Generalization of Workflows in Process-Oriented Case-Based Reasoning

Gilbert Müller and **Ralph Bergmann**
Department of Business Information Systems II
University of Trier
54286 Trier, Germany
http://www.wi2.uni-trier.de
{muellerg}{bergmann}@uni-trier.de

Abstract

In this paper, we introduce the concept of generalized cases into process-oriented case-based reasoning. We present the formal foundations for the generalization of workflow cases as well as a new algorithm for generalizing semantic workflows, guided by ontological knowledge of the domain. Further, the specialization of workflows w.r.t. a current query is addressed. An experimental evaluation demonstrates the capability of the approach for workflow adaptation showing that the adapted workflows have a similar quality compared to that of original workflows. Furthermore, the retrieval performance can be improved by a reduction of the case-base size while the coverage of cases is significantly increased.

Introduction

In case-based reasoning (CBR), new problems are solved by retrieval and adaption of cases stored in a case base. Traditionally, a case represents a single specific problem-solving experience, which is captured and stored for reuse. However, from the very beginning, CBR research has also investigated extensions of the traditional concept of a case by considering generalized knowledge structures (Schank and Abelson 1977; Bareiss 1989). While a traditional case is a single point in the representation space, a generalized case (Zito-Wolf and Alterman 1992; Bergmann and Vollrath 1999; Maximini, Maximini, and Bergmann 2003) covers a subspace of it. As a single generalized case has an increased coverage, it immediately provides solutions to a set of closely related problems rather than to a single problem only. Hence, by learning such generalized cases from the case base, adaptation knowledge can be determined and thus a larger spectrum of new problems can be solved. Therefore, the generalized case must be specialized after retrieval according to the specific circumstances expressed in the query. In this regard, generalization and specialization are transformations used for case adaptation. Additionally, retrieval performance can be improved as only the generalized case has to be stored, reducing the size of the case base.

Previous work on generalized cases mainly focussed on attribute-value or object-oriented representations. In this paper, we introduce the idea of generalized cases into process-oriented CBR (POCBR) (Minor, Montani, and Recio-Garca 2014), which deals with CBR applications for process-oriented information systems. POCBR aims at assisting domain experts in their work with workflows, in particular by supporting workflow reuse. Two important problems of workflow reuse are the retrieval of similar workflows from potentially large repositories (Bergmann and Gil 2014) as well as the adaptation of workflows (Müller and Bergmann 2014). In this paper, we introduce the concept of generalized cases into the context of POCBR as a new approach to address both issues. We present the formal foundations for the generalization of workflow cases as well as a new algorithm for generalizing semantic workflows. Further, the specialization of workflows w.r.t. a current query is described. Generalization and spezialization are based on a light-weight ontology of data and task items only. In particular, the additional acquisition of specific adaptation knowledge for workflows is avoided.

Process-Oriented Case-Based Reasoning

We now briefly introduce relevant previous work in the field of POCBR and illustrate it in the domain of cooking recipes, which are represented as workflows.

Broadly speaking, a workflow consists of a set of *activities* (also called *tasks*) combined with *control-flow structures* like sequences, parallel (AND split/join) or alternative (XOR split/join) branches, and loops. Tasks and control-flow structures form the *control-flow*. In addition, tasks exchange certain *products*, which can be of physical matter (such as ingredients for cooking tasks) or data. Tasks, products, and relationships between the two of them form the *data flow*. Today, graph representations for workflows are widely used. We use a workflow representation based on semantically labeled graphs (see Fig. 1) as introduced by Bergmann and Gil (2014).

Definition 1 *A workflow is a directed graph* $W = (N, E, S, T)$ *where* N *is a set of nodes and* $E \subseteq N \times N$ *is a set of edges. The function* T *assigns each node and edge a type. Further, nodes have a semantic description from a semantic meta data language* Σ*, which is assigned by the function* $S : N \to \Sigma$.

For this work, Σ is defined by domain specific light-weight

ontologies, restricted to a taxonomical representation of terms. In particular, we use one ontology for tasks and one ontology for data items. In the cooking domain, the task ontology organizes the various cooking steps in a taxonomical order and the data ontology represents the ingredients.

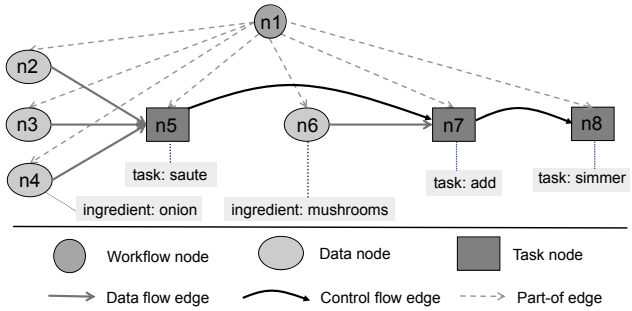

Figure 1: Simple workflow graph with node and edge types

Figure 1 shows a simple fragment of a workflow graph from the cooking domain with different types of nodes and edges. The graph for a workflow has one workflow node. The task nodes and data nodes represent tasks and data items, respectively. The data-flow edge is used to describe the linking of the data items consumed and produced by the tasks. The control-flow edge is used to represent the control flow of the workflow, i.e., it links tasks with successor tasks or control-flow elements.

Taxonomies

For the semantic annotation of workflow nodes and as background-knowledge for the generalization process, we use a taxonomical representation of terms as defined below:

Definition 2 *A taxonomy ψ is a tree of a partially ordered set of semantic terms $\Gamma = \{\gamma_0, \ldots, \gamma_n\}$, whereas $\gamma_i \sqsubset \gamma_j$ denotes that γ_j is more general than γ_i (γ_i is more specific than γ_j). Further, $\gamma_i \sqsubseteq \gamma_j$ holds, iff $\gamma_i \sqsubset \gamma_j \vee \gamma_i = \gamma_j$. A taxonomy consists of a single most general term (root term) denoted by $\bar{\psi} \in \Gamma$, i.e. $\nexists \gamma' \in \Gamma : \bar{\psi} \sqsubset \gamma'$. The set of most specific terms (leaf terms in the tree) is denoted by $\underline{\psi} = \{\gamma \in \Gamma | \nexists \gamma' \in \Gamma : \gamma' \sqsubset \gamma\}$.*

In the following, we further introduce some notations regarding the taxonomy.

Definition 3 *$\Gamma_{\downarrow}(\gamma)$ defines the one-step specializations of term γ, i.e., $\Gamma_{\downarrow}(\gamma) = \{\gamma_x \in \Gamma | \gamma_x \sqsubset \gamma \wedge \nexists \gamma_y \in \Gamma : \gamma_x \sqsubset \gamma_y \sqsubset \gamma\}$. Likewise $\Gamma_{\uparrow}(\gamma)$ defines the one-step generalization of term γ, i.e., $\Gamma_{\uparrow}(\gamma) = \gamma_x \in \Gamma$ s.t. $\gamma \in \Gamma_{\downarrow}(\gamma_x)$. Let further $LCA(\gamma_1, \gamma_2) \in \Gamma$ be the lowest common ancestor (most specific generalization) of $\gamma_1, \gamma_2 \in \Gamma$ in taxonomy ψ, i.e., $LCA(\gamma_1, \gamma_2) = \gamma \in \Gamma$ s.t. $\gamma_1 \sqsubseteq \gamma \wedge \gamma_2 \sqsubseteq \gamma \wedge \nexists \gamma' : \gamma' \sqsubset \gamma \wedge \gamma_1 \sqsubseteq \gamma' \wedge \gamma_2 \sqsubseteq \gamma'$.*

We use two distinct taxonomies as meta data language Σ, one for the task nodes ψ_{tasks} (preparation steps) and one for the data nodes ψ_{data} (ingredients), i.e., $\Sigma = \Gamma_{\psi_{\text{tasks}}} \cup \Gamma_{\psi_{\text{data}}}$. Hence, the function S assigns to each node $n \in N$ an appropriate term from $\Gamma_{\psi_{\text{tasks}}}$ or $\Gamma_{\psi_{\text{data}}}$.[1] As an example, figure 2 shows the ingredients taxonomy, in which the following holds: $beef \sqsubset meat$ and $sidedish = \Gamma_{\uparrow}(rice) \in \Gamma_{\downarrow}(vegeterian)$. Please note that in taxonomies, the leaf nodes represent concrete entities that may occur in executable workflows. For example, a recipe may include potatoes and beef as potential ingredients, but usually not terms from the inner nodes, such as vegetarian or meat. An inner node γ represents a generalized term that stands for the set of most specific terms below it. For example, the generalized term $vegeterian$ stands for the set $\{potatoes, rice, noodles\}$. Further on in the paper we use inner nodes in generalized workflows to represent that an arbitrary ingredient from the set of its specializations can be chosen.

[1] We omit the index if it is obvious which ontology is referenced.

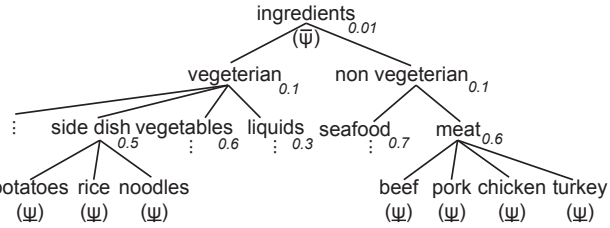

Figure 2: Example for a data taxonomy

Semantic Similarity

For workflow retrieval in general as well as to identify a set of similar workflows used for generalization, we apply the semantic workflow similarity framework by Bergmann & Gil (2014). This framework extends traditional approaches to similarity in CBR and allows to model similarity measures which are inline with experts assessments. The similarity model is based on a local similarity measure for the terms assigned to the nodes of the workflow. We apply the taxonomy similarity approach by Bergmann (1998) to model the similarity between two taxonomy nodes. Thus, each term that is not a leaf term in the taxonomy $\gamma \in \psi \backslash \underline{\psi}$ is annotated with a similarity value sim_{ψ} defining the similarity between all child terms of γ, (e.g, $sim_{\psi}(\text{meat}) = 0.6$ in Fig. 2). The similarity $sim_{\psi}(\gamma_x, \gamma_y)$ between two terms $\gamma_x, \gamma_y \in \psi$ reflects the closeness in the ontology using the similarity value of the lowest common ancestor (see (Bergmann 1998) for further details):

$$sim_{\psi}(\gamma_x, \gamma_y) = \begin{cases} 1 & \text{, if } \gamma_x \sqsubseteq \gamma_y \\ sim_{\psi}(LCA(\gamma_x, \gamma_y)) & \text{, otherwise} \end{cases}$$

The similarity $sim_N : N^2 \rightarrow [0,1]$ of two nodes is then defined based on sim_{ψ}, i.e., $sim_N(n_1, n_2) = sim_{\psi}(S(n_1), S(n_2))$. The similarity $sim(QW, CW)$ between a query workflow QW and a case workflow CW is defined by means of an admissible mapping $m : N_q \cup E_q \rightarrow N_c \cup E_c$, which is a type-preserving, partial, injective mapping function of the nodes and edges of QW to those of CW. This means that nodes are only mapped to nodes of the same type and edges are only mapped if their corresponding nodes are mapped as well. Partial means that not all nodes of the case workflow must occur in the image of the mapping.

For each query node x mapped by m, the similarity to the respective case node is computed by $sim_N(x, m(x))$. The overall workflow similarity with respect to a mapping m, named $sim_m(QW, CW)$ is then computed by an aggregation function (we use the average) combining the previously computed similarity values. Finally, the overall workflow similarity is determined by the best possible mapping m, i.e., $sim(QW, CW) = \max\{sim_m(QW, CW) \,|\, \text{admissible map } m\}$. Thus, similarity assessment is defined as an optimization problem aiming at finding the best possible mapping, reflecting the best possible way to reuse the case workflow. This mapping is also referred to as $m_{\max}$. In general this similarity measure assesses how well the query workflow is covered by the case workflow. In particular, the similarity is 1 if the query workflow is exactly included in the case workflow as a subgraph. Please note that this similarity measure is not symmetrical.

Generalized Cases for POCBR

We now introduce the concept of generalized cases into POCBR. A generalized case becomes a generalized workflow, which stands for a set of specific workflows. We define when a workflow is a generalization of another workflow and propose algorithms for generalizing and specialization.

Generalization of Workflows

We now define when a workflow is a generalization of an other workflow w.r.t. the defined domain taxonomies.

Definition 4 *A workflow W^* is a generalization of the workflow W (we write $W \sqsubseteq W^*$), iff there exists a graph isomorphism $I : N \to N^*$ between the Graphs W and W^* such that $\forall n \in N : S(n) \sqsubseteq S(I(n))$.*

Moreover, we introduce the following notations.

Definition 5 *Two workflows W_1, W_2 are equivalent, denoted by $W_1 \equiv W_2$, iff $W_1 \sqsubseteq W_2 \wedge W_2 \sqsubseteq W_1$. Furthermore, $W \sqsubset W^*$ denotes that W^* is a strict generalization of workflow W, iff $W \sqsubseteq W^* \wedge W^* \not\sqsubseteq W$.*

Algorithm for Workflow Generalization We now introduce an algorithm for learning generalized workflows from a case base of specific workflows. In order to guide the generalization, we use the taxonomy as background knowledge, which provides the vocabulary from which generalized tasks and data items are selected as well as the generalization relation among them. The generalization of a workflow is based on a comparison with similar workflows from the case base CB. The assumption is that if similar workflows contain similar terms from the taxonomy, then these terms can be inductively generalized. Hence, only reasonable generalizations should be learned in order to ensure adaptation quality.

A similarity threshold parameter $\Delta \in [0, 1]$ is introduced that defines which workflows are compared. In particular, each workflow W is compared with the workflows $CB_W \subseteq CB$ that have a similarity value of at least Δ, i.e., $CB_W = \{W' \in CB | W' \not\equiv W \wedge sim(W, W') \geq \Delta\}$. For instance, lets assume that the workflows W_1 and W_2, which are partially illustrated in Figure 3, are similar enough to W to be contained in CB_W regarding a similarity threshold Δ.

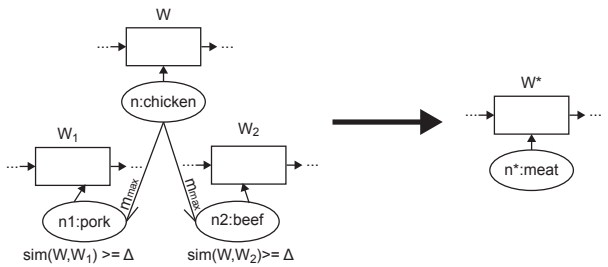

Figure 3: Example generalization

As we apply the introduced similarity framework, for each workflows in CB_W the best possible mapping m_{max} is also available as a side effect of the computation of CB_W. This mapping allows to link workflow nodes form W to the related nodes from the workflows in CB_W, defined as follows: $\Omega(n) = \{\gamma' | \exists (N', E', S', T') \in CB_W \, \exists n' \in N' : m_{max}(n) = n' \wedge \gamma' = S(n')\} \cup S(n)$. For the example in Figure 3, it holds: $\Omega(n) = \{\text{pork}, \text{beef}, \text{chicken}\}$.

According to the definition of the mapping, nodes are only mapped to nodes of the same type, thus the annotated terms are all within the same taxonomy. Further, the overall workflow similarity, which also includes the mapping of edges, makes sure that the "context" of nodes is regarded as well, i.e., the control and data flow in which the nodes occur.

Now, we introduce a second threshold parameter $\Delta_\psi \in [0, 1]$ to specify whether the term $\gamma = S(n)$ of the node n is generalized to its one-step generalization $\gamma^* = \Gamma_\uparrow(\gamma)$. This threshold regards two criteria: the similarity value of the one-step generalization $sim_\psi(\gamma^*)$ and the fraction $\varphi(\gamma^*)$ of possible one-step specializations of γ^* that have been found in CB_w, i.e., $\varphi(\gamma^*) = \frac{|\{\gamma' | \gamma' \in \Omega(n) \wedge \gamma' \in \Gamma_\downarrow(\gamma^*)\}|}{|\{\gamma' | \gamma' \in \Gamma_\downarrow(\gamma^*)\}|}$.

Considering the annotated similarity value $sim_\psi(\gamma^*)$ ensures that terms that are more similar to each other are more likely generalized. Regarding $\varphi(\gamma^*)$ allows to restrict generalization only to situations, in which enough inductive evidence is found in the case base that the generalization is valid. We combine both criteria into a single one and apply the threshold Δ_ψ to determine whether a term is generalized to its direct generalization, i.e., if $\frac{sim_\psi(\gamma^*) + \varphi(\gamma^*)}{2} \geq \Delta_\psi$ holds. Furthermore, for each node the original specialized term is stored within $OrigSpec(n)$, if the term is a leaf node (in order to retain the most specialized term), which may be used later during specialization. This generalization step is executed for all nodes $n \in N$. Algorithm 1 shows the complete algorithm for workflow generalization.

Algorithm for Case Base Generalization A case base is generalized by generalizing each workflow $W \in CB$ and storing it in a new case base CB^*. The generalized workflow is only stored if CB^* does not already contain a workflow which is even more general than the workflow to be stored. Thus, only the most generalized workflows are retained, which may represent multiple original workflows. The iterative generalization is repeated until no more workflow can be generalized (see Alg. 2).

Algorithm GENERALIZE_WF(W,CB);
Input: Workflow $W = (N, E, S, T)$ and Case base CB
Output: Generalized Workflow W^*
Compute CB_W;
forall the $n \in N$ **do**
  Compute $\Omega(n)$;
  $\gamma \leftarrow S(n)$;
  $\gamma^* \leftarrow \Gamma_{\uparrow}(\gamma)$;
  if $\frac{sim_\phi(\gamma^*)+\varphi(\gamma^*)}{2} \geq \Delta_\psi$ **then**
    $S(n) \leftarrow \gamma^*$;
    if $\gamma \in \underline{\psi}$ **then**
      $OrigSpec(n) \leftarrow \gamma$;
return W

Algorithm 1: Workflow generalization

Algorithm GENERALIZE_CB(CB) ;
Input: Case Base CB
Output: Generalized Case Base CB^*
generalization $\leftarrow$ true ;
while *generalization* **do**
  generalization $\leftarrow$ false ;
  $CB^* \leftarrow \emptyset$;
  forall the $W \in CB$ **do**
    $W^* \leftarrow GENERALIZE\_WF(W, CB)$;
    if $\nexists W' \in CB^* : W^* \sqsubseteq W'$ **then**
      $CB^* \leftarrow CB^* \cup W^*$;
      if $W^* \not\equiv W$ **then**
        generalization $\leftarrow$ true ;
  $CB \leftarrow CB^*$;
return CB

Algorithm 2: Generalization of a Case Base

Specialization of Workflows

When reasoning with generalized workflows, specialization is required to turn a retrieved generalized workflow from CB^* into a specific workflow, which is executable. The resulting specific workflow is usually different from the orignial workflow that was generalized, thus it can be considered an adaptation of it. To specialize a workflow means to determine a specific workflow according to the following definition, such that it is most similar to the query at hand.

Definition 6 *The set of specific workflows W_+ of a workflow W is defined as $W_+ = \{W' | W' \sqsubseteq W \wedge \forall n' \in N' : S(n') \in \underline{\psi}\}$. Further, we call $|W_+|$ the coverage of W.*

Ideally, the best specific workflow is the most similar workflow w.r.t. the query, selected from the union of all specific workflows that can be derived from all generalized cases in the case base, i.e., from $\bigcup_{W \in CB^*} W_+$. This, however, would be intractable and would diminish the effect of generalization in shrinking the case base to speed-up retrieval. To avoid the search in the set of specific workflows, we propose an alternative algorithm also being able to identify the best specialization. Therefor, the workflow with the highest similarity in $W \in CB^*$ is retrieved and specialized

Algorithm SPEZIALIZE_WF(W, Q) ;
Input: Generalized Workflow $W^* = (N^*, E^*, S^*, T^*)$,
  Query $Q = (N_q, E_q, S_q, T_q)$
Output: Specialized Workflow $\in W_+$
forall the $n^* \in N^*$ **do**
  if $\exists n_q \in N_q$ *s.t.* $n^* = m_{max}(n_q) \wedge S(n_q) \sqsubset S(n^*)$ **then**
    $S(n^*) \leftarrow S(n_q)$;
  else
    $S(n^*) \leftarrow OrigSpec(n^*)$;
return W

Algorithm 3: Specialization of a case

node by node, aiming at maximizing each node similarity w.r.t. the related node in the query (see Alg. 3). For this purpose, we again exploit the best possible mapping of query nodes to the nodes of the most similar generalized case, produced as a side effect of the semantic similarity computation during retrieval. This again ensures that the "context" of the nodes is regarded.

Let's assume a node n_q of query Q is mapped to a node n^* in the generalized workflow W^*, i.e., $m_{max}(n_q) = n^*$. If the query node is more specific than the case node, i.e., $n_q \sqsubseteq n^*$, then the generalized node n^* is specialized directly to the specific query term. Otherwise, we chose the respective value from the original case, which is stored for this purpose in $OrigSpec(n^*)$. This avoids the impact of potential overgeneralization. Further, this procedure does not influence the mapping of the similarity and has no impact on the similarity values as $sim_\psi(S(n_q), S(n^*)) = sim_\psi(S(n_q), S(n_q)) = 1$. In the other case, i.e., if $n_q \not\sqsubseteq n^*$, any arbitrary specialization of the node n^* can be chosen as this would also have no impact on the similarity or the mapping as for a specialized term $\gamma_+ \in \Gamma_{\downarrow}(S(n^*)) : sim_\psi(S(n_q), S(n^*)) = sim_\psi(S(n_q), \gamma_+) = sim_\psi(LCA(S(n_q), S(n^*))) = sim_\psi(LCA(S(n_q), \gamma_+))$. Thus, the specialization algorithm produces a specific workflow with the same similarity as the most similar workflow from $\bigcup_{W \in CB^*} W_+$.

Empirical Evaluation

The described approach is fully implemented and evaluated focussing on two hypotheses reflecting the expected benefits of generalized cases for POCBR.

H1. The adaptation by specialization of a generalization workflows leads to a workflow with a quality similar to the original workflow that was generalized.

H2. Case base generalization reduces the size of the case base and thus leads to a reduction of retrieval time if similar and structural identical workflows are present.

We manually constructed 60 pasta recipe workflows from the textual recipe descriptions on www.studentrecipes.com with an average size of 25 nodes and 64 edges. Altogether, they contain 162 different ingredients and 67 tasks. For ingredients and tasks, a taxonomy was manually constructed.

The extracted workflows, contained AND, XOR, as well as LOOP structures. We randomly chose a set of 10 workflows QCB which we use as query workflows. The remaining set of 50 workflows is used as the case base CB during the evaluation. We examined the 50 workflows manually and discovered that they do not contain pairs of structurally identical workflows. Thus, for the evaluation of hypothesis H2, we additionally generated a second case base of 70 workflows VCB containing the 50 recipes of CB as well as 20 automatically generated variations based on a randomly chosen recipe. The variations were generated by replacing each node term of the random workflow with a probability of 0.3 by a random but similar leaf sibling term of the taxonomy. Further, it was ensured that at least 3 terms were replaced. Thus, a case base of 70 recipes was created containing similar workflows with an identical structure.

All experiments were executed on a PC with an Intel Core i7-870 CPU @ 2.93 GHz and 8 GB RAM running Windows-7 Enterprise 64-bit. If not otherwise stated, we chose the parameter $\Delta_\psi = 0.5$, which means that if either all sibling terms have been mapped or the similarity of the parent term is 1, generalization is executed. We also choose $\Delta = 0.5$, which means that at least half of the workflow elements must be identical to be considered during generalization.

Experimental Evaluation and Results

To evaluate Hypothesis H1 a blinded experiment was performed involving 5 human experts. The experts compared the quality of 10 pairs of a specialized workflow WF_S and an original workflow WF_O. The original workflow is the corresponding workflow from CB which was the basis workflow of the generalization of WF_S. The workflow of the pairs were presented in random order, so the experts did not know which workflow was the specialized workflow. The experts compare the two workflows of the pairs based on a scale from -3 to +3 (0 means equal quality). The experts rated the criteria correctness of the preparation[2], as well as the culinary quality. The ratings from the 5 experts of all 10 workflow pairs were acquired, leading to 50 overall ratings. We further computed an aggregated quality, which represents the average value of both item ratings.

Table 1: Item rating assessment

	better WF_O	better WF_S	equal quality
correctness of preparation	12	3	35
culinary quality	23	13	14
aggregated quality	22	15	13

Table 1 illustrates the number of workflows for which the original workflow or the specialized workflow was rated better, as well as the number of workflows which were rated equally. It shows that in only 24% of the cases, the adaptation by generalization and specialization reduces the correctness of the workflow. The culinary quality was negatively affected in 46%, the aggregated quality in 44% of the cases. We also investigated the score value of the ratings. We computed the average value on the item ratings, which was 0.24 for the correctness, 0.26 for the culinary quality, and 0.25 for the aggregated quality in favor of the original workflows. Additionally, a paired t-test on the aggregated quality showed that the quality difference between the original and the adapted workflows is statistically not significant ($p = 0.11$). Overall, this mostly confirms hypothesis H1.

[2]i.e. cutting oil would violate the correctness

Table 2: Case base evaluation

casebase	size	coverage	avg. retrieval time (s)
CB	50	50	1.39
VCB	70	70	1.71
CB^*	50	$\sim 6 \cdot 10^{10}$	1.33
VCB^*	58	$\sim 2 \cdot 10^{13}$	1.49

To verify hypothesis H2 we generalized the case bases CB and VCB leading to the case bases CB^* and VCB^*. We then searched for the most similar workflows using the 10 queries of QCB within all 4 case bases. If the retrieved workflow was a generalized workflow, it was also specialized according to the given query. The results are shown in Table 2. It can be seen that the size of VCB is reduced by about 17%, which leads to a reduction of the average retrieval time of about 13%. The retrieval time includes the time for the specialization of generalized workflows, which however only took about 6 ms in average. Thus, hypothesis H2 is confirmed. Further, Table 2 illustrates that generalization has produced generalized workflows with high coverage, thus leading to a highly reusability.

Table 3: Evaluation of the Δ parameter (CB^*/VCB^*)

Δ	size	coverage	avg. retrieval time (s)
0.7	50/65	$\sim 5 \cdot 10^{10}$/$\sim 8 \cdot 10^{10}$	1.32/1,61
0.6	50/65	$\sim 5 \cdot 10^{10}$/$\sim 8 \cdot 10^{10}$	1.32/1,62
0.5	50/58	$\sim 6 \cdot 10^{10}$/$\sim 2 \cdot 10^{13}$	1.33/1.49
0.4	50/53	$\sim 6 \cdot 10^{16}$/$\sim 4 \cdot 10^{17}$	1.26/1.35
0.3	50/50	$\sim 3 \cdot 10^{19}$/$\sim 9 \cdot 10^{18}$	1.39/1.26

Table 4: Evaluation of the Δ_ψ parameter (CB^*/VCB^*)

Δ_ψ	size	coverage	avg. retrieval time (s)
0.7	50/69	$\sim 3 \cdot 10^3$/$\sim 1 \cdot 10^4$	1.28/1.68
0.6	50/67	$\sim 7 \cdot 10^6$/$\sim 3 \cdot 10^7$	1.29/1.67
0.5	50/58	$\sim 6 \cdot 10^{10}$/$\sim 2 \cdot 10^{13}$	1.33/1.49
0.4	50/50	$\sim 4 \cdot 10^{21}$/$\sim 4 \cdot 10^{21}$	1.31/1.31
0.3	50/50	$\sim 2 \cdot 10^{25}$/$\sim 2 \cdot 10^{25}$	1.27/1.27

Additionally, we investigated the influence of the Δ parameter. It can be seen in Table 3 that for CB and VCB

more generalizations are produced if a lower similarity value between the workflows to be compared is enforced. Moreover, for VCB a lower Δ parameter also means that the size of the case base is further reduced. The evaluation of the Δ_ψ parameter leads to a similar result (see Table 4): the higher the value, the lower the number of generalizations produced for CB and VCB. Again, for VCB a lower Δ_ψ parameter further reduces the size of the case base.

Discussion and Related Work

We presented a novel approach for the generalization and specialization of workflow cases for POCBR and we evaluated this approach with respect to adaptation quality and retrieval performance. Approaches for generalizing cases have been already investiaged in CBR for various purposes (Maximini, Maximini, and Bergmann 2003) and they are used in various domains such as medical diagnosis (Schmidt et al. 2001) in planning (Kambhampati 1994; Sánchez-Ruiz and Ontanón 2014), or product design (Purvis and Pu 1995). In POCBR, the use of generalization was already proposed for the purpose of workflow adaptation (Gil 2008). Garijo et al. (2013) generalized scientific workflows based on taxonomies and a comparison of similar workflow fragments. However, their approach did not yet address the adaptation based on generalized workflows. From a machine learning point of view, reasoning with generalized cases moves CBR from the purely lazy learning approach towards an eager learning (or hybrid) approach. Unlike the use of generalization in an offline-phase during case-base construction, generalization is also used on demand as part of an adaptation approach. A recent example is Tuuurbine (Gaillard et al. 2014), which adapts cases stored in RDF format by generalization and specialization, guided by a generalization cost function and adaptation rules. Similar to generalization, abstraction is also discussed in the literature as an approach to reduce the size of the case base and to increase case coverage (Bergmann and Wilke 1996). However, abstraction is different from generalization as it would require reducing the overall granularity of workflows (e.g. less tasks and data items) and it would require abstract terms within the ontology. Our future work will also investigate workflow abstraction as well as approaches to use more knowledge-intensive ontologies, including not just taxonomic relations but also abstractions as well as attributes and constraints.

Acknowledgments

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

References

Bareiss, R. 1989. *Exemplar based knowledge acquisition: a unified approach to concept representati on, classification, and learning*. Academic Press Professional, Inc.

Bergmann, R., and Gil, Y. 2014. Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.* 40:115–127.

Bergmann, R., and Vollrath, I. 1999. Generalized cases: Representation and steps towards efficient similarity assessment. In *Proceedings of the 23rd Annual German Conference on Artificial Intelligence*, volume 1701 of *LNCS*, 195–206. Springer.

Bergmann, R., and Wilke, W. 1996. On the role of abstraction in case-based reasoning. In *Advances in Case-Based Reasoning, Third European Workshop, EWCBR-96*, volume 1168, 28–43. Springer.

Bergmann, R. 1998. On the use of taxonomies for representing case features and local similarity measures. In *Proceedings of the 6th German Workshop on Case-Based Reasoning,GWCBR-98*.

Gaillard, E.; Infante-Blanco, L.; Lieber, J.; and Nauer, E. 2014. Tuuurbine: A generic CBR engine over RDFS. In *Case-Based Reasoning Research and Development, ICCBR-14*, volume 8765 of *LNCS*. Springer. 140–154.

Garijo, D.; Corcho, O.; and Gil, Y. 2013. Detecting common scientific workflow fragments using templates and execution provenance. In *Proceedings of the 7th International Conference on Knowledge Capture*, K-CAP '13, 33–40. ACM.

Gil, Y. 2008. From data to knowledge to discoveries: Scientific workflows and artificial intelligence. *Scientific Programming* 16(4).

Kambhampati, S. 1994. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artif. Intell.* 67(1):29–70.

Maximini, K.; Maximini, R.; and Bergmann, R. 2003. An investigation of generalized cases. In Ashley, K. D., and Bridge, D. G., eds., *Case-Based Reasoning Research and Development, ICCBR-03*, volume 2689 of *LNCS*, 261–275. Springer.

Minor, M.; Montani, S.; and Recio-Garca, J. A. 2014. Process-oriented case-based reasoning. *Information Systems* 40(0):103 – 105.

Müller, G., and Bergmann, R. 2014. Workflow streams: A means for compositional adaptation in process-oriented CBR. In *Case-Based Reasoning Research and Development, ICCBR-14*, volume 8765 of *LNCS*. Springer. 315–329.

Purvis, L., and Pu, P. 1995. Adaptation using constraint satisfaction techniques. In *Case-Based Reasoning Research and Development, ICCBR-95*, LNCS, 289–300. Springer.

Sánchez-Ruiz, A. A., and Ontanón, S. 2014. Least common subsumer trees for plan retrieval. In *Case-Based Reasoning Research and Development, ICCBR-14*. Springer. 405–419.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Schmidt, R.; Montani, S.; Bellazzi, R.; Portinale, L.; and Gierl, L. 2001. Cased-based reasoning for medical knowledge-based systems. *International Journal of Medical Informatics* 64(2):355–367.

Zito-Wolf, R., and Alterman, R. 1992. Multicases: A case-based representation for procedural knowledge. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 331–336.