Revealing Hidden Relations among Web Services Using Business Process Knowledge

Ahmed Awad and Mohammed AbuJarour

Hasso-Plattner-Institut, University of Potsdam, Germany
{firstname.lastname}@hpi.uni-potsdam.de

Abstract. The wide spread of Service-oriented Computing and Cloud Computing has been increasing the number of web services on the Web. This increasing number of web services complicates the task of service discovery, in particular because of lack of rich service descriptions. Relations among web services are usually used to enhance service discovery. Formal service descriptions, logs of service invocations, or service compositions are typically used to find such relations. However, using such sources of knowledge enables finding simple relations only. In a previous work, we proposed to use business processes (BPs) to refine relations among web services used in the configurations of these BPs. That approach was limited to web services directly consumed by a *single* business process. In this paper, we generalize that approach and aim at predicting rich relations among web services that were not directly used together in any process configuration yet. To achieve this goal, we take all individual business processes (from a business process repository) and their configurations over web services (from a service registry) in the form of so-called extended behavioral profiles. These disparate profiles are then merged so that a single global profile is derived. Based on the aggregated knowledge in this global profile, we reveal part of the unknown relations among web services that have not been used together yet. We validate our approach through a set of experiments on a collection of business processes from SAP reference model.

Keywords: service discovery, behavioral profiles, business process configurations.

1 Introduction: Relations among Web Services

The increasing number of web services is one of the main factors that make *service discovery* a major challenge in Service-oriented Computing [3,4,11]. Finding relations among web services has been used to handle this challenge. The traditional techniques that have been proposed to achieve this task are input-output matching of web services using their technical service descriptions [8], semantic approaches [13], using compositions of web services [5], and using consumer-consumer similarity [15] (Sec. 2). The main goal of these approaches is finding any type of relations between web services, i.e., their outcome is whether two web services are related or not. No further refinement of these found relations is suggested. Additionally, these approaches are not able to find *indirect* relations among web services, because they use knowledge about web services that are used together only. Relations between web services that are not used together remain missing.

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 235–249, 2011.

Missing relations between web services do not necessarily indicate their independence. Several reasons can lead to such hidden relations, such as lack of knowledge about web services and their functionalities, multiple web services with equivalent functionalities, and non-functional requirements (e.g., price, quality). We consider such missing relations as *hidden* ones and aim at revealing (part of) them. One approach of revealing such hidden relations is using knowledge concealed in the configurations of business processes that use these web services. Each configuration is considered as an identifier for its tasks and its web services. Multiple tasks that have different labels are similar if they are bound with the same web service. Similarly, web services that have different names are considered similar if they are bound with tasks that share the same label.

In our previous work [2], we introduced an approach to discover advanced relations among web services in the form of linkage patterns using business process knowledge represented as behavioral profiles [19]. Linkage patterns are generated among web services that are used in the same business process. As different consumers use web services in multiple business processes with different relations among them, multiple configurations over the same set of web services appear. These configurations are *local* to each individual process. In this paper, we develop an approach to derive a *global* behavioral profile over the entire set of web services in a service registry and reveal *hidden* relations among web services – that have never been used together by investigating indirect relationships among them – within this global profile.
The main contributions of this paper are:

- An approach to merge behavioral profiles of business processes into a global behavioral profile.
- An approach to reveal hidden relations among web services that have not been used together, yet.

To validate our approach, we use a set of business processes from the SAP reference model [7]. These models represent possibilities to configure SAP R/3 ERP systems. Thus, it is analogous to business process configurations over a service landscape.

The rest of the paper is organized as follows: We summarize related work in Sec. 2. After that, we give preliminary concepts and definitions in Sec. 3. Then, we introduce our definition of extended behavioral profiles in Sec. 4. In Sec. 5, we show our approach to derive a global behavioral profile and resolve its unknown relations. We present a set of experiments to evaluate our approach in Sec. 6. In Sec. 7, we summarize this paper and show our future work.

2 Related Work

Finding relations among web services has been considered by several researchers in the community. Approaches that tackle this problem can be grouped roughly in four groups:

- **Input/output matching approaches:** These approaches match inputs and outputs of operations of web services to find relations among them [8]. They assume the existence of complete, rich, and correct service descriptions, e.g., `WSDL`. However, such assumption is not always realistic [10]. Additionally, experiments have shown that `WSDL` descriptions only are not sufficient [17]. Additionally, these approaches may lead to unrealistic/unusable relations, and misses relations between web services with *manual* tasks in between. The main goal of these approaches is to investigate composability among web services [14,16].
- **Semantic approaches:** These approaches apply Artificial Intelligence planning techniques to find valid compositions of web services [12,13]. They are based on the assumption that web services are described formally using ontologies, such as `OWL-S`, `WSMO`, etc. In practice, semantic web services are not common [6]. Our approach does not necessarily require the existence of such formal descriptions. However, their existence could enable it to find additional relations.
- **Service compositions-based approaches:** These approaches are based on the idea that web services used in a service composition are related [5,20], i.e., they give only binary decisions. However, these approaches are not able to specify the type of relations between web services based on their usage in service compositions. These approaches depend on the co-occurrence of web services in service compositions to decide that there is a relation among them. On the other hand, Eshuis et al. [9] discover related web services based on structural matches of BPEL processes. To find related composite services with respect to a query, the behavioral relations among component services are compared to those in the query and ranked according to some heuristics. Compared to our approach, Eshuis et al.'s approach is unable to reveal hidden relations among web services that were never used in the same process model.
- **Consumer-consumer similarity approaches:** These approaches use the idea that similar service consumers usually use the same web services [15]. However, they assume the ability to track web services used by service consumers. This setting is not the typical one in practice. Moreover, it has been shown that similar service consumers do not necessarily use the same web services.

3 Preliminaries

A business process model consists of a set of tasks and theie execution ordering. Such execution ordering specifies generally which tasks are executed in sequence, concurrent or alternative to each other. Tasks can be *manual* or *service* tasks. The former are performed by humans, whereas the latter are executed by web services. Process models have special types of tasks that determine the start and end points of a process instance, respectively. Moreover, explicit control routing nodes are used to describe the above mentioned execution ordering. Thus, we represent a process model as a graph whose nodes are typed. The definition of a process model is given in Definition 1.

Definition 1 (Process Model). *A process model P is a connected, directed graph* (N, E) *where* $N = T \cup G \cup \{n_{in}, n_{out}\}$ *is a finite set of nodes, with Tasks T, Gateways G, and exactly one start and end event* n_{in} *and* n_{out}*, and* $E \subseteq (N \setminus \{n_{out}\}) \times (N \setminus \{n_{in}\})$ *is a set of edges connecting the nodes.*

A behavioral profile represents an abstract description of a business process, where it identifies a behavioral relationship between any pair of nodes within that process [19]. This relationship can be: (1) strict order $\rightsquigarrow$, (2) concurrent $\|$, (3) exclusive # or (4) inverse order $\leftsquigarrow$. The formal definition of behavioral profiles is given in Definition 2.

Definition 2 (Behavioral Profile). *Let N be the set of nodes within a business process model. The* behavioral profile *of a business process model is a function* $bhp_{bp} : N \times N \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#\}$ *that assigns a behavioral property, strict order, inverse order, parallel, or exclusive, between each pair of nodes within the business process model.*

If two tasks a, b appear in strict order within a business process x, $bhp_x(a, b) = \rightsquigarrow$, then task a executes before task b. Similarly, if two tasks are concurrent then they can be executed in any order. Exclusiveness means that at most one of the two tasks can execute within a process instance. To derive useful behavioral properties between tasks of a BP, we remove cyclic edges from such BPs because their existence makes all connected tasks concurrent.

Transforming business process models into executable processes is done through a *configuration* step. In this step, business engineers assign web services to *service* tasks in the considered model. This assignment represents a service discovery and selection task. The formal definition of BP configuration is given in Definition 3.

Definition 3 (BP Configuration). *A service registry usually contains a collection of web services*[1]*. The* configuration *of a business process model – containing tasks T – is a function* $conf : T \rightarrow WS_i$ *that assigns a web service* (WS_i) *for each* service *task in that business process model.*

Based on these behavioral profiles, we derive linkage patterns among web services [2]. Therefore, deriving a *consistent* global behavioral profile is a key to avoid inconsistencies among linkage patterns and to predict useful relations among web services that have not been used together yet, i.e., do not appear in the same BP.

4 The Extended Behavioral Profile

Revealing hidden relations among web services requires a global behavioral profile, where all services in the considered registry are involved. A global profile is the result of *merging* all individual behavioral profiles of available business processes. Merging two relations from two profiles results in *unknown* relations between web services that do not appear together in one business process. Moreover, this merging step might result in *contradicting* relations, e.g., merging $a\#_x b$ and $a \rightsquigarrow_y b$. Therefore, the four basic

[1] For simplicity, we assume that each web service contains a single operation.

behavioral relations of the original behavioral profile in Definition 2 are not sufficient. We *extend* the four basic relations to capture such situations when merging individual profiles by introducing two additional relations: unknown (?) and contradicts (※). These two relations do not appear on the level of individual raw profiles. They appear only when profiles are merged as we show in Sec. 5. Additionally, we record the *distance* between tasks bound to web services in the process configuration. This distance is used in the derived linkage patterns among web services to rank services during service discovery. We obtain this distance by counting the edges on the shortest path between the nodes representing the tasks in the process graph.

In this section, we present the formal notion of *extended* behavioral profiles (Sec. 4.1). In Sec. 4.2, we introduce a business process with its extended behavioral profile that is used as a running example in the rest of this paper.

4.1 Formal Model

The original definition of behavioral profiles is concerned with behavioral relations among tasks within a business process. However, in our approach, we are interested in discovering relations among web services used to configure such business processes.

Definition 4 (Extended Behavioral Profile). *Let $\mathcal{W}$ be the set of web services within a service registry. The* extended behavioral profile *of web services in $\mathcal{W}$ is a function $xbhp : \mathcal{W} \times \mathcal{W} \rightarrow \mathcal{P}(\{\rightsquigarrow, \leftsquigarrow, \|, \#, ?, ※\} \times \mathbb{N})$ that assigns a set of pairs of a behavioral property (strict order, inverse order, parallel, exclusive, unknown or contradicts) and a distance between each pair of web services within the service registry.*[2]

Comparing Definition 4 of the extended behavioral profile with Definition 2, we notice that the behavioral relations are leveraged from the level of tasks within individual process models (configurations) to the level of web services within the service registry. Moreover, the extended profile records the distance between web services consumed within an individual profile. This distance is greater than zero if the behavioral relation is either $\rightsquigarrow$ or $\leftsquigarrow$ and zero otherwise. Finally, multiple behavioral properties can exist between two web services in the global behavioral profile (Sec. 5.1).

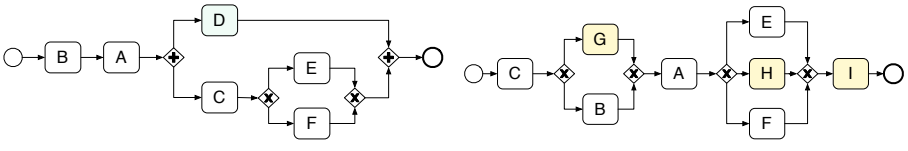
An individual behavioral profile (Definition 2) of a process can be turned into an extended profile by adding all web services in the service registry to the services consumed by that process where their behavioral relations are set to unknown. For simplicity, we ignore these unknown relations for input behavioral profiles.

Definition 5 (Projections over an Extended Behavioral Profile). *Let $\mathcal{W}$ be the set of web services within a serivce registry and let x be an extended behavioral profile. The function $rel_x : \mathcal{W} \times \mathcal{W} \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#, ?, ※\}$ projects the behavioral relation between two web services a and b in the registry with respect to profile x. Similarly $dist_x : \mathcal{W} \times \mathcal{W} \rightarrow \mathbb{N}$ projects the distance between the two services with profile x. For simplicity, we express $rel_x(a, b) = \{*\}$ as $a *_x b$ in the rest of the paper.*

[2] For simplicity, we assume that each web service has only one operation.

4.2 Running Example

In Fig. 1, we introduce two anonymized business processes from the SAP reference model that are used as a running example through this paper. We use the labels $A - I$ as identifiers for *both* tasks and web services. The common anonymized labels between both business processes indicate using the same service in their configuration. BP_1 has 6 tasks where only task D is not a common task with BP_2. On the other hand, BP_2 has 8 tasks among which 3 tasks are not common with BP_1, namely G, H, and I.



(a) Business Process BP_1 (b) Business Process BP_2

Fig. 1. Two anonymized Business Processes from SAP reference model

The extended behavioral profile of BP_1 is shown in Table 1 and that of BP_2 can be generated similarly, we omit it due to space limitations. It is worth mentioning that both BPs are configured such that each task is bound with a web service to execute it. According to Table 1, $xbhp_{BP_1}(E, F) = \{(\#, 0)\}$ and $xbhp_{BP_1}(A, D) = \{(\rightsquigarrow, 2)\}$. Relations that are not shown in this profile are implicitly *unknown*, e.g., $xbhp_{BP_1}(A, G) = \{(?, 0)\}$.

Table 1. The *extended* behavioral profile of BP_1 shown in Fig. 1(a)

	A	B	C	D	E	F
A	(‖, 0)	(⇜, 1)	(⇝, 2)	(⇝, 2)	(⇝, 4)	(⇝, 4)
B	(⇝, 1)	(‖, 0)	(⇝, 3)	(⇝, 3)	(⇝, 5)	(⇝, 5)
C	(⇜, 2)	(⇜, 3)	(‖, 0)	(‖, 0)	(⇝, 2)	(⇝, 2)
D	(⇜, 2)	(⇜, 3)	(‖, 0)	(‖, 0)	(‖, 0)	(‖, 0)
E	(⇜, 4)	(⇜, 5)	(⇜, 2)	(‖, 0)	(‖, 0)	(# , 0)
F	(⇜, 4)	(⇜, 5)	(⇜, 2)	(‖, 0)	(# , 0)	(‖, 0)

5 Deriving a Global Behavioral Profile

Knowledge about relations among web services is usually scattered in disparate profiles of business processes that are configured to use these web services. Collecting this knowledge into one single profile is essential to reveal hidden relations among these web services. We call the result of this step a *global* behavioral profile. We describe our approach to derive this global profile from individual profiles in Sec. 5.1. The global profile might include unknown or contradicting relations among some pairs of web

services. We inspect the gained knowledge in the global profile to *predict* possible resolutions for its unknown relations. We describe our approach to resolve unknown relations in the global profile in Sec. 5.2. Both steps, i.e., deriving a global profile and predicting unknown relations are of iterative nature. That is, at the point that a new process configuration is available, this new profile is merged with the global profile, to obtain a new global profile, and the prediction of unknown relations is rerun.

5.1 Merging Individual Behavioral Profiles

Given a set of behavioral profiles, we aim at deriving a global profile that contains pairwise relations between all used web services. We achieve this deriving by merging all individual profiles iteratively in a pairwise manner. The result of each merging iteration is an intermediate profile that is merged with another profile. This step is repeated until all individual profiles are incorporated. Merging individual profiles might result in unknown or contradicting relations among web services. Unknown relations appear between web services that are not used together in the same business process. Whereas contradicting relations appear due to *conflicting* relations in source profiles. For instance, the relations $(a\#_x b)$ (i.e., a and b are exclusive in profile x) and $(a \rightsquigarrow_y b)$ (i.e., a precedes b in profile y) includes at least one incorrect relation. Exclusiveness usually means that web services do similar or complementary jobs [2]. Currently, we propagate such conflicts to the resulting intermediate profile by adding two relations $(a \between_z b)$ and $(b \between_z a)$ that represent a contradiction to the resulting intermediate profile z.

We merge two relations between web services a and b that appear in both input profiles x and y into the global profile t according the set of rules that is summarized in Table 2. These rules can be grouped as follows:

1. Merging $(a *_x b)$ and $(a *_y b)$ gives $(a *_t b)$, where $*$ is the same type of relation.
2. Merging $(a \reverse{\rightsquigarrow}_x b)$ and $(a \rightsquigarrow_y b)$ gives $(a \parallel_t b)$ or $(a \between_t b)$.
3. Merging $(a *_x b)$ and $(a \bullet_y b)$ gives $(a \parallel_t b)$ or $(a \between_t b)$, where $* \in \{\rightsquigarrow, \reverse{\rightsquigarrow}\}$ and $\bullet = \parallel$.
4. Merging $(a *_x b)$ and $(a\#_y b)$ gives $(a\#_t b)$ if $* = \#$, and $a \between_t b$ otherwise.
5. Merging $(a?_x b)$ and $(a *_y b)$ gives $(a *_t b)$, where $*$ is a basic relation.
6. Merging $(a \between_x b)$ and $(a *_y b)$ gives $a \between_t b$.

Some merging rules are non-deterministic, i.e., produce multiple alternatives (See Table 2). For instance, merging $(a \rightsquigarrow_x b)$ and $(a \reverse{\rightsquigarrow}_y b)$ gives two options: $(a \parallel_t b)$ and $(a \between_t b)$. Parallelism means that there is no dependency between a and b and they can be used in any order. On the other hand, a dependency between a and b means that either profile x or y is incorrect, where it includes a data anomaly, e.g., missing data [18]. In this case, we conclude that there is a contradiction $(a \between_t b)$. To resolve such non-determinism, a human intervention is needed, which is out of scope of this paper.

The second component in the extended behavioral profile (besides the relation's type) is *distance* between services. This distance between two services in an intermediate profile is calculated as the shortest distance in the corresponding profiles unless one of both distances is zero, i.e., # or ||.

Table 2. Merging relationships from two profiles x and y into an intermediate profile t

Profile	$a \rightsquigarrow_x b$	$a \leftsquigarrow_x b$	$a \parallel_x b$	$a\#_x b$	$a?_x b$	$a\between_x b$
$a \rightsquigarrow_y b$	$a \rightsquigarrow_t b$	$a \parallel_t b$ or $a\between_t b$	$a \parallel_t b$ or $a\between_t b$	$a\between_t b$	$a \rightsquigarrow_t b$	$a\between_t b$
$a \leftsquigarrow_y b$	$a \parallel_t b$ or $a\between_t b$	$a \leftsquigarrow_t b$	$a \parallel_t b$ or $a\between_t b$	$a \between_t b$	$a \leftsquigarrow_t b$	$a\between_t b$
$a \parallel_y b$	$a \parallel_t b$ or $a\between_t b$	$a \parallel_t b$ or $a\between_t b$	$a \parallel_t b$	$a\between_t b$	$a \parallel_t b$	$a\between_t b$
$a\#_y b$	$a\between_t b$	$a\between_t b$	$a\between_t b$	$a\#_t b$	$a\#_t b$	$a\between_t b$
$a?_y b$	$a \rightsquigarrow_t b$	$a \leftsquigarrow_t b$	$a \parallel_t b$	$a\#_t b$	$a?_t b$	$a\between_t b$
$a\between_y b$	$a\between_t b$	$a\between_t b$	$a\between_t b$	$a\between_t b$	$a\between_t b$	$a\between_t b$

Example. Consider that the two processes from Fig. 1 are the only individual profiles in our knowledge base. By applying our merging rules shown in Table 2, we get the global profile shown in Table 3. This global profile has 9 web services that represent the union of all services in its source profiles, i.e., BP_1 and BP_2. For instance, merging relations ($\leftsquigarrow$, 1) and ($\leftsquigarrow$, 2) between web services A and B from BP_1 and BP_2, receptively, gives the relation ($\leftsquigarrow$, 1) in the global profile. The distance of the relation in the global profile is the minimum distance from input relations. Some merging rules produce multiple alternatives. For instance, A and C has the relation ($\rightsquigarrow$, 2) and ($\leftsquigarrow$, 4) in BP_1 and BP_2, respectively. Merging both relations gives two alternatives in the global profile between A and C: ($\parallel$, 0) and ($\between$, 0). The remaining relations can be derived in the same way. Merging extended behavioral profiles of BPs that do *not* have the same exact set of web services results in unknown relations between web services that do not appear in the same BP. For instance, relations between D from one side and G, H, and I on the other side in the global profile. In the sequel, we aim at using the knowledge gained from merging both profiles to reveal such unknown relations.

Table 3. Merging profiles of BP_1 and BP_2 (Fig. 1(a) &1(b)) in one global profile

	A	B	C	D	E	F	G	H	I
A	($\parallel$, 0)	($\leftsquigarrow$, 1)	($\parallel$, 0) ($\between$, 0)	($\rightsquigarrow$, 2)	($\rightsquigarrow$, 2)	($\rightsquigarrow$, 2)	($\leftsquigarrow$, 2)	($\rightsquigarrow$, 2)	($\rightsquigarrow$, 8)
B	($\rightsquigarrow$, 1)	($\parallel$, 0)	($\parallel$, 0) ($\between$, 0)	($\rightsquigarrow$, 3)	($\rightsquigarrow$, 4)	($\rightsquigarrow$, 4)	(# , 0)	($\rightsquigarrow$, 4)	($\rightsquigarrow$, 10)
C	($\parallel$, 0) ($\between$, 0)	($\parallel$, 0) ($\between$, 0)	($\parallel$, 0)	($\parallel$, 0)	($\rightsquigarrow$, 2)	($\rightsquigarrow$, 2)	($\rightsquigarrow$, 2)	($\rightsquigarrow$, 8)	($\rightsquigarrow$, 14)
D	($\leftsquigarrow$, 2)	($\leftsquigarrow$, 3)	($\parallel$, 0)	($\parallel$, 0)	($\parallel$, 0)	($\parallel$, 0)	**(?, 0)**	**(?, 0)**	**(?, 0)**
E	($\leftsquigarrow$, 2)	($\leftsquigarrow$, 4)	($\leftsquigarrow$, 2)	($\parallel$, 0)	($\parallel$, 0)	(# , 0)	($\leftsquigarrow$, 4)	(# , 0)	($\rightsquigarrow$, 2)
F	($\leftsquigarrow$, 2)	($\leftsquigarrow$, 4)	($\leftsquigarrow$, 2)	($\parallel$, 0)	(# , 0)	($\parallel$, 0)	($\leftsquigarrow$, 4)	(# , 0)	($\rightsquigarrow$, 2)
G	($\rightsquigarrow$, 2)	(# , 0)	($\leftsquigarrow$, 2)	**(?, 0)**	($\rightsquigarrow$, 4)	($\rightsquigarrow$, 4)	($\parallel$, 0)	($\rightsquigarrow$, 4)	($\rightsquigarrow$, 10)
H	($\leftsquigarrow$, 2)	($\leftsquigarrow$, 4)	($\leftsquigarrow$, 8)	**(?, 0)**	(# , 0)	(# , 0)	($\leftsquigarrow$, 4)	($\parallel$, 0)	($\rightsquigarrow$, 2)
I	($\leftsquigarrow$, 8)	($\leftsquigarrow$, 10)	($\leftsquigarrow$, 14)	**(?, 0)**	($\leftsquigarrow$, 2)	($\leftsquigarrow$, 2)	($\leftsquigarrow$, 10)	($\leftsquigarrow$, 2)	($\parallel$, 0)

5.2 Predicting Potential Resolutions for Unknown Relations ($a?b$)

Merging two profiles that do not have the same set of web services results in a global profile with unknown relations among web services that do not appear in both source profiles. In this section, we describe our approach to reveal such *unknown* relations by predicting potential resolutions for them.

We predict potential resolutions for the unknown relation between web services a and b in the global profile g with the help of a common service between them, e.g., c. Our goal is to resolve the relation $(a?_g b)$ into $(a \leftsquigarrow_g b)$, $(a \rightsquigarrow_g b)$, $(a \parallel_g b)$, or $(a\#_g b)$ by investigating the relations between a and c on one hand and between b and c on the other hand. We select a common service c such that we can derive useful information from its relations with the considered services. For instance, selecting c such that $(a \divideontimes_g c)$ is not of value. Therefore, the common service c has to be in one of the basic four relations with both a and b. Furthermore, the predicted relation has to be consistent with existing relations in the global profile. Finding a useful resolution for unknown relations depends on the used knowledge, therefore it is not always possible to predict such a resolution. In such cases, the unknown relation between a and b $(a?_g b)$ in the global profile g remains and a human expert is informed about the situation to find a resolution manually.

We predict potential resolutions for each unknown relation between web services a and b in the global profile g – i.e., $(a?_g b)$ – using a common service, c, according to the set of rules that is summarized in Table 4. These rules can be grouped together as follows:

1. Order-order: resolving $(a *_g c)$ and $(b *_g c)$ gives $(a \rightsquigarrow_g b)$, $(a \leftsquigarrow_g b)$, $(a \parallel_g b)$, and $(a\#_g b)$, where $* = \rightsquigarrow$ or $* = \leftsquigarrow$. Each of these predicted relations still preserves the existing relations $(a \rightsquigarrow c)$ and $(b \rightsquigarrow c)$ or $(a \leftsquigarrow c)$ and $(b \leftsquigarrow c)$.
2. Transitivity: resolving $(a \leftsquigarrow_g c)$ and $(b \rightsquigarrow_g c)$ gives $(a \leftsquigarrow_g b)$. Any other relation, e.g., $(a \rightsquigarrow b)$, does not preserve the existing relation between a,b on the one hand and c on the other hand. For instance, $(a \rightsquigarrow b)$ means that b executes *before* a, that contradicts $(a \leftsquigarrow c)$. Similarly, we cannot deduce that $(a\#b)$ as it contradicts $(b \rightsquigarrow c)$, since that implies either $(c \leftsquigarrow b)$ or $(c\#b)$, which is not the case.
3. Branch-order: resolving $(a *_g c)$ and $(b \bullet_g c)$ gives $(b *_g a)$ and $(a \bullet_g b)$, where $* \in \{\leftsquigarrow, \rightsquigarrow\}$ and $\bullet \in \{\|, \#\}$.
4. Similar Branch-Branch: resolving $(a *_g c)$ and $(b *_g c)$ gives $(a *_g b)$, $(a \rightsquigarrow_g b)$, and $(a \leftsquigarrow_g b)$, where $* = \|$ or $* = \#$.
5. Different Branch-Branch: resolving $(a *_g c)$ and $(b \bullet_g c)$ gives $(a *_g b)$ and $a \bullet_g b$, where $* \in \{\|, \#\}$ and $\bullet \in \{\|, \#\}$, and $\bullet \neq *$.

Distances of the predicted $\rightsquigarrow$ and $\leftsquigarrow$ relations in the global profile are calculated according to the functions shown in Table 5. Distances are used to rank relevant web services during service discovery [2]. Additionally, we use them to prune possible resolutions. For some cases, the new distance is the absolute value of the difference of two distance. As an example, consider the case where we have $(a \rightsquigarrow c)$ and $(b \rightsquigarrow c)$. According to Table 4, all four basic relations are valid resolutions. For the predicted $(a \parallel b)$ and $(a\#b)$ we set distance to zero. However, for the two remaining cases, i.e. $(a \rightsquigarrow b)$ and $(a \leftsquigarrow b)$, the distance is the absolute value of the difference in input distances.

Table 4. Resolving the unknown relation $a?_g b$ via a common service c

Relation	$a \rightsquigarrow c$	$a \lessdot c$	$a \parallel c$	$a\#c$
$b \rightsquigarrow c$	$a \rightsquigarrow b$ $a \lessdot b$ $a \parallel b$ $a\#b$	$a \lessdot b$	$a \parallel b$ $a \lessdot b$	$a\#b$ $a \lessdot b$
$b \lessdot c$	$a \rightsquigarrow b$	$a \lessdot b$ $a \rightsquigarrow b$ $a \parallel b$ $a\#b$	$a \parallel b$ $a \rightsquigarrow b$	$a\#b$ $a \rightsquigarrow b$
$b \parallel c$	$a \parallel b$ $a \rightsquigarrow b$	$a \parallel b$ $a \lessdot b$	$a \parallel b$ $a \rightsquigarrow b$ $a \lessdot b$	$a \parallel b$ $a\#b$
$b\#c$	$a\#b$ $a \rightsquigarrow b$	$a\#b$ $a \lessdot b$	$a \parallel b$ $a\#b$	$a\#b$ $a \rightsquigarrow b$ $a \lessdot b$

When we have no information to calculate the distance, we set it to infinity, e.g., the case of $(a \parallel c)$ and $(b \parallel c)$. For the cases where there is no order in the predicted relation between a and b, we express this using N/A in the table.

We calculate the distance of the predicted $\rightsquigarrow$ and $\lessdot$ relations in the global profile according to these rules:

1. Order-order: The distance is defined as $|diff()|$ between distances of input relations.
2. Transitivity: The distance is defined as $sum()$ of distances of input relations.
3. Branch-order: The distance is defined as distance between web services b and c.
4. Similar Branch-Branch: The distance is defined as ∞, i.e., an artificial value.
5. Different Branch-Branch: The distance is not applicable, i.e., N/A.

Table 5. Distances of the predicted $a?b$ via a common service c

Relation	$a \rightsquigarrow c$	$a \lessdot c$	$a \parallel c$	$a\#c$
$b \rightsquigarrow c$	$\|diff()\|$	$sum()$	$dist(b,c)$	$dist(b,c)$
$b \lessdot c$	$sum()$	$\|diff()\|$	$dist(b,c)$	$dist(b,c)$
$b \parallel c$	$dist(a,c)$	$dist(a,c)$	∞	N/A
$b\#c$	$dist(a,c)$	$dist(a,c)$	N/A	∞

According to our rules of resolution shown in Table 4, possible resolutions to an unknown relation $a?b$ can include both $a \rightsquigarrow b$ and $a \lessdot b$. We use the distance information to prune one or both of these resolutions according to the following rules. Consider two relations $(a *_x b)$ and $(b \bullet_y c)$ with distances d_x and d_y, respectively, where $*$ and $\bullet$ are either $\rightsquigarrow$ or $\lessdot$, and Δd is defined as $d_x - d_y$, we identify three cases:

- $\Delta d = 0$: The unknown relation $(a?b)$ cannot be predicted to $(a \rightsquigarrow b)$ or $(a \leftsquigarrow b)$.
- $\Delta d > 0$: The unknown relation $(a?b)$ can be predicted to $(a \rightsquigarrow b)$, but not to $(a \leftsquigarrow b)$.
- $\Delta d < 0$: The unknown relation $(a?b)$ can be predicted to $(a \leftsquigarrow b)$, but not to $(a \rightsquigarrow b)$.

Table 4 shows possible resolutions of $a?b$ using one common service c. However, a and b might have a set of common services, which includes services that have useful behavioral relations ($\rightsquigarrow$,$\leftsquigarrow$,||, or #) with both a and b. We use each element in this set to predict the unknown relation between a, b according to the rules in Table 4. After that, we intersect all possible resolutions deduced from each element in that set. The result of this intersection is then used as a resolution to that unknown relation between a and b. If this intersection gives an empty set (e.g., due to contradictions), we are unable to predict resolutions for $a?b$. These steps are shown in Algorithm 1.

Algorithm 1. Predicting unknown relations in the global profile

Input: g the global profile
Output: g' the global profile with some unknown relations revealed

```
pred ← ∅
for all a?b ∈ g do
    CT ← getCommonTasks(a, b)
    for all c ∈ CT do
        ac ← rel_g(a, c)
        bc ← rel_g(b, c)
        tmp ← predictRelaton(ac, bc) {According to Tables 4 and 5}
        if pred = ∅ then
            pred ← tmp
        else
            pred ← intersect(pred, temp)
            if pred = ∅ then
                break
            end if
        end if
    end for
    g ← merge(g, pred) {According to Table 2}
end for
g' ← g
return g'
```

Example. In Table 3, we show the global profile that we get by merging the extended global profiles of BP_1 (Fig. 1(a)) and BP_2 (Fig. 1(b)). That global profile has three unknown relations between service D on the one hand and services G, H, and I on the other hand, because these services are not used in the same BP. However, BP_1 and BP_2 have other common web services, e.g., A, B and C. We use such common services to predict resolutions for (part of) these three unknown relations.

To predict ($D?G$), we select the set of common tasks among them. In this example, this set is {A, B, C, E, F}. Because ($D \leftsquigarrow A$) and ($G \rightsquigarrow A$), we deduce that ($D \leftsquigarrow G$) according to the transitivity rule. Similarly, we deduce all potential relations between D and G using their common services as shown in Table 6. The intersection of these alternatives is ϕ, i.e., there is no common relation among potential relations. Therefore, the relation between D and G in the global profile remains unknown.

We follow the same steps to predict the relation ($D?H$). The set of common tasks is the same. Intersecting all potential relations between D and H gives the new relation ($D \parallel H$). Again, the same set of common tasks is used to reveal the ($D?I$). In this case, the intersection of all potential relations between these tasks gives two alternatives: ($D \parallel I$) and ($D \rightsquigarrow I$). The distance in the new strict order relation is the minimum distance between I and the common tasks. In this case, the distance is 2.

6 Experiments and Evaluation

In this section, we show a set of experiments that we did to evaluate our approach of predicting potential relations among web services using business process knowledge. We use a set of business processes from the SAP reference model [7], because these models represent possibilities to configure SAP R/3 ERP systems. Thus, it is analogous to business process configurations over a service landscape. We use 18 BPs with related missions from the SAP reference model. In particular, they are concerned with purchase order/requisition processing. These processes include 146 tasks in total. On average, each BP has about 8 tasks. Among the 146 tasks, 81 tasks are distinct, i.e., bound (configured) with distinct web services. We did this configurations manually and verified the results manually, as well. We analyzed the labels of the tasks and decided which labels (tasks) that can be bound to the same web service. Additionally, we had to manually restructure the models to have a single start and a single end nodes so that the behavioral profile calculation algorithm can be applied to them. Moreover, we excluded loops to obtain useful behavioral relations among tasks. A loop yields relations among all nodes within that loop concurrent.

The baseline approach is predicting relations among tasks of BPs *without* using their configurations information, i.e., only identical labels of tasks in different BPs are considered similar. Following this approach, we are able to predict resolutions for (54.8%) of all unknown relations in the generated global behavioral profile. The ratio of resolved relations using labels of tasks only depends considerably on the degree of similarity and cohesion among labels. Using the configurations of these BPs where semantically

Table 6. Possible relations between services D & G via common services {A, B, C, E, F}

Common Task	**A**	**B**	**C**	**E**	**F**
Relation with D	D $\leftsquigarrow$ A	D $\leftsquigarrow$ B	D $\parallel$ C	D $\parallel$ E	D $\parallel$ F
Relation with G	G $\rightsquigarrow$ A	G # B	G $\leftsquigarrow$ C	G $\rightsquigarrow$ E	G $\rightsquigarrow$ F
Deduced Relation	D $\leftsquigarrow$ G	D $\leftsquigarrow$ G D # G	D $\rightsquigarrow$ G D $\parallel$ G	D $\leftsquigarrow$ G D $\parallel$ G	D $\leftsquigarrow$ G D $\parallel$ G

Table 7. Revealing hidden relations in the global profile of BP_1 and BP_2

	A	B	C	D	E	F	G	H	I
A	(‖, 0)	(⇜, 1)	(‖, 0) (※, 0)	(⇝, 2)	(⇝, 2)	(⇝, 2)	(⇜, 2)	(⇝, 2)	(⇝, 8)
B	(⇝, 1)	(‖, 0)	(‖, 0) (※, 0)	(⇝, 3)	(⇝, 4)	(⇝, 4)	(# , 0)	(⇝, 4)	(⇝, 10)
C	(‖, 0) (※, 0)	(‖, 0) (※, 0)	(‖, 0)	(‖, 0)	(⇝, 2)	(⇝, 2)	(⇝, 2)	(⇝, 8)	(⇝, 14)
D	(⇜, 2)	(⇜, 3)	(‖, 0)	(‖, 0)	(‖, 0)	(‖, 0)	**(?, 0)**	**(‖, 0)**	**(‖, 0) (⇝, 2)**
E	(⇜, 2)	(⇜, 4)	(⇜, 2)	(‖, 0)	(‖, 0)	(# , 0)	(⇜, 4)	(# , 0)	(⇝, 2)
F	(⇜, 2)	(⇜, 4)	(⇜, 2)	(‖, 0)	(# , 0)	(‖, 0)	(⇜, 4)	(# , 0)	(⇝, 2)
G	(⇝, 2)	(# , 0)	(⇜, 2)	**(?, 0)**	(⇝, 4)	(⇝, 4)	(‖, 0)	(⇝, 4)	(⇝, 10)
H	(⇜, 2)	(⇜, 4)	(⇜, 8)	**(‖, 0)**	(# , 0)	(# , 0)	(⇜, 4)	(‖, 0)	(⇝, 2)
I	(⇜, 8)	(⇜, 10)	(⇜, 14)	**(‖, 0) (⇜, 2)**	(⇜, 2)	(⇜, 2)	(⇜, 10)	(⇜, 2)	(‖, 0)

similar tasks are bound to a single web services, we are able to predict resolutions for around (72%) of all unknown relations among tasks used in our experiments.

We are able to reveal different types of relations among web services. In Table 8, we show the ratio of each type of relations with respect to the total number of relations in source raw profiles, their derived global profile, and after revealing part of the hidden relations in that global profile. Note that percentages in this table are local to each column. The total number of relations is not the same in global profile and revealed global as an unknown relation can be predicted in multiple resolutions.The majority of relations in the revealed global profile are parallel (34%). Additional knowledge about such tasks and their bound web services can be used to resolve such relations in more concrete ones. This further resolution is part of our future work. Conflicting relations appear due to inaccurate configurations of BPs or due to lack of sufficient knowledge about tasks and web services. Unknown relations are still in the global profile even after applying our resolutions approach. Either the used knowledge is not sufficient to reveal such relations or there are no such useful relations. For instance, a music web service and a web service for Gene analysis.

7 Discussion

In this paper we introduced an approach to reveal hidden relations among web services by exploiting process configurations over these services. The relations we address are of four basic types; strict order, inverse order, exclusive, and concurrent. To reveal such relations, we extended the notion of behavioral profiles by adding two additional relations, namely, contradicts and unknown, and distance between activity nodes, within a process, that are bound to the web services under investigation. In practice, several process configurations exist with an organization. Therefore, we had to merge these individual profiles into a single global profile. After that, unknown relations within the

Table 8. Types of relations and their ratios in raw profiles, derived global profile, and resolved profile

Type	Raw Processes	Global Profile	Revealed Global Profile
Strict Order $\rightsquigarrow$	33.25%	3.87%	14.48%
Inverse Order $\leftsquigarrow$	33.25%	3.87%	14.48%
Parallel $\|\|$	9.7%	1.60%	34.03%
Exclusive #	23.8%	3.42%	17.97%
Conflict $\divideontimes$	0%	0.15%	0.12%
Unknown ?	0%	87.10%	18.91%

global profiles were input to our prediction approach in order to reveal possible behavioral relations that might exist among them. To reveal these relations, we use common services between the two services with an unknown relation. We applied our approach to a subset of the SAP reference models and our experiments show that we could reveal about 72% of the unknown relations in the global profile.

Our prediction algorithm is sensitive to the input global profile. If the global profile is derived from widely different processes that have the least in common, the prediction algorithm cannot reveal much. However, one nice result we found about our algorithm is the tendency to find clusters of related web services.

We limited our experiments to process models without loops, because of the sensitivity of the behavioral-relation computation algorithm. For instance, if two tasks A and B are in sequence and that sequence is nested in a loop, the behavioral-relation computation algorithm would indicate that $A \,||\, B$ instead of $A \rightsquigarrow B$. Knowing that $A \rightsquigarrow B$ is more decisive and yields more useful results by the prediction algorithm. One possibility to overcome the loop-sensitiveness limitation is to employ other behavioral-relation computation algorithms, e.g., those of the α-algorithm for process mining [1]. The results of our prediction algorithm is *independent* from the behavioral-relation computation algorithm as all of these algorithms provide the four basic relationships.

There are several directions for future work. First, we plan to tackle the problem of revealing conflicting relations identified during the merge of individual profiles. Second, we intended to extend the discovery beyond binary relations. That is, discover relations among fragments of web services. Also, we aim at applying further experiments on other collections of process configurations. Also, we intend to use more information from the process configurations artifacts in order to enhance our prediction algorithm. In particular, we aim at exploiting data dependencies among web services to help refine our predictions. Data dependencies can also be exploited to reveal conflicts that arise while deriving the global behavioral profile.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
2. AbuJarour, M., Awad, A.: Discovering Linkage Patterns among Web Services using Business Process Knowledge. In: Proceeding of the 8th International Conference on Services Computing. IEEE Computer Society (2011)

3. AbuJarour, M., Naumann, F.: Dynamic Tags For Dynamic Data Web Services. In: Workshop on Enhanced Web Service Technologies. ACM, Ayia Napa (2010)
4. Al-Masri, E., Mahmoud, Q.H.: Investigating Web Services on the World Wide Web. In: Proceeding of the 17th International Conference on World Wide Web, WWW 2008, pp. 795–804. ACM, New York (2008)
5. Basu, S., Casati, F., Daniel, F.: Toward Web Service Dependency Discovery for SOA Management. In: Proceedings of the 2008 IEEE International Conference on Services Computing, vol. 2, pp. 422–429. IEEE Computer Society (2008)
6. Bose, A.: Effective Web Service Discovery using a Combination of a Semantic Model and a Data Mining Technique. Master's thesis, Queensland University of Technology, Queensland, Australia (2008)
7. Curran, T.A., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model, 1st edn. Prentice Hall (1997)
8. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, vol. 30, pp. 372–383. VLDB Endowment (2004)
9. Eshuis, R., Grefen, P.: Structural Matching of BPEL Processes. In: Proceedings of the Fifth European Conference on Web Services, pp. 171–180. IEEE Computer Society (2007)
10. Fensel, D., Keller, U., Lausen, H., Polleres, A., Toma, I.: WWW or What is Wrong with Web Service Discovery? In: Proceedings of the W3C Workshop on Frameworks for Semantics in Web Services (2005)
11. Hagemann, S., Letz, C., Vossen, G.: Web Service Discovery - Reality Check 2.0. In: NWESP 2007: Proceedings of the Third International Conference on Next Generation Web Services Practices, pp. 113–118. IEEE Computer Society, Washington, DC, USA (2007)
12. Lecue, F., Leger, A.: Semantic Web Service Composition Based on a Closed World Assumption. In: Proceedings of the European Conference on Web Services, pp. 233–242. IEEE Computer Society (2006)
13. Lin, L., Arpinar, I.B.: Discovery of Semantic Relations Between Web Services. In: Proceedings of the IEEE International Conference on Web Services, pp. 357–364. IEEE Computer Society, Washington, DC, USA (2006)
14. Omer, A.M., Schill, A.: Web Service Composition Using Input/Output Dependency Matrix. In: Proceedings of the 3Rd Workshop on Agent-Oriented Software Engineering Challenges for Ubiquitous and Pervasive Computing, AUPC 2009, pp. 21–26. ACM (2009)
15. Rong, W., Liu, K., Liang, L.: Personalized Web Service Ranking via User Group Combining Association Rule. In: Proceedings of the 2009 IEEE International Conference on Web Services, ICWS 2009, pp. 445–452. IEEE Computer Society (2009)
16. Segev, A.: Circular Context-Based Semantic Matching to Identify Web Service Composition. In: Proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation, CSSSIA 2008, pp. 7:1–7:5. ACM (2008)
17. Sharma, S., Batra, S.: Applying Association Rules For Web Services Categorization. International Journal of Computer and Electrical Engineering 2(3), 465–468 (2010)
18. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Sheng, O.R.L.: Formulating the Data-Flow Perspective for Business Process Management. Info. Sys. Research 17(4), 374–391 (2006)
19. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Efficient Computation of Causal Behavioural Profiles Using Structural Decomposition. In: Lilius, J., Penczek, W. (eds.) PETRI NETS 2010. LNCS, vol. 6128, pp. 63–83. Springer, Heidelberg (2010)
20. Winkler, M., Springer, T., Trigos, E.D., Schill, A.: Analysing Dependencies in Service Compositions. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 123–133. Springer, Heidelberg (2010)