstract trace inclusion of the query are highlighted bold. Process model (b) is a perfect match and therefore has an optimal closeness. This is in line with the human assessment, which ranks (b) high. For the remaining two models, we notice that model (c) requires execution of one activity, *C*, between *A* and *B*, whereas process model (d) requires execution of two activities, *D* and *E*. Hence, model (c) is closer to the query than model (d). However, for this case, the human ranking preferred model (d). We attribute this to the fact that model (c) is larger than model (d), i.e., it has more nodes. We observed that models requiring less effort to complement the query were often ranked higher than more complex models. As such, the appropriateness of the closeness measure for ranking models that show significant differences in their sizes has to be further investigated in future work.

Turning to the evaluation of efficiency, Fig. 8 shows the required search time in milliseconds for the baseline sequential search (dashed curve) and efficient search (solid curve) on a logarithmic scale. For the efficient search, we further illustrate the amount of time that has been used by the Lucene inverted index to exclude models in the filter phase (dotted curve). The difference between the dotted and the solid curve represents the amount of time required for the validation phase of efficient search.

For the sequential search, we observe a strong raise until 50 models that results from the choice of query models. At these times, less then half of all query models are contained in the collection. Hence, only very few models are found that share correspondences with a query and qualify for deciding a match. From there on, the curve indicates linear growth from 12 to 30 s median search time over the increasing process model collection. At a collection size of 450 models, sequential search consumed approximately 15.5 s on average. The negative peaks in the curve stem from the complexity of the chosen queries. Some queries were trivial or rather simple. If many such queries were chosen for search at a certain collection size, the median search time dropped significantly.

Using the inverted index, in turn, shows a significant gain in search speed with a median remaining below 25 ms for all 450 models in the collection and a maximal average search time being 551 ms. With these numbers, the index shows a performance of two orders of magnitudes faster than sequential search. Looking at the curve for filter pruning (dot-

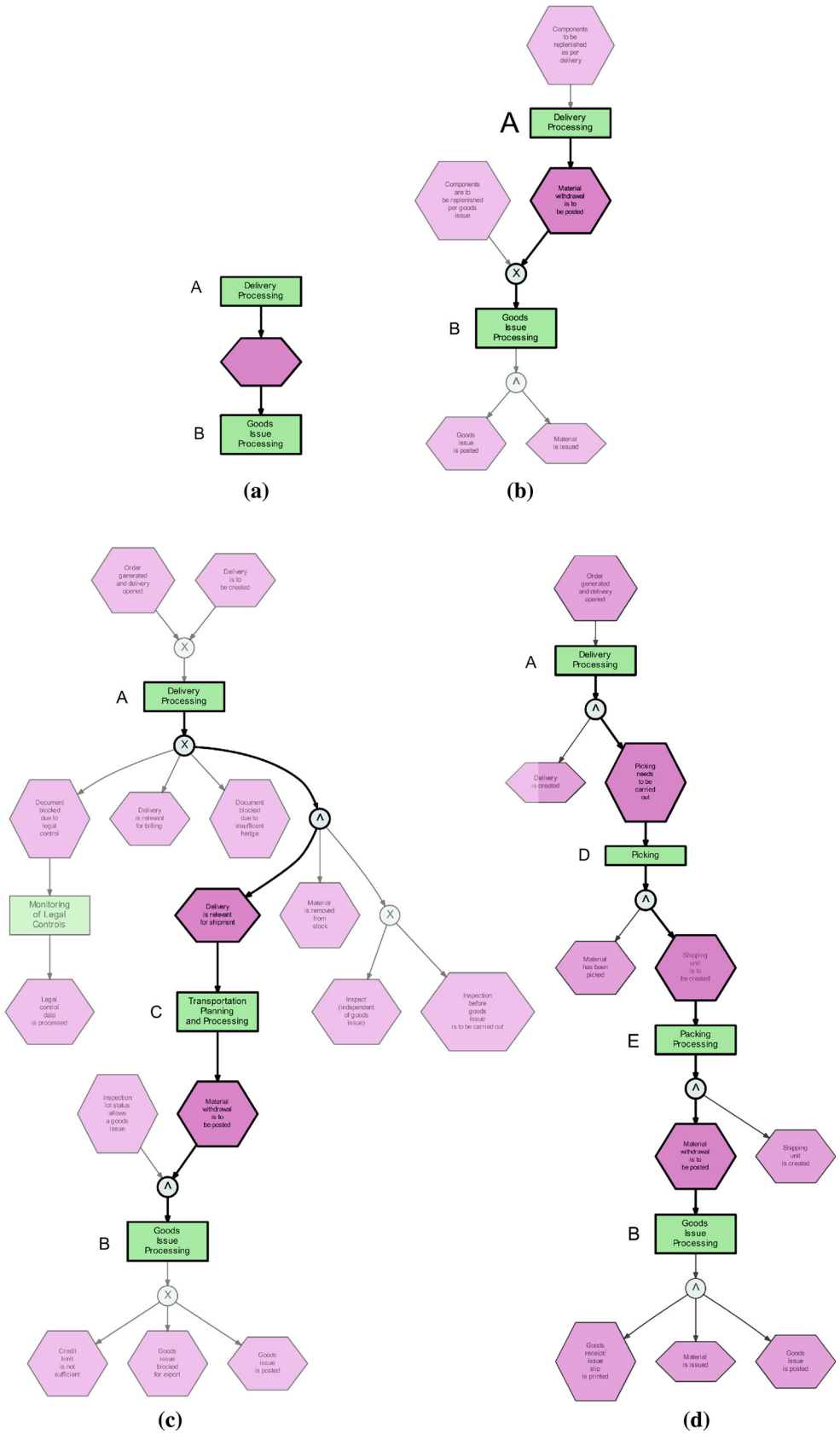


Fig. 7 Example query **a** and process models **b–d** from the SAP reference model that share activities *A* and *B* in the same execution order. Identical activities in are marked by indices *A–E*. Due to space limitations, labels are not readable on purpose

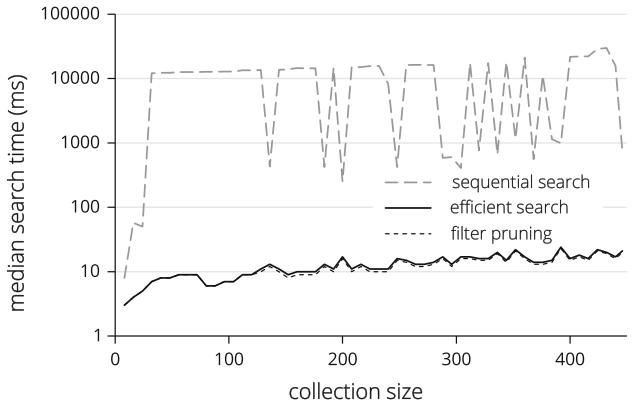


Fig. 8 Search efficiency over increasing size of a model collection for sequential (*dashed curve*) and efficient (*solid curve*) search

ted curve), the inverted index proves to be very effective in excluding invalid candidates early. That is, the median verification time constitutes only 1 ms, expressed by the closeness and affinity of the curves for efficient search and filter pruning.

7 Related work

This work has been inspired by Query by Example [79], a visual query language for relational data bases, where a user provides constant values for some attributes and leaves the others blank. All records that contain these values for the respective attributes are returned in response. Hence, the user proposes an example that becomes completed by the query processor. The simplicity of this approach suggests that also non-expert users become able to query a database [79]. We applied this idea to the domain of business process models, where a user proposes an incomplete example, which is completed by the query response.

Process model search Process model search is generally approached from two different perspectives: similarity search and querying [24]. Similarity search addresses such use cases, where duplicates or sufficiently similar process models are sought, given a rather complete specification of a query process model. For example, to integrate two process model collections or to identify reference processes for a set of process variants [24]. A number of similarity measures for process models has been investigated exhaustively in [7]. In contrast to similarity search, also referred to as inexact matching, querying imposes exact matching, where the query is a precise specification of only those aspects that are relevant. A match can extend a query to a large extent, which would result in a rather low similarity of query and match. Consequently, means for matching must ensure that the aspects formulated in the query are consistently met in a matching model. Querying is particularly useful in the phase of process model design, e.g., to find models that already comprise the desired process or to discover potential completions for an incomplete process model [41,53]. Search may be guided by the structure or the behavior of process models [7,25,42].

Structural search Simple approaches to structural similarity compute the ratio of common features, e.g., activities, compared with the overall number of these features [26, 76] or map more complex features into high-dimensional spaces and compute a distance between corresponding vectors [38]. The former can easily be reused for querying, by requiring that all features of the query must be contained in a matching model [9,63]. This is more complicated for approaches based on vector spaces, as distance functions consider all dimensions and not only those of a query.

Similarity search techniques that resort to the graph edit distance [13], e.g., [21,47], can be used for querying only to a very limited extent, as the graph edit distance is a symmetric relation. Common subgraph isomorphisms [14] have been used to assess the similarity of process models, cf. [31,34]. Such approaches can be altered to suit querying by requiring that each edge in the query model is either represented as an edge or a path in a matching model [20,50]. Closely related is [37], where a path-based index is proposed that stores paths of length n from a process model graph in an inverted index. An index structure that stores hierarchical decompositions of process model graphs is introduced in [66], where identical fragments of process models are unified. This allows for efficient identification and retrieval of process models from a repository that share exact-matching fragments with a query.

Behavioral search Graph based approaches to business process model search suffer from the issue that most languages are not grounded on a small set of semantically orthogonal concepts but allow for different ways to model equal or very similar behavior. This suggests relying on the execution semantics for search.

Techniques to evaluate the behavioral similarity of process models may be based on the state space [1,57,64] or sets of execution sequences of process models [1,30]. However, due to the state explosion problem [67], computation of both is hard and requires various reduction techniques [29]. For instance, in [71], the authors propose principal transition sequences that abstract from recurring states, e.g., due to loops, and hence truncate infinite sequences. We tackled the problem of infinite traces by the notion of shortest successor maximal traces, cf. Definition 9.

To avoid the exponential growth of these models induced by concurrency, successor relations may be used as grounding for similarity assessment. Such relations, also used in

this work, express whether a pair of activities can occur in a certain order in any execution of the process model. Two variants of such relations have been used for process model search: (a) 1-successor relations that capture pairs of activities that can occur directly after one another [27,36,78]; (b) eventual successor relations that capture pairs of activities that may co-occur in a process execution, also if they do not occur directly after one another [27,35,42]. Successor relations may be derived from the model structure [74] or Petri net unfoldings [35,73].

Most of the approaches use the Jaccard-coefficient of successor relations to assess the behavioral similarity of a query and a candidate model [27,36,42,78]. Since the Jaccard-coefficient is symmetric, these approaches apply to similarity search. However, successor relations may be compared asymmetrically, cf., [35,44], to be used for querying; an approach also followed in this work.

Often query and candidate models are assumed to stem from the same domain, i.e., are represented as process graphs. For querying, specific query languages have also been proposed for specific process modeling languages. For instance, BP-QL [8] focuses on BPEL processes, BPMN-Q [4,5] addresses BPMN, and IQM-QL [15] requires a proprietary XML-based representation of process models. Linear temporal logic may also be used as a query language, see [65]. Our approach avoids introducing a new language to express queries, but rather enables reuse of the process modeling language. We argue that this approach can be more easily adopted by non-expert users who have little to no background in formulating queries by means of formal languages.

The authors of [35] propose a query language that comprises behavioral relations combined with logical operators. This is similar to our approach, in that we also extract behavioral relations from the query and use them to decide on matching models. However, in [35] a query is a textual specification of successor relations. Moreover, their approach cannot ensure that several behavioral relations can be satisfied in one trace; for each behavioral relation there might be a distinct trace which contains this relation. In our approach, behavioral inclusion is based on traces, and hence, all behavioral relations that appear in a trace of the query must also appear in a trace of a match.

Indexing and ranking Most work on process model search neglects efficiency of search, but rather focuses on the matching techniques. The time to search for matching process models is approximately the product of the number of comparison operations required and the average time for comparison computation. In general, obtaining a search result requires exhaustive search, i.e., comparing the query with each candidate model. It is virtually infeasible to provide a meaningful ordering among process models, which inhibits the use of many traditional index structures. Mapping features to vector spaces, e.g., by activities, control flow edges, or behavioral relations, yields thousands of dimensions that will grow whenever models are added to the repository and therefore become unmanageable [10].

In the context of similarity search, this can be tackled with the use of metric space index structures [77], which partition a set of models purely by distance and allow excluding partitions of the candidate model set, if they cannot contain satisfactory matches. However, this requires the similarity measure, or distance, to be a metric. Both the graph edit distance and the Jaccard coefficient yield a metric [48] and have been used for metric space indexing in [42,43].

Other solutions use a two-phase approach of filtering and verification discussed earlier. For instance, in [4,8,34–37] an inverted index has been applied to exclude those models that share no activities with the query. We followed a similar route in this work, and used containment of successor relations as a necessary condition for a match. The inverted index is used to exclude invalid candidates early, and we precisely decide a match and assess the quality of a match in terms of closeness in the verification phase.

Almost none of the discussed approaches toward querying of process models addresses the ranking of search results. The approach presented in [50] proposes using precision and recall measures introduced in [1] that quantify shared behavior in comparison with the allowed behavior of the match and the query, respectively. In prior work [44], we leveraged the ratio of common activities to the size of the matching model to prefer models that add less additional behavior than others. Here, we introduced the notion of closeness to measure the amount of abstraction required to match a model with a query.

8 Conclusion

Nowadays, companies maintain large collections of business process models, often using repositories to manage them. Although such collections bear a knowledge asset of great value, we see that business processes are often designed from scratch, whereas the existing knowledge could have improved the design process in many ways. Reuse suggests to increase efficiency and quality of designed models and to improve consistency among several models. Yet, in order to reuse this knowledge asset, effective process modeling search capabilities are required, in particular, by means of process querying.

In this article, we presented a novel querying approach for process models. A query comprises an example process model that contains only few yet relevant activities and their ordering relations. Model that are complete with respect to the query and are able to replay it are considered a

match and are presented in a ranked order. Ranking is guided by the semantic closeness of a match to the query, i.e., by the amount of behavioral abstraction from the query.

Our approach is based on abstract trace inclusion, which requires that each trace a query can produce is included in an abstraction of at least one trace of a matching model. A ranking score, closeness, evaluates how much abstraction is needed to match a process model with a query and is used to provide a ranking among several matches to one query. For the class of sound free-choice WF-systems, successor relations have been proven to decide a match and compute closeness efficiently. We illustrated how these concepts can be used to query process models and support the construction of an index structure.

We evaluated our approach toward its effectiveness by means of a user study, where we focused on the appropriateness of our approach compared with a human assessment of potential matches. In a quantitative experiment, we measured search times for increasing sizes of process model collections and showed that the index structure provides good scalability in a practical setting.

Our experiments indicated that also partial matching of a query model is worth to be followed upon. This requires further investigation toward human assessment and expectations when searching by means of behavior inclusion. Accordingly, we may adapt semantics of extracting behavioral relations from queries and matching these with behavioral relations of candidate models. Another direction for future work is the investigation of approaches, where one query cannot be satisfied by a single model, but by a composition of models. This would assist in the creation of new models out of existing knowledge in the fashion of dynamically assembling a model from modules. We have approached this in earlier work [5,62], yet this addressed only structural search approaches with limited capabilities.

References

1. Aalst, W.M.P.V.D., Medeiros, A.K.A.D., Weijters, A.J.M.M.: Process equivalence: comparing two process models based on observed behavior. In: International Conference on Business Process Management (BPM 2006), vol. 4102 of Lecture Notes in Computer Science, pp. 129–144. Springer (2006)
2. Akkiraju, R., Ivan, A.: Discovering business process similarities: an empirical study with SAP best practice business processes. In: Service-Oriented Computing, Lecture Notes in Computer Science, vol. 6470, pp. 515–526. Springer (2010)
3. Awad, A., Decker, G., Lohmann, N.: Diagnosing and repairing data anomalies in process models. In: Rinderle-Ma, S., Sadiq S.W., Leymann F. (eds.) Business Process Management Workshops, Lecture Notes in Business Information Processing, vol. 43, pp. 5–16. Springer (2009)
4. Awad, A., Polyvyanyy, A., Weske, M.: Semantic querying of business process models. In: Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference, pp. 85–94. IEEE Computer Society, Washington, DC (2008). doi:10.1109/EDOC.2008.11
5. Awad, A., Sakr, S., Kunze, M., Weske, M.: Design by selection: a reuse-based approach for business process modeling. In: Proceedings of the 30th International Conference on Conceptual Modeling, ER'11. Springer
6. Basten, T., van der Aalst, W.M.P.: Inheritance of behavior. J. Log. Algebr. Program. **47**(2), 47–145 (2001)
7. Becker, M., Laue, R.: A comparative survey of business process similarity measures. Comput. Ind. **63**(2), 148–167 (2012)
8. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes with bp-ql. Inf. Syst. **33**(6), 477–507 (2008). doi:10.1016/j.is.2008.02.005
9. Belhajjame, K., Brambilla, M.: Ontology-based description and discovery of business processes. In: Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing, vol. 29, pp. 85–98. Springer, Berlin (2009)
10. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In. In International Conference on Database Theory, pp. 217–235 (1999)
11. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace alignment in process mining: opportunities for process diagnostics. In: Proceedings of the 8th International Conference on Business Process Management, BPM'10, pp. 227–242. Springer, Berlin (2010)
12. Branco, M.C., Troya, J., Czarnecki, K., Küster, J.M., Völzer, H.: Matching business process workflows across abstraction levels. In: R.B. France, J. Kazmeier, R. Breu, C. Atkinson (eds.) MoDELS, Lecture Notes in Computer Science, vol. 7590, pp. 626–641. Springer (2012)
13. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognit. Lett. **1**(4), 245–253 (1983). doi:10.1016/0167-8655(83)90033-8
14. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recogn. Lett. **19**, 255–259 (1998)
15. Choi, I., Kim, K., Jang, M.: An xml-based process repository and process query language for integrated process management. Knowl. Process Manag. **14**(4), 303–316 (2007). doi:10.1002/kpm.290
16. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: S. Kambhampati, C.A. Knoblock (eds.) IIWeb, pp. 73–78 (2003)
17. Curran, T., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice-Hall Inc., Upper Saddle River, NJ (1997)
18. Decker, G., Mendling, J.: Process instantiation. Data Knowl. Eng. **68**, 777–792 (2009). doi:10.1016/j.datak.2009.02.013
19. Desel, J., Esparza, J.: Free-Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (1995)
20. Deutch, D., Milo, T.: Querying structural and behavioral properties of business processes. In: Proceedings of the 11th International Conference on Database Programming languages, DBPL'07, pp. 169–185. Springer, Berlin (2007)
21. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Inf. Syst. 36(2), 498–516 (2011). doi:10.1016/j.is.2010.09.006. Special Issue: Semantic Integration of Data, Multimedia, and Services
22. Dijkman, R.M., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning business process models. In: EDOC, pp. 45–53. IEEE Computer Society (2009)
23. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in bpmn. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)

24. Dijkman, R.M., Rosa, M.L., Reijers, H.A.: Managing large collections of business process models: current techniques and challenges. Comput. Ind. **63**(2), 91–97 (2012)
25. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity search of business process models. IEEE Data Eng. Bull. **32**(3), 23–28 (2009)
26. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: APCCM '07: Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling, pp. 71–80. Australian Computer Society Inc., Darlinghurst, Australia (2007)
27. Eshuis, R., Grefen, P.: Structural matching of BPEL processes. In: Proceedings of the 5th European Conference on Web Services, pp. 171–180. IEEE Computer Society, Washington, DC (2007). doi:10.1109/ECOWS.2007.26
28. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Berlin (2007)
29. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: U. Dayal, J. Eder, J. Koehler, H.A. Reijers (eds.) BPM, Lecture Notes in Computer Science, vol. 5701, pp. 278–293. Springer (2009)
30. Gerke, K., Cardoso, J., Claus, A.: Measuring the compliance of processes with reference models. In: On the Move to Meaningful Internet Systems: OTM 2009, Lecture Notes in Computer Science, vol. 5870, pp. 76–93. Springer, Berlin (2009)
31. Grigori, D., Corrales, J.C., Bouzeghoub, M.: Behavioral matchmaking for service retrieval. In: Proceedings of the IEEE International Conference on Web Services, pp. 145–152. IEEE Computer Society, Washington, DC (2006). doi:10.1109/ICWS.2006.37
32. Hack, M.: Decidability questions for petri nets. Ph.D. thesis, M.I.T. (1976)
33. Hoare, C.A.R.: A Model for Communicating Sequential Processes. Tech. rep., Oxford University Computing Laboratory (1980)
34. Jin, T., Wang, J., Wen, L.: Efficient retrieval of similar business process models based on structure. In: On the Move to Meaningful Internet Systems: OTM 2011, Lecture Notes in Computer Science, vol. 7044, pp. 56–63. Springer, Berlin (2011)
35. Jin, T., Wang, J., Wen, L.: Querying business process models based on semantics. In: Proceedings of the 16th International Conference on Database Systems for Advanced Applications: Part II, DASFAA'11, pp. 164–178. Springer, Berlin (2011)
36. Jin, T., Wang, J., Wen, L.: Efficient retrieval of similar workflow models based on behavior. In: Web Technologies and Applications, Lecture Notes in Computer Science, vol. 7235, pp. 677–684. Springer, Berlin (2012)
37. Jin, T., Wang, J., Wu, N., Rosa, M.L., ter Hofstede, A.H.M.: Efficient and accurate retrieval of business process models through indexing—(short paper). In: Meersman et al. [54], pp. 402–409
38. Jung, J.Y., Bae, J., Liu, L.: Hierarchical business process clustering. In: IEEE SCC (2), pp. 613–616. IEEE Computer Society (2008)
39. Kiepuszewski, B., Hofstede, A.H.M.T., van der Aalst, W.: Fundamentals of control flow in workflows. Acta Inform **39**, 143–209 (2002)
40. Knuth, D.E.: The Art of Computer Programming, vol. 3: Sorting and Searching, 2nd edn. Addison-Wesley, Reading, MA (1973)
41. Koschmider, A.: Ähnlichkeitsbasierte Modellierungsunterstützung für Geschäftsprozesse. Ph.D. thesis, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften (2007)
42. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity: a proper metric. In: Proceedings of the 9th International Conference on Business Process Management, BPM '11, pp. 166–181. Springer, Heidelberg (2011)
43. Kunze, M., Weske, M.: Metric trees for efficient similarity search in process model repositories. In: Proceedings of the 1st International Workshop on Process in the Large (IW-PL '10). Hoboken, NJ (2010)
44. Kunze, M., Weske, M.: Local behavior similarity. In: BPMDS 2012 and EMMSAD 2012, LNBIP, vol. 113, pp. 107–120. Springer (2012)
45. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R.M., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: A.P. Barros, A. Gal, E. Kindler (eds.) BPM, Lecture Notes in Computer Science, vol. 7481, pp. 319–334. Springer (2012)
46. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Doklady **10**, 707 (1966)
47. Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. In: Q. Li, S. Spaccapietra, E.S.K. Yu, A. Olivé (eds.) ER, Lecture Notes in Computer Science, vol. 5231, pp. 248–264. Springer (2008)
48. Lipkus, A.: A proof of the triangle inequality for the Tanimoto distance. J. Math. Chem. **26**, 263–265 (1999)
49. Lohmann, N., Verbeek, E., Dijkman, R.: Petri net transformations for business processes: a survey. In: Transactions on Petri Nets and Other Models of Concurrency, chap. 2, pp. 46–63. Springer, Berlin (2009)
50. Lu, R., Sadiq, S.: On the discovery of preferred work practice through business process variants. In: Proceedings of the 26th International Conference on Conceptual Modeling, ER'07, pp. 165–180. Springer, Berlin (2007)
51. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA (1999)
52. Markovic, I., Costa Pereira, A., Francisco, D., Mu noz, H.: Querying in Business Process Modeling, pp. 234–245 (2007)
53. Markovic, I., Pereira, A.C.: Towards a formal framework for reuse in business process modeling. In: Business Process Management Workshops, Lecture Notes in Computer Science, vol. 4928, pp. 484–495. Springer, Berlin (2007). doi:10.1007/978-3-540-78238-4-49. http://www.springerlink.com/content/xqg31444r0255660/
54. Meersman, R., Dillon, T.S., Herrero, P. (eds.): On the move to meaningful internet systems: OTM 2010—Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Hersonissos, Crete, Greece, October 25–29, 2010, Proceedings, Part I, Lecture Notes in Computer Science, vol. 6426. Springer (2010)
55. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Lecture Notes in Business Information Processing, vol. 6. Springer (2008)
56. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM **38**, 39–41 (1995). doi:10.1145/219717.219748
57. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P.: Matching and merging of statecharts specifications. In: Proceedings of the 29th International Conference on Software Engineering, ICSE '07, pp. 54–64. IEEE Computer Society, Washington, DC (2007). doi:10.1109/ICSE.2007.50
58. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer, New York (1985)
59. Reisig, W.: Petri Nets: An Introduction, Monographs in Theoretical Computer Science. An EATCS Series, vol. 4. Springer (1985)
60. Rosemann, M.: Potential pitfalls of process modeling: part B. Bus. Process Manag. J. **12**(3), 377–384 (2006). doi:10.1108/14637150610668024
61. Sakr, S., Al-Naymat, G.: Graph indexing and querying: a review. Int. J. Web Inf. Syst. 6(2), 101–120 (2010). doi:10.1108/17440081011053104. http://www.emeraldinsight.com
62. Sakr, S., Awad, A., Kunze, M.: Querying process models repositories by aggregated graph search. In: Proceedings of the 3rd International Workshop on Reuse in BPM, p. (to appear) (2012)

63. Shao, Q., Sun, P., Chen, Y.: Wise: A workflow information search engine. In: Proceedings of the 2009 IEEE International Conference on Data Engineering, pp. 1491–1494. IEEE Computer Society, Washington, DC (2009). doi:10.1109/ICDE.2009.89
64. Sokolsky, O., Kannan, S., Lee, I.: Simulation-based graph similarity. In: Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), pp. 426–440 (2006)
65. Song, L., Wang, J., Wen, L., Wang, W., Tan, S., Kong, H.: Querying process models based on the temporal relations between tasks. In: Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops, EDOCW '11, pp. 213–222. IEEE Computer Society, Washington, DC (2011) doi:10.1109/EDOCW.2011.12
66. Uba, R., Dumas, M., García-Bañuelos, L., Rosa, M.L.: Clone detection in repositories of business process models. In: S. Rinderle-Ma, F. Toumani, K. Wolf (eds.) BPM, Lecture Notes in Computer Science, vol. 6896, pp. 248–264. Springer (2011)
67. Valmari, A.: The state explosion problem. In: Petri Nets, Lecture Notes in Computer Science, vol. 1491, pp. 429–528. Springer (1996)
68. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets, pp. 407–426. Springer, London (1997)
69. van der Aalst, W.M.P.: The application of petri nets to workflow management. J. Circ. Syst. Comput. **8**(1), 21–66 (1998)
70. Vanhatalo, J., Völzer, H., Leymann, F., Moser, S.: Automatic workflow graph refactoring and completion. In: Proceedings of the 6th International Conference on Service-Oriented Computing, ICSOC '08, pp. 100–115. Springer, Berlin (2008)
71. Wang, J., He, T., Wen, L., Wu, N., Ter Hofstede, A.H.M., Su, J.: A behavioral similarity measure between labeled petri nets based on principal transition sequences. In: Proceedings of the 2010 International Conference on On the Move to Meaningful Internet Systems, vol. Part I, OTM'10, pp. 394–401. Springer, Berlin (2010)
72. Weidlich, M., Dijkman, R., Mendling, J.: The icop framework: identification of correspondences between process models. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering, CAiSE'10, pp. 483–498. Springer, Berlin (2010)
73. Weidlich, M., Elliger, F., Weske, M.: Generalised computation of behavioural profiles based on petri-net unfoldings. In: M. Bravetti, T. Bultan (eds.) WS-FM, Lecture Notes in Computer Science, vol. 6551, pp. 101–115. Springer (2010)
74. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Softw. Eng. **37**(3), 410–429 (2011)
75. Weidlich, M., van der Werf, J.M.E.M.: On profiles and footprints—relational semantics for petri nets. In: S. Haddad, L. Pomello (eds.) Petri Nets, Lecture Notes in Computer Science, vol. 7347, pp. 148–167. Springer (2012)
76. Yan, Z., Dijkman, R.M., Grefen, P.: Fast Business Process Similarity Search with Feature-Based Similarity Estimation. In: Meersman et al. [56], pp. 60–77
77. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach. Springer, Secaucus, NJ (2005)
78. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. Comput. Ind. **61**(5), 463–471 (2010)
79. Zloof, M.M.: Query by example. In: Proceedings of the May 19–22, 1975, National Computer Conference and Exposition, AFIPS '75, pp. 431–438. ACM, New York, NY (1975). doi:10.1145/1499949.1500034

Author Biographies

Matthias Kunze is a postdoctoral researcher at the Hasso Plattner Institute (HPI) of IT Systems Engineering at the University of Potsdam, Germany. He received his Masters degree from the University of Potsdam in 2009 and his Ph.D. in 2013. The core subject of his research is on methodologies to search efficiently in large process model repositories. He made contributions to process model similarity and querying, data structures and algorithms for indexing, and methods to facilitate the reuse of search results. His further research interests include event processing and correlation with process models, and process modeling methodologies. Matthias Kunze has published his results in several workshops and conferences.

Matthias Weidlich is a research fellow and adjunct lecturer at the Technion—The Israel Institute of Technology. He received his Ph.D. in Computer Science from the Hasso Plattner Institute (HPI), University of Potsdam, Germany, in 2011. His research focuses on processes as the basis of behavioral modeling and analysis with contributions related to process model matching, consistency analysis, process transformations, and compliance monitoring. Further, his research interests include data interoperability and reasoning in event-processing environments. His results appeared in journals, such as IEEE Transactions on Software Engineering, Information Systems, The Computer Journal, and Acta Informatica. He is on the editorial board of Elsevier's Information Systems and a member of ACM and IEEE.

Mathias Weske is chair of the business process technology research group at Hasso Plattner Institute of IT Systems Engineering at the University of Potsdam, Germany. His research interests include business process modeling, process choreographies, business process methodologies, and service-oriented computing. Dr. Weske has published twelve books and over 100 scientific papers in journals and conferences. He is on the steering committee of the BPM conference series, a member of ACM, IEEE, and GI. Dr. Weske has published a textbook on business process management, and he leads the BPM Academic Initiative.