Evaluation of Workflow Similarity Measures in Service Discovery

Andreas Wombacher, Maarten Rozie
University of Twente,
7500 AE Enschede, The Netherlands
a.wombacher@utwente.nl

Abstract: Service discovery of state-dependent services has to take workflow aspects into account. To increase the usability of a query result, the results should be ordered with regard to their relevance, that is, the similarity of the query and the result list entry. Although there exist service discovery solutions considering workflow aspects, there is no support of ordering the result list. However, there exist several similarity measures in different research areas, which are applied in this paper to workflow similarity based on Finite State Automata.

1 Introduction

Web Services are an emerging technology. Although right now most applications mainly make use of the common communication protocols (HTTP and SOAP) and the good integration support based on a common interface specification language (WSDL), the loosely coupling is less applied. Reasons for this are the wish of the designer to manually control and coordinate the application, and the lack of expressiveness and precision of the service discovery (UDDI) supporting the design and the loosely coupling.

With regard to expressiveness of service discovery there exist more expressive approaches then key-value pairs as supported by standard UDDI. Such extensions of UDDI consider other aspects like semantic descriptions of the functionality [BK02, KB01, Coa04], syntactical structure of the interfaces as specified in WSDL, Quality of Service [BS04, LNZ04], or process aspects [WFMN04, WMN04] representing the behavior of a service.

As a consequence, having more expressive queries may increase the precision of the query results since a matching service has to fit to a more precise service description. Hence, a query might also be too specific in the sense that there is no service providing the complete service description. Quite vague queries on the other hand side provide a huge bunch of services supporting the query to the extend as it has been specified. In either case - over and under specification of a query - a similarity measure is of help. In the first case representing the over specification, a user aims for the service matching the query best. In the second case representing the under specification, a user aims to get the best fitting service from the bunch of possible services derived. However, addressing similarity for all the different aspects mentioned above is too challenging therefore in the following the discussion focuses on workflow aspects.

Workflow aspects in service discovery means to decide whether two workflow specifications guarantee successful interaction, that is, are considered to be consistent. We assume that the workflow models rely on standard message dictionaries like RoesettaNet which have a commonly agreed semantics. As a consequence the semantic equivalence of messages as the basic building blocks of workflows drills down to syntactic equivalence of message names. So far we haven't seen work done on workflow similarity. Hence, in this paper different notions of workflow similarity are discussed. In particular, known similarity measures from other domains are presented and are applied to a set of exemplary behavioral service descriptions represented as workflows. Finally, the similarity measures are compared and future research issues are raised.
The paper is structured as follows: In Section 2 the general properties of similarity measures are discussed and the basis for most presented similarity measures, that is, the *edit distance*, is introduced. The used workflow model and the sample data set used for the discussion of the similarity measures is introduced in Section 3. Section 4 captures similarity measures based on the structure of automata, while Section 5 discusses language based approaches. Both sections contain the results and a brief discussion of the findings on the sample data set derived by applying the corresponding approach. A comparison of the different approaches again based on the sample data set is discussed in Section 6. Finally, Section 7 concludes the paper.

2 Similarity Measures

A similarity measure is a method of estimating the "closeness" between two entities. In particular, the extreme values of similarity are equivalence and non-relatedness of the two entities. The function deriving a similarity value of two automata has to fulfill the following general properties:

- the function is symmetric, that is, the similarity measure of automata A and B equals the similarity measure of automata B and A;
- the values provided by the similarity measure are between zero (unrelated) and 1 equivalent;
- the triangular inequality applies to all approaches based on a metric space, that is, the similarity measure of automata A and B plus the one of automata B and C is greater then the similarity measure of A and C directly.

A similarity measure supporting these properties is the similarity of strings, which usually drills down to the distance of strings. In particular, there exist different distance measures for strings like for example the Hamming distance used in information theory [TH93] or the edit distance usually applied on strings in the context of text. The edit distance (or Levenshtein distance) [Lev66] between two words is the smallest number of substitutions, insertions, and deletions of symbols that can be used to transform one of the words into the other.

Lets consider for example the strings $abab$ and aab then the first string can be transformed into the second one by the following transformations $abab \rightarrow aba\varepsilon \rightarrow aaa\varepsilon \rightarrow aab\varepsilon$ requiring three operations. Alternatively, the following transformation can be applied $abab \rightarrow a\varepsilon ab$ requiring only a single operation. Since the edit distance is the minimum number of operations required to transform a string into another, the edit distance of the two strings is one.
Based on this distance value d the similarity value sim can be calculated by subtracting the distance value from the maximum difference m and diving the difference by the maximum difference m, that is, $sim := \frac{m-d}{m}$. However, the calculation of the maximum distance in case of infinite words is potentially infinite which limits the approach generally to finite strings. Further, also in a finite case the determination of the maximum edit distance is not that straight forward. Let's consider the following example based on the two finite languages $L(A) = \{a, b\}$ and $L(A') = \{aaa\}$. The maximum distance between the languages is the product of the maximum number of words contained in one of the languages and the length of the longest word contained in one of the languages, that is,

$$m = Max(|L(A)|, |L(A')|) * Max_{s \in L(A) \cup L(A')}(|s|)$$

where $|L(A)|$ specifies the length of the language $L(A)$ and $|s|$ the length of a string s. With regard to the example, the maximum distance m is $m = 2 * 3 = 6$.
In the following, similarity of automata is investigated based on the structure of automata or based on the language accepted by automata. However, most of the approaches are based on a distance measure or use directly edit distance as introduced above.

3 Workflow Model

A definition of workflow similarity must be based on a formal workflow model. Different formal models representing workflows exist, in the following, Finite State Automata [HMU01] are used. Other more expressive models with operations requiring higher computational complexity are for example Petri Nets [Pet81], Workflow Nets [AH02], or statecharts [Har87]. We use the least complex model because the more complex models still having polynomial computational complexity can be represented as Finite State Automata. Since this work is motivated by a service discovery scenario applying a workflow model having non-polynomial computational complexity will result in applications which are inapplicable.
Besides the expressiveness the different approaches can be classified according to their underlying communication model: The models suggested by van der Aalst [Aal99] and Kindler et.al. [KMR00], for example, support asynchronous communication. By contrast synchronous communication is supported by Wombacher et.al. [WFMN04]. Since Web Services often use synchronous communication based on the HTTP protocol, the goal workflow model is an extension of Finite State Automata which has been introduced as annotated Finite State Automata [WFMN04].

This reduction to Finite State Automata is therefore feasible and has been applied in the Web Service domain. In particular, a mapping of the orchestration language BPEL to annotated Finite State Automata has been proposed in [WFN04] and has been used in implementing a service discovery engine [WMN05].

3.1 Formal Definition

Finite State Automata (FSA) are well studied. Formally, Finite State Automata can be represented as follows [HMU01]:

Definition 1 (Finite State Automata (FSA)) *A Finite State Automaton A is represented as a tuple* $A = (Q, \Sigma, \Delta, q_0, F)$ *where :*

- Q *is a finite set of states,*
- Σ *is a finite set of messages,*
- $\Delta : Q \times \Sigma \times Q$ *represents labeled transitions,*
- q_0 *a start state with* $q_0 \in Q$*, and*
- $F \subseteq Q$ *a set of final states.*

A FSA A generates a language $L(A)$ which enumerates the (possibly infinite) set of all message sequences supported by a business process. The empty word ε indicates a transition changing a state without a contribution to the accepted word. The graphical representation of a FSA depicts states as circles and transitions as arcs (annotated with labels). Final sates are depicted as states with a thick line. Example FSA are depicted in Figure 1.
With regard to the workflows representing the behavior of a service as mentioned in the introduction, the alphabet of the automaton contains the set of operations which are used and provided by a service. As a consequence, an accepted word contained in the language of an automaton represents an execution sequence supported by a service representing the order in which the operations are performed.

3.2 Example Workflows

Within this paper, the preliminary analysis of similarity measures is based on a set of sample data. The sample data is based on an abstract set of operations consisting of the characters a to e. It contains different complexity classes of automata. In particular, the sample set contains eight acyclic automata accepting a finite language and four cyclic automata accepting an infinite language. In either automaton class, different similarities have been considered, which can be used to evaluate the similarity measures derived by the different approaches. In the following, the different relations are discussed and introduced as evaluation criteria.
Six acyclic automata are depicted in Figure 1 representing three automata accepting the same language, although the structure of the automata is different (automaton

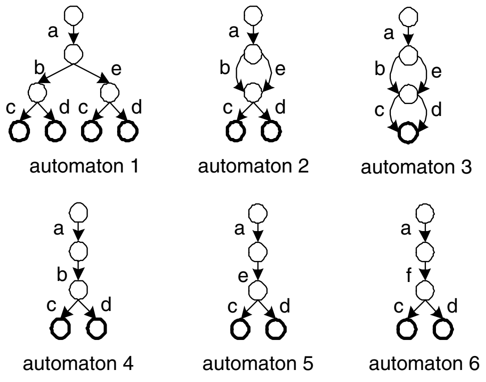

Figure 1: Workflow Models: Part 1

1, 2 and 3). In case of language equivalence, the corresponding similarity measure must be the same.

Criteria 1 *The similarity measures between each combination of automaton 1, 2, and 3 must be the same.*

The languages accepted by automata 4 and 5 are complementary subsets of the languages accepted by automata 1, 2, or 3. Since the subsets have the same size, the similarity measure must be the same.

Criteria 2 *The similarity measures of one out of automaton 1, 2, or 3 and automaton 4 and 5 respectively are the same.*

Automaton 6 accepts a language which is different from the first five automata although all words contained in the accepted language have the same length. Therefore the similarity measures of the first five automata with automaton 6 must be the smallest. Further, since languages of automata 4 and 5 also do not share a word, the similarity measures of two out of these three automata must be the same.

Criteria 3 *The similarity measures of two out of automata 4, 5, and 6 are the same.*

The automata depicted in Figure 2 contain two acyclic ones (automata 7 and 8) and four cyclic ones (automata 9 to 12). The two languages of the acyclic automata accept longer words then the first six ones do and the language of automaton 7 is a subset of the one of automaton 8. The languages accepted by automaton 9 and 10 contain the language of automaton 8. Further, the language of automaton 11 contains the language of automaton 10. The language of automaton 12 shares some words with the languages of all automata except automaton 6.
Be aware that automata 9 and 10 contain simple cycles, while the cycles in automata 11 and 12 are considered to be complex cycles, where complex cycles are specified

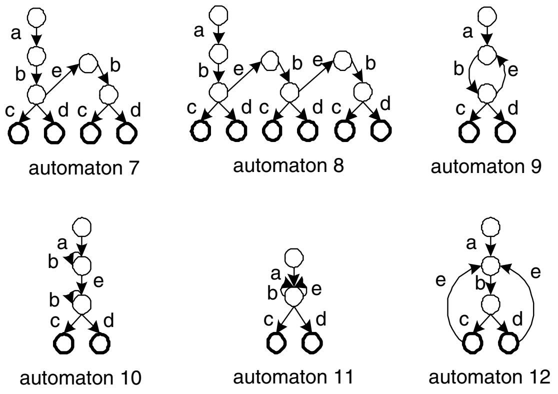

Figure 2: Workflow Models: Part 2

by at least two cycles sharing at least a single state. The existence of complex cycles effects some of the investigated approaches by increasing significantly the computational complexity.
The size of the language of an automaton not supported by another automaton causes differences in the similarity measure compared with another automaton while keeping the shared messages equal.

Criteria 4 *(a) The similarity measure of automata 7 and 8 must be higher then the similarity measure of automata 7 and 9.*
(b) The similarity measure of automata 7 and 9 must be higher then the similarity measure of automata 7 and 11.

The language of automata 9 and 10 are contained in the language of automaton 11, while the language of automaton 11 has a non-empty intersection with the language of automaton 12. Right now, there is no specific criterion which can be derived from this.

4 Structural Automaton Similarity

In the following two approaches of structural similarity are discussed, which are suffering from the fact that different structures may result in accepting the same language, that is, the same set of message sequences.

4.1 Graph Similarity based on Edit-Distance

An automaton can be interpreted as a directed graph and therefore graph similarity measures can be applied. An exemplary similarity measure based on edit distance has been proposed in [CKS98] addressing graph isomorphism, while [MB98] addresses subgraph isomorphism.

In [MB98] an error correcting subgraph isomorphism detection is introduced which is based on edit operations on the graph structure. However, the provided distance measure is asymmetric, which contradicts the stated similarity measure properties. The asymmetry is caused by ignoring parts of one automaton only to calculate the subgraph isomorphism. However, the general idea of operations modifying the graph structure seems to be interesting.
[CKS98] can be applied to graphs of the same order and is based on a one-to-one mapping ϕ of two graphs/automata. The underlying idea is that in two isomorphic graphs the distance d between two states u and v in one automaton equals the distance d of the associated states $\phi(u)$ and $\phi(v)$ in the other automaton. In particular, the distance d based on the mapping ϕ between two automata A and A' is defined as the sum of the absolute difference between the distance values between all pairs of states, that is,

$$d_\phi(A, A') = \sum_{u,v \in Q} |d_A(u,v) - d_{A'}(\phi(u), \phi(v))|$$

where Q is the set of states in automaton A. Based on this definition, the distance of two automata A and A' is the minimal distance value based on a mapping ϕ. A quite limiting factor in this definition is the requirement of a one-to-one mapping, which only applies for automata having equally sized sets of states. However, graph isomorphism is different from language equivalence in automata theory. In particular, in automata theory several automata may represent the same language having different automata representations. For example the example automata 1, 2 and 3 accept the same language although having different graph structures.
As a consequence of this investigation, the graph based approaches are not applicable to the addressed problem space, thus are not considered further.

4.2 Inheritance Similarity

The algorithm of automated reconciliation as presented in [DHL+05] inherently uses a similarity measure. The approach focus on the common alphabet of two automata and removes the exclusively used messages of the alphabets. Further, automata transformation rules as specified in [vdABry] for Workflow-Nets are used to transform both automata to a same automaton using only the shared alphabet. If this transformation is not possible in accordance with the specified transformation rules, the similarity measure is zero, otherwise the similarity equals the degree of overlap of the shared with the original alphabet.
Due to the similarity definition based on the overlap of the own alphabet with the shared one, the similarity measure is asymmetric in case the automata have alphabets of different size. This contradicts the claimed properties of a similarity measure as stated in Section 2. Therefore, the similarity measure has been changed to represent the overlap of the shared alphabet with the union of both automata alphabets. Applying this modified definition to the sample data set results in the similarity mea-

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
A1	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A2		1.00	1.00	0.88	0.88	0.67	0.00	0.00	0.00	0.00	0.00	0.00
A3			1.00	0.88	0.88	0.67	0.00	0.00	0.00	0.00	0.00	0.00
A4				1.00	0.75	0.75	0.00	0.00	0.88	0.00	0.00	0.00
A5					1.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00
A6						1.00	0.00	0.00	0.75	0.00	0.67	0.00
A7							1.00	0.00	0.00	0.00	0.00	0.00
A8								1.00	0.00	0.00	0.00	0.00
A9									1.00	0.00	0.00	0.00
A10										1.00	0.00	0.00
A11											1.00	0.00
A12												1.00

Table 1: Similarity measure based on IPR

sures depicted in Table 1. The table contains only the upper triangular matrix to improve the readability. The matrix has been proven to be symmetric.
The checking of the introduced criteria reveals that the Criterion 1 stating the equivalence of automata 1, 2 and 3 is fulfilled by this approach indicated by having a 1.00 similarity measure at each combination of these automata.
The Criterion 2 is not fulfilled because the similarity measure of automata 1 and 4 is zero, while the one of automata 2 and 4 is 0.88 although they are supposed to be the same. The reason for this difference is that the approach does not support the removal of an alternative in an automaton. In the above cases, the transition labeled e has to be removed from automata 1 and 2. However, a transition can only be removed if the transition ends in a state which is still connected to the start state after removing that transition. In case of automaton 1 this is not the case, while it is the case for automaton 2. As a consequence the two similarity measures are different contradicting Criterion 2.
Criterion 3 addressing empty subsets is fulfilled. This can be observed in the table when comparing the similarity measure of automata 4 and 5 with the similarity measure of automata 4 and 6 or 5 and 6 respectively. The Criterion 4 on the subset relation of automata can hardly be evaluated, because all similarity measures mentioned in the criteria are 0 thus indicating unrelated automata.
Finally, the similarity measure only provides values in case the automata are structurally quite similar, thus, do not have structural alternatives resulting in a new subautomaton as observed in Criterion 2. In particular, in case at least one execution sequence is not shared, the similarity value is zero, which can be considered as too restrictive.

5 Automaton Language Distance Measures

In the following language based approaches are discussed, which are usually suffering from handling infinite languages.

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	0	0	2	2	6	6	14
A2		0	0	2	2	6	6	14
A3			0	2	2	6	6	14
A4				0	4	4	4	12
A5					0	4	8	16
A6						0	10	20
A7							0	4
A8								0

Table 2: Edit Distance on Sets of Strings

5.1 Edit-Distance on Language

A straight forward approach based on string edit distance is to consider a language as a potentially infinite set of accepted words. The intuitive algorithm is a comparison of every word of the first language with every word in the second language, calculating the edit distance, and summing up the derived edit distance values. Applying this approach to the finite languages in the sample data set results in the edit distance values contained in Table 2. The table contains only the upper triangular matrix to improve the readability. The matrix has been proven to be symmetric.
Based on these edit distance values the similarity measure can be calculated as stated in Section 2. Criterion 1 related to the equivalence of automata turns out to be valid because all combinations of automata 1, 2 and 3 provide an edit distance zero. Criterion 2 is also fulfilled because the edit distance contained in the table is equal for all pairs of automata 1, 2, or 3 with automata 4 or 5. Further, Criterion 3 on the empty subsets is valid since the same distance values are provided in the table. Criterion 4 can not be applied because the approach is limited to acyclic automata only, thus, does not provide values for automata 9 and 11.
This approach does not work in case at least one language is infinite. However, in [Moh03a, Moh03b] an approach has been proposed based on composition of automata and so called transducers. A transducer represents a transformation step of a transition label and associating costs to these changes. The edit distance is then the shortest path on the composed automata containing the change operation costs associated to the transitions. Actually, the approach calculates the minimal distance of a word accepted by the first automaton with a word accepted by the second automaton. Applying this approach to the sample data set results in the edit distance values contained in Table 3 The table contains only the upper triangular matrix to improve the readability. The matrix has been proven to be symmetric.
Due to the construction of the example, quite a lot of automata have at least one word in common, thus making the distance value zero. The remaining automata each have a word which has an edit distance of one. As a consequence, of this uniform results contained in Table 3 it turns out that this approach can not be considered to be useful although it can handle cyclic automata. Therefore the analysis of the criteria is omitted.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
A1	0	0	0	0	0	1	0	0	0	0	0	0
A2		0	0	0	0	1	0	0	0	0	0	0
A3			0	0	0	1	0	0	0	0	0	0
A4				0	1	1	0	0	0	1	0	0
A5					0	1	1	1	1	0	0	1
A6						0	1	1	1	1	1	1
A7							0	0	0	0	0	0
A8								0	0	0	0	0
A9									0	0	0	0
A10										0	0	1
A11											0	0
A12												0

Table 3: Edit Distance based on Transducers with Standard Automata

5.2 Edit-Distance on Automata

In the transducer approach the edit distance is defined as the minimum distance of two strings in the two languages. However, the remaining strings in the language are not influencing the edit distance value. As a consequence, the similarity measure represents similarity of the intersection of the two languages rather then the similarity of the equivalence of the languages.
Alternatively, an approach is presented, which takes all strings into account by considering the probability of the occurrences of the different words. In particular, shorter words are considered to be more likely then longer words. Thus, the expectation value of the minimum number of change operations is calculated based on transforming every string of one language into a string of the other language and vice versa.

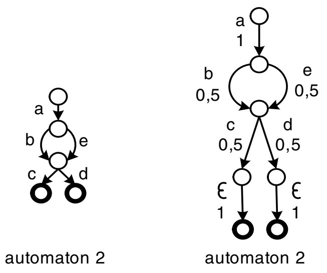

Figure 3: Automaton 2 and weighted Automaton 2

The algorithm is based on assigning the transitions of automata with probabilities, which result in weighted automata. Figure 3 depicts the weighted automata derived from automaton 2. The weights are calculated such that the sum of all weights assigned to transitions leaving a state is one. The final states are represented as

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0.00	0.00	0.00	0.02	0.02	0.07	0.04	0.05
A2		0.00	0.00	0.02	0.02	0.07	0.04	0.05
A3			0.00	0.02	0.02	0.07	0.04	0.05
A4				0.00	0.03	0.03	0.01	0.01
A5					0.00	0.03	0.04	0.04
A6						0.00	0.04	0.04
A7							0.00	$4.8 * 10^{-4}$
A8								0.00

Table 4: Edit Distance based on Transducers with Weighted Automata

explicit states connected via an ε transition, because terminating the processing is an option which has to be represented explicitly. The probability of an execution sequence is calculated by multiplying all weights along the path. It is guaranteed that the sum of weights of all paths is below 1.
In [Moh03a, Moh03b] the edit distance of weighted acyclic automata is presented. The approach is again based on transducers along the lines as described in Section 5.1. Applying the approach on the acyclic automata of the sample data set results in the distance values contained in Table 4. The table contains only the upper triangular matrix to improve the readability. The matrix has been proven to be symmetric.
Criterion 1 addressing the equivalence of automata is fulfilled by the approach as indicated in the table by the distance of zero of automata 1, 2 and 3 respectively. Criterion 2 addressing the comparable subsets is also fulfilled, and the distance values contained in the table are 0.02. The Criterion 3 of the empty subsets is also provided resulting in a distance value of 0.03 in the table. The Criterion 4 is not applicable due to the lack of support of cyclic automata, which is a limitation on the applicability of the approach.

5.3 Edit-Distance on n-gram Representations

An alternative approach of representing infinite languages derived from cyclic automata is proposed in [MWF05] based on n-grams lists. n-grams have been applied in text indexing approaches in particular for substring matching [BY92]. The general idea behind an n-gram list is representing a single string by an ordered list of substrings with length n. In particular, only the first occurrence of an n-gram is considered, while later occurrences of the same n-gram are omitted in the n-gram list. Thus, an n-gram list results in a single representation of strings with different length. In particular, an infinite language can be represented as a finite set of n-gram lists, which comes for the price of a loss of information introducing imprecision. In particular, the length n on the one hand side controls the complexity of all operations required in the concrete application scenario, and on the other hand influences the fuzziness of the operation result applied on the n-gram representation.
With regard to the example, the infinite language of automaton 10 can be represented

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
A1	0	0	0	4	4	12	16	16	16	88	930	82
A2		0	0	4	4	12	16	16	16	88	930	82
A3			0	4	4	12	16	16	16	88	930	82
A4				0	8	8	12	12	12	92	934	78
A5					0	8	20	20	20	84	934	88
A6						0	24	24	24	120	1136	102
A7							0	0	0	48	508	46
A8								0	0	48	508	46
A9									0	48	508	46
A10										0	-	88
A11											0	-
A12												0

Table 5: String Edit Distance on 2-gram Lists

as 2-grams in accordance to [MWF05] as follows:
$\{ \quad <\$a, ab, bc, c\# >, <\$a, ab, bd, d\# >,$
$<\$a, ab, be, eb, bc, c\# >, <\$a, ab, be, eb, bd, d\# >\}$
In the set a single 2-gram list is represented in $< \cdots >$ signs and the different 2-grams are comma separated in the list notation. In particular, the string abc accepted by automaton 10 is represented as a 2-gram list as $< \$a, ab, bc, c\# >$ where the character \$ is used as a symbol representing the start of a string needed in case the string length is shorter then the n-gram itself. thus, the first 2-gram is $\$a$ combining the string start symbol and the first character. Next, the first two characters a and b form the second 2-gram ab, b and c form the third 2-gram, and c and the termination symbol $\#$ of a string forms the forth and last 2-gram $c\#$.
Following this construction principle and having in mind that only the first occurrence of an n-gram is represented in an n-gram list, the strings $abebc$, $abebebc$, and $abebebebc$ are all represented by the same 2-gram list $< \$a, ab, be, eb, bc, c\# >$. Thus, the infinite language of a cyclic automaton can be represented as a finite set of n-gram lists.
Considering an n-gram list as a string, the approach in Section 5.1 can be applied. In particular, the edit distance now can also handle cyclic automata as opposed to the original approach. Applying this approach to the sample data set results in the edit distance values presented in Table 5. The table contains only the upper triangular matrix to improve the readability. The matrix has been proven to be symmetric.
Criterion 1 of the automaton equivalence is fulfilled as indicated in the table by the edit distances of zero. Criterion 2 of comparable subsets is also supported. The table contains 4 as the common distance value. Criterion 3 of the empty subsets is supported and the common distance value is 8 as indicated in the table. Criterion 4 is only partly supported. The edit distance of automata 7 and 8 is zero and equals the similarity measure of automata 7 and 9. This is conflicting with the criterion. The explanation of this result is that automata 7, 8 and 9 have the same 2-gram representation. A change of the encoding of the automata to a higher n increases

the precision of the representation and can resolve this issue. However, increasing the n results in higher operational costs, which is not intended when considering the already high computational effort as indicated for example by the edit distance of automata 6 and 11. Thus, this approach can handle cyclic automata on the price of higher computational complexity and potentially imprecise edit distance measures.

6 Comparison of Approaches

An analytical comparison of the approaches would require a reference solution of similarity. Such a reference would indicate what a human workflow expert considers to be most similar. Such an investigation has not been performed so far and a corresponding reference similarity value is not available (in case it exists at all). The work presented here is the first step in the direction of a workflow similarity measure which requires further investigation. Therefore the evaluation can not be performed in an analytical way but must be performed on an empirical evaluation of the different results.
Comparing the different approaches it turns out that the structural based approaches as discussed in Section 4 are not applicable. This is because equivalence of automata as the highest possible similarity is defined in terms of equivalence of languages. Since automata with different structure may represent the same language these approaches are ruled out very fast.
The language based approaches as discussed in Section 5 turn out to be quite helpful for finite languages, that is, acyclic automata. The results contained in Table 2 and 4 are a good basis for the similarity measure due to the differences in values. However, comparing the tables one by one, it turns out that a relation of distance values of the two tables can be defined which is violated only in a few cases. These differences are the interesting parts, which require a deeper investigation on the usability of these similarity measures.
The language approaches supporting cyclic automata require more research work. The approach based on weighted automata turns out to provide a distance measure which provides only a coarse grained differentiation of automata (see Table 4). However, the n-gram based approach (see Table 5) has two major draw backs: it requires high computational effort in case of automata with complex cycles, that is, at least two cycles in an automaton share at least a single state. Further, due to the abstraction introduced by the n-grams, the certain imprecision is introduced resulting in automata to be considered equivalent although they are not (see automata 8 and 9 in Table 5).

7 Conclusion and Future Work

In this paper, the need of a similarity measure of workflow models in the Web Service domain is motivated and several similarity measures are investigated based on a sample data set, which has been constructed such that different relationships of automata are covered. Applying and analyzing the different approaches reveals that there are applicable solutions for acyclic automata, while further research is needed to get efficient approaches for cyclic automata. The similarity measures of acyclic automata result in an ordering of automata, but further research is required to understand better which similarity measure fits best the human intuitive similarity measure (in case there exists something like that).

References

[Aal99] W.M.P. van der Aalst. Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.

[AH02] W.M.P. van der Aalst and Kees van Hee. *Workflow Management - Models, Methods, and Systems*. MIT Press, 2002.

[BK02] Abraham Bernstein and Mark Klein. Discovering Services: Towards High-Precision Service Retrieval. In *Proceedings of CAiSE International Workshop, Web Services, E-Business, and the Semantic Web (WES)*, LNCS 2512, pages 260–275. Springer, 2002.

[BS04] A. Soydan Bilgin and Munindar P. Singh. A DAML-based Repository for QoS-Aware Semantic Web Service Selection. In *Proceedings of IEEE International Conference on Web Services*, pages 368–375, 2004.

[BY92] R. A. Baeza-Yates. Text retrieval: theory and practice. In J. van Leeuwen, editor, *Proceedings of the 12th IFIP World Computer Congress*, pages 465–476, Madrid, Spain, 1992. North-Holland.

[CKS98] Gary Chartrand, Grzegorz Kubicki, and Michelle Schultz. Graph Similarity and Distance in Graphs. *Aequationes Mathematicae*, 55:129–145, 1998.

[Coa04] OWL Service Coalition. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2004.

[DHL$^+$05] Zongxia Du, Jinpeng Huai, Yunhao Liu, Chunming Hu, and Lei Lei. IPR: Automated Interaction Process Reconciliation. In *Proceedings of IEEE/ACM International Conference on Web Intelligence (WI)*, 2005. accepted for publication.

[Har87] D. Harel. Statecharts: A Visual Formalism For Complex Systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.

[KB01] Mark Klein and Abraham Bernstein. Searching for Services on the Semantic Web using Process Ontologies. In *Proceedings of 1st Semantic Web Working Symposium (SWWS)*, Stanford, 2001.

[KMR00] Ekkart Kindler, Axel Martens, and Wolfgang Reisig. Inter-operability of Workflow Applications: Local Criteria for Global Soundness. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 235–253. Springer-Verlag, 2000.

[Lev66] L. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals,. *Soviet Physics–Doklady*, 10(8):707–710, 1966.

[LNZ04] Y. Liu, A. H. H. Ngu, and L. Zeng. QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of WWW*, page 66ff, 2004.

[MB98] Bruno T. Messmer and Horst Bunke. A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.

[Moh03a] Mehryar Mohri. Edit-Distance of Weighted Automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *Proceedings of International Conference on Implementation and Application of Automata (CIAA)*, volume 2608 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2003.

[Moh03b] Mehryar Mohri. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982, 2003.

[MWF05] Bendick Mahleko, Andreas Wombacher, and Peter Fankhauser. Process-annotated Service Discovery facilitated by an n-gram based Index. In *Proceedings of IEEE International Confernce on e-Technology, e-Commerce and e-Service (EEE)*, pages 2–8, 2005.

[Pet81] James L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[TH93] Hans Tzschach and Gerhard Hasslinger. *Codes fuer den stoerungssicheren Datentransfer*. Oldenburg Verlag, 1993.

[vdABry] W. M. P. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2):125–203, uary.

[WFMN04]Andreas Wombacher, Peter Fankhauser, Bendick Mahleko, and Erich Neuhold:. Matchmaking for Business Processes Based on Choreographies. *Intl. Journal of Web Services*, 1(4):14–32, 2004.

[WFN04] A. Wombacher, P. Fankhauser, and E. Neuhold. Transforming BPEL into annotated Deterministic Finite State Automata enabling process annotated service discovery. In *Proceedings of International Conference on Web Services (ICWS)*, pages 316–323, 2004.

[WMN04] Andreas Wombacher, Bendick Mahleko, and Erich Neuhold. IPSI-PF: A Business Process Matchmaking Engine. In *Proceedings of Conference on Electronic Commerce (CEC)*, pages 137–145, 2004.

[WMN05] Andreas Wombacher, Bendick Mahleko, and Erich Neuhold. IPSI-PF:A Business Process Matchmaking Engine based on annotated Finite State Automata. *Journal on Information Systems and E-Business Management*, 3(2):127–150, 2005.