Towards a Trace Index Based Workflow Similarity Function

Pol Schumacher and Mirjam Minor

Goethe Universität Frankfurt - Institut für Informatik**
D-60325 Frankfurt am Main, Germany
[schumacher|minor]@cs.uni-frankfurt.de

Abstract. In this paper we present first steps towards an index based workflow similarity function. Classical approaches on workflow similarity perform a pairwise comparison of workflows, thus the workflow similarity function presented in this paper can speedup the calculation as the comparison is performed on the index.

Keywords: Knowledge representation and reasoning, case-based reasoning, workflow reasoning

1 Introduction

Similarity is a subjective concept, suited for approximate rather than exact reasoning [1]. Similarity functions are used in a manifold of applications for example to retrieve similar cases of a case-base in a case-based reasoning scenario. *Workflows* are "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [2]. Multiple different similarity functions for workflows have been presented in the past [3–6]. There is a need for improvement of the computational performance of workflow similarities. The similarity function which we present in this paper employs an index to calculate the similarity between two workflows. The similarity computation of two workflows based on this novel index is much more efficient than the existing approaches which perform a pairwise comparison on the workflow structure. A workflow instructs a user to perform certain activities in a certain order. The workflow index which we present in this paper is based on the precedence relation for activities. The comparison of two workflows is done by comparing the indices of the workflows. The index can determine multiple categories of differences between workflows. It can detect missing activities, it gathers differences in the positioning of activities, it incorporates discrepancies of control-flow nodes and it includes the importance of certain activities into the similarity computation. This paper is organized as follows. In the next section we introduce our index

** This work was funded by the German Research Foundation, project number BE 1373/3-1.

and how it is used to determine the similarity of two workflows. It includes a running example illustrating the index creation. After that we are going to present some related work on workflow similarity functions and discuss the differences to our approach. Finally, we draw a short conclusion and provide an outlook on our future work.

2 Workflow similarity

This section begins with the presentation of our workflow description language. Then we are going to recapitulate some basic definitions on multisets which are necessary for the workflow similarity calculation. After that we present the construction of the index and finally we are going to employ the presented index to calculate workflow similarities. We have chosen an XML-based language [7] for workflow descriptions that is part of the CAKE[1] system. It is a block-oriented workflow description language: The control-flow can contain sequences, XOR-, AND-, LOOP-blocks. These building blocks cannot be interleaved but they can be nested. Sequences contain at least an activity, XOR-, AND or, LOOP-block. XOR- and AND-blocks contain at least two sequences. In the case of an XOR-block one sequence is chosen which is executed. In the case of AND-blocks both sequences are executed. Fig. 1 contains on the top a sample for a workflow in BPMN. Activities are represented by rounded rectangles and control-flow nodes by the diamond with "X" for XOR-nodes and an "+" for AND-nodes. The arrows between the elements represent the control-flow.

We are going to recapitulate some basic definitions of multisets [8] as our similarity function uses multisets. Multisets are sets which can contain multiple occurrences of the same element.

Definition 1 (Multiset) *Let D be a set. A multiset over D is a pair $\langle D, f \rangle$, where D is a set and $f : D \to \mathbb{N}$. is the function.*

Definition 2 (Multiset operations) *Suppose that $\mathcal{A} = \langle A, f \rangle$, $\mathcal{B} = \langle A, g \rangle$ and $\mathcal{C} = \langle A, h \rangle$ are multisets then multiset operations cardinality, sum and intersection are defined as follows:*

$$card(\mathcal{A}) = \sum_{a \in A} f(a) \ .$$

$$\mathcal{A} \uplus \mathcal{B} = \mathcal{C}, with \ h(a) = f(a) + g(a) \ .$$
$$\mathcal{A} \cap \mathcal{B} = \mathcal{C}, with \ h(a) = \min\big(f(a), g(a)\big) \ .$$

The index creation is illustrated in Fig. 1. The index for a workflow is created using its executions traces depicted in the middle layer of Fig. 1. A workflow execution trace is a sequence of activities which represents the order of execution of activities, as it follows a workflow model. An activity cannot occur multiple

[1] Collaborative Agile Knowledge Engine

times in a trace. If a workflow model contains an AND-, XOR- or LOOP-block it can produce multiple different workflow execution traces. For a given workflow model it is possible to produce the set of possible execution traces by traversing every possible path through the workflow. Please note that workflows containing AND-blocks can produce a large (exponential growth) number of different traces. The large number of traces is only a problem during the offline creation of the index. We expect that this problem can be overcome in future by modifying the trace creation process. Two workflows can be compared using their trace set. A similarity function based on the trace sets could be derived from the cardinality of the intersection of the sets. Unfortunately this would result in a very pessimistic similarity function as one missing activity in a trace compared to another trace is sufficient that the traces are regarded as unequal. Therefore it is necessary to split the traces into smaller chunks. A trace of a workflow can be used to derive a precedence relation of that particular trace. The precedence relation of two activities is the information that an activity must be performed before another activity. Due to control-flow nodes a workflow can contain multiple different, even conflicting precedence relations for two activities. For example sequences of an AND-block can produce conflicting relations, as the order of execution between the two sequences in the block is not defined. This conflicting precedence relations are not a problem for our similarity function as it does not rely on a single relation but it does collect all relations for a workflow in an index. A set of traces induces a set of precedence relations. The participants of the precedence relations are collected by a multiset sum. The resulting multiset is used as index. It is important that we are using multisets instead of normal sets. If the index would use normal sets, it would lose a lot of distinctiveness. Fig. 1 illustrates the creation of the index for a sample workflow.

Definition 3 (Trace precedence relation) *Let W be a workflow. Let A be the set of activities a_i of W. Let the sequence of activities $w = (a_m, ..., a_n)$ with $a_m, ..., a_n \in A$ be an execution trace of W. Then $(<_w, A, A)$ is the precedence relation within a trace w. $a_q <_w a_s$ if a_q has an earlier position in w than a_s.*

Definition 4 (Trace index) *Let W be a workflow and let T_W be the set of all trace precedence relations which can be produced by the workflow W, then the trace index of W denoted $index(W)$ is defined as:*

$$index(W) = \biguplus_{\forall t_i \in T_W} <_{t_i} \quad .$$

The similarity between two workflows can be calculated by means of their index. The index can be precomputed , thus the actual similarity computation is very fast. The similarity of two indices is computed with a similarity measure which is an extension of the Dice coefficient [9]. The Dice coefficient is a well-known similarity measure for normal sets, for two sets A and B $Dice(A, B) = \frac{2\,|A \cap B|}{|A| + |B|}$ [9].

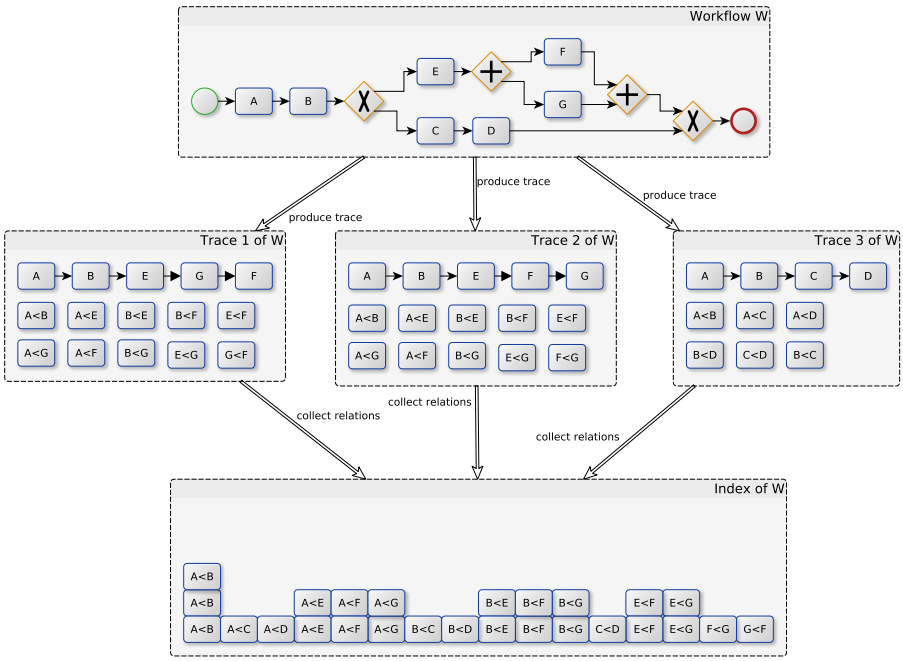

Fig. 1: Illustration of index creation of a workflow.

Definition 5 (Workflow similarity) *Let W and G be workflows then their similarity denoted $sim(W, G)$ is defined as:*

$$sim(W, G) = \frac{2\ card(index(W) \cap index(G))}{card(index(W) \uplus index(G))}\ .$$

Fig. 2 shows how different types of differences between workflows are covered by the trace index. The first example shows how the similarity functions takes into account missing activities. In the right workflow, the activity B is missing and the trace index of the right workflow has only one precedence relation element in common with the trace index of the left workflow. The example in middle shows how the similarity function penalizes a misplaced activity. In the sample workflows the last two activities swapped the position. The trace index of the left and the right workflow contain one precedence relation element which is not contained in the trace index of the other workflow. The higher the difference in positioning of the activities is, the more precedence relation elements differ and thus the lower similarity is. The last example illustrates how different control-flow nodes are taken into account. The trace index does not contain control-flow nodes explicitly but the semantics of the control-flow nodes are used during the trace index creation therefore the trace index of the left workflow does

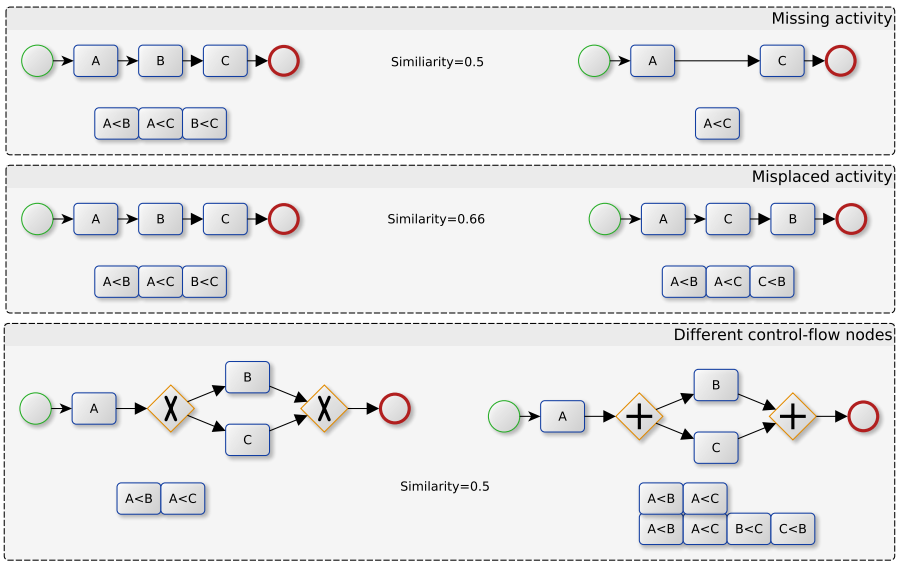

Fig. 2: Illustration of the use of the index for the detection of differences between workflows.

only contain two precedence relation elements while the right one does contain four precedence relation elements.

3 Related work

Multiple workflow similarity functions have been presented in the past. Wombacher and Li represent workflows based on its finite set of states [4]. In their representation states are n-grams, i.e. a sequence of n transition labels ending in that state. To calculate the distance of two workflows, they calculate the sum of the minimal distance of the states of a workflow with the states of the other workflow. The distance of two states is the string edit-distance of the n-grams. They use multiplicity of the state multisets to weight the distances of the states. The idea to use state n-grams is similar to our idea of the trace relations and they too use the multiplicity of multisets as means of weighting. The main difference to our approach is that they need to calculate the minimal edit-distance for every state for every pair of workflows in the repository while in our approach workflows are compared based on their relation-fingerprint. Eshuis and Grefen define relations between activities of a BPEL process [3]. Based on these relations and the activities of the workflow a graph can be defined. The authors define different exact and inexact matchings on that graph. The main advantage of that approach is that it ignores irrelevant syntactic differences and that it

can be computed in an efficient way too. Our approach takes into account more details (e.g. the number of occurrences of an activity) and can be computed efficiently too. Kunze et al. calculate the similarity of two processes based on their behavioral profile [5]. For the profile they define the relations strict order, exclusiveness and interleaving order. The profile is built by creating a matrix with a column and a row for every activity in the process. For every pair of activities the respective relation name is inserted into the matrix. The similarity between two workflows is calculated by the Jaccard similarity coefficient of the two behavioral profiles. The authors show that using the strict order relation provides the best result. The strict order relation which is used by the author is similar to our trace relation, but as we are using multisets we take into account the weight of a relation. Earlier related work [6, 10] use graph based approaches to compute workflow similarities. In contrast our approach uses an index.

4 Conclusion and future work

In this paper we presented first steps towards a similarity calculation for workflows based on an index induced by execution traces. In our future work we will further improve the approach. We aim to develop a method to handle the large number of traces created by AND-nodes. In addition we will extend the method to cover more elements of a workflow like input- and output-objects of activities. To compare our approach with others we are planning to perform an evaluation.

References

1. Richter, M.M., Weber, R.O.: Case-Based Reasoning - A Textbook. Springer (2013)
2. Workflow Management Coalition: Workflow management coalition glossary & terminology (1999) last access 05-23-2007.
3. Eshuis, R., Grefen, P.: Structural matching of BPEL processes. In: Web Services, 2007. ECOWS'07. Fifth European Conference on, IEEE (2007) 171–180
4. Wombacher, A., Li, C.: Alternative approaches for workflow similarity. In: Services Computing (SCC), 2010 IEEE International Conference on, IEEE (2010) 337–345
5. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity - a proper metric. In Rinderle-Ma, S., Toumani, F., Wolf, K., eds.: BPM. Volume 6896 of Lecture Notes in Computer Science., Springer (2011) 166–181
6. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Inf. Syst. **40** (2014) 115–127
7. Minor, M., Schmalen, D., Bergmann, R.: XML-based representation of agile workflows. In Bichler, M., Hess, T., Krcmar, H., Lechner, U., Matthes, F., Picot, A., Speitkamp, B., Wolf, P., eds.: Multikonferenz Wirtschaftsinformatik 2008, GITO-Verlag Berlin (2008) 439–440
8. Syropoulos, A.: Mathematics of multisets. In: Multiset Processing. Springer (2001) 347–358
9. Carullo, M., Binaghi, E., Gallo, I.: An online document clustering technique for short web contents. Pattern Recognition Letters **30**(10) (2009) 870–876
10. Minor, M., Tartakovski, A., Schmalen, D.: Agile workflow technology and case-based change reuse for long-term processes. IJIIT **4**(1) (2008) 80–98