Semi-Automatic Adaptation of Mappings between Life Science Ontologies

Anika Groß[1], Julio Cesar Dos Reis[2,3], Michael Hartung[1], Cédric Pruski[2], and Erhard Rahm[1]

[1] Department of Computer Science, University of Leipzig, Germany
[2] CR SANTEC, Public Research Centre Henri Tudor, Luxembourg
[3] LRI, University of Paris-Sud XI, France
{gross,hartung,rahm}@informatik.uni-leipzig.de,
{julio.dosreis,cedric.pruski}@tudor.lu

Abstract. The continuous evolution of life science ontologies requires the adaptation of their associated mappings. We propose two approaches for tackling this problem in a largely automatic way: (1) a *composition-based adaptation* relying on the principle of mapping composition and (2) a *diff-based adaptation* algorithm individually handling change operations to update the mapping. Both techniques reuse unaffected correspondences, and adapt only the affected mapping part. We experimentally assess and confirm the effectiveness of our approaches for evolving mappings between large life science ontologies.
Keywords: mapping adaptation, mapping migration, mapping evolution, ontology evolution, ontology mapping, ontology alignment

1 Introduction

Ontologies and their applications have become increasingly important especially in the life sciences [1,2]. Typically there are many ontologies within a domain with overlapping information, *e.g.*, more than 30 anatomy-related ontologies in the OBO foundry [3]. Mappings between such related ontologies are useful for various data integration and enhanced analysis tasks. For instance, mappings are needed to merge several ontologies into an integrated ontology, *e.g.*, the multi-species anatomy ontology Uberon [4]. While manually curated mappings are especially valuable to interrelate the concepts of ontologies, it is often too time-consuming for large ontologies. Hence, semi-automatic matching approaches are increasingly needed for mapping creation [5,6,7].

The life sciences are a very dynamic field and new research results lead to a continuous evolution of ontologies so that new versions are periodically released [8]. Ontology changes include the addition, revision or deletion of concepts and relationships, and their frequency may substantially vary between ontologies or different parts of one ontology [9]. Ontology evolution can have an impact on different dependent artifacts such as ontology mappings [10,11], annotation mappings [12,13] and ontology-based queries [14,15]. As mappings may become invalid and out-dated their adaptation is required. For example, a new version

of an ontology in Bioportal [16] or UMLS [17] may require the adaptation of the associated mappings, so that users and dependent applications can consume the most recent ones.

In this paper, we study different methods for a largely automatic adaptation of ontology mappings. In particular, we aim to avoid an expensive redetermination of the complete mapping and to reuse all stable parts from the old mapping. Migrating ontology mappings is not trivial for complex ontology changes such as the split of a concept into several new concepts. In this case an earlier correspondence with the unsplit concept may have to be changed to another or several new correspondences, and an expert user should be supported to select the correct result. Each type of ontology change may require different actions to update an ontology mapping. There is only little research so far on how to best perform the adaptation of mappings (see Sec. 2). Typically, previous approaches did not consider the impact of different ontology changes on mappings and also ignored new correspondences introduced by added concepts.

We therefore make the following contributions:

- We present a *composition-based* approach that uses ontology matching to create mappings between versions of an evolved ontology as well as the principle of mapping composition to create the adapted ontology mapping (Sec. 4).
- We propose a *diff-based* approach relied on a diff result consisting of the set of changes that led from the old to the new version of an ontology. The approach uses a library of change handlers to realize change-specific mapping adaptations (Sec. 5).
- We evaluate the approaches by adapting mappings between three large life science ontologies extracted from UMLS. Results reveal that we can adapt mappings largely automatically. We can also suggest specific mapping adaptations for certain types of ontology changes to simplify mapping curation (Sec. 6).

Additionally, we discuss related work in Sec. 2, present preliminaries on ontologies, mappings and the change model in Sec. 3, and conclude in Sec. 7.

2 Related Work

While a significant amount of research has already coped with the evolution of ontologies [18], the evolution of dependent mappings has received relatively little attention. In the context of schema evolution and model management [19,20], it has been proposed to evolve a previously determined mapping by composing it with a match mapping between the old and the new version of an updated schema or model. This composition approach has been explored in [21] for schema mappings and was shown to avoid the full re-calculation of existing mappings. We investigate and enhance the composition approach for adapting ontology mappings by not only reusing stable parts of the previous mapping, but by also extending the mapping, *e.g.*, for added ontology concepts.

Only few studies specifically investigated the maintenance and evolution of ontology mappings. In [22] the use of reasoners has been proposed for detecting and repairing invalid correspondences after ontology changes. Khattak *et al.* [23] propose to re-compute only those correspondences associated with changed ontology elements. Martins & Silva [24] propose that mapping evolution should behave similarly to strategies applied for ontology evolution. However, correspondences are only adapted when concepts are removed from the ontology. Kondylakis & Plexousakis [14] focus on the automatic detection of queries affected by ontology evolution. They assist developers to find and adapt invalid queries by suggesting sequences of changes affecting such queries.

In our previous work, we empirically analyzed which ontology changes lead to the addition or deletion of correspondences in an ontology mapping [11]. Dos Reis *et al.* [10] have proposed a framework for mapping evolution highlighting the role of different types of ontology changes for mapping adaptation, as well as the importance of considering different semantic types of correspondences in the adaptation process.

In contrast to prior studies, we not only aim at reusing stable parts of previous ontology mappings, but also extend the mappings for new ontology concepts. In addition to a composition-based method we propose a diff-based approach to individually handle different types of ontology changes and to solicit user feedback on adapted and newly determined correspondences. Unlike previous studies, we also evaluate the quality of the adapted mappings for large life science ontologies.

3 Preliminaries

We first define the considered ontology and mapping model (Sec. 3.1) and then describe the general scenario we investigate in this paper (Sec. 3.2).

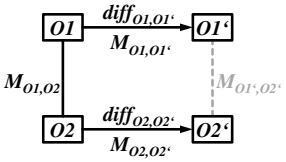
3.1 Ontology Versions and Mappings

An ontology $O = (C, R, A)$ consists of a set of concepts C interrelated by directed relationships R. Each concept $c \in C$ is identified by an unambiguous accession number c_{acc}. Further attributes $a \in A$ describe a concept in more detail, *e.g.*, labels, synonyms or definitions. A special attribute *obsolete* indicates whether a concept is outdated and should thus not be used anymore. A relationship $r \in R$ interconnects two concepts and has a specific type, *e.g.*, 'is_a' or 'part_of'. An ontology version is a release of O, *i.e.*, a particular version is valid until a newer version becomes available. In the following, we denote two versions of an evolved ontology with O (old version) and O' (new version), respectively.

An ontology mapping $M_{O1,O2}$ interconnects concepts of two different ontologies $O1/O2$ by so-called correspondences:

$$M_{O_1,O_2} = \{(c_1, c_2, sim, semType, status) | c_1 \in O_1, c_2 \in O_2, sim \in [0,1], \\ semType \in \{=, \leq, \geq, \approx\}, \\ status \in \{"handled", "toverify"\}\}$$

(a)



(b)

Change operation	Description
addC(c), delC(c)	addition/deletion of concept *c*
toObsolete(c), revokeObsolete(c)	set/revoke ‚to obsolete' status of *c*
split(s,T)	split a source concept *s* into several target concepts *T*
merge(S,t)	merge several source concept *S* into one target concept *t*
substitute(c,c')	Substitute concept *c* by concept *c'*
move(c,P,P')	move a concept *c* from parents *P* to parents *P'*
addR(r), delR(r)	addition/deletion of a relationship *r*
chgAttValue(c,att,v1,v2)	change value of *att* in *c* from *v1* to *v2*
addA(a)/delA(a)	Addition/deletion of an attribute *a*

Fig. 1. (a) General scenario. (b) Considered change operations of COnto-Diff.

A correspondence $(c_1, c_2, sim, semType, status)$ interrelates two concepts $c_1 \in O1$ and $c_2 \in O2$. We use three further independent attributes to describe a correspondence in more detail. The sim value represents the similarity measure between c_1 and c_2. The higher the value, the more related are both concepts. We assign a similarity of 1 to manually created correspondences. We further use a $semType$ to differentiate the semantic connection type. For instance, concepts can be equivalent (*e.g.*, 'torso'='trunk'), one concept can be less or more general than the other (*e.g.*, 'thumb'$\leq$'finger') or concepts can be somehow related ($\approx$). A *status* signals the state of the correspondences during adaptation. In particular, a correspondence can be adapted (*handled*) or needs verification by an expert (*to verify*).

To create new mappings between ontologies we rely on semi-automatic match strategies because a purely manual mapping generation has become increasingly infeasible for large and complex ontologies [6,7]. For this purpose we use a successfully applied match strategy based on a concept's name and synonyms described in [25].

We also support the inversion of ontology mappings, *e.g.*, to get a mapping M_{O_2,O_1} out of M_{O_1,O_2}. To this end, we will use an inverse operator that inverts each correspondence as follows: $(c_1, c_2, sim, semType, status) \mapsto (c_2, c_1, sim, new\ SemType, status)$. In particular, the order of matching concepts is reversed, the similarity and the status values remain unchanged. The $semType$ is adapted using the following rules: $=\mapsto=$, $\leq\mapsto\geq$, $\geq\mapsto\leq$ and $\approx\mapsto\approx$.

3.2 General Scenario and Change Model

The general scenario investigated in this paper is depicted in Fig. 1a. There are two ontologies in their old ($O1$,$O2$) and new versions ($O1'$,$O2'$). A mapping $M_{O1,O2}$ interconnects the old versions of the two ontologies. The task investigated is to determine the new mapping $M_{O1',O2'}$ which interrelates concepts of the new ontology versions $O1'$ and $O2'$. For this purpose, we need further mappings between the ontology versions involved. In particular, there are two mappings $M_{O1,O1'}$ and $M_{O2,O2'}$ which interconnect concepts between the versions. These mappings provide information about how concepts in an old version

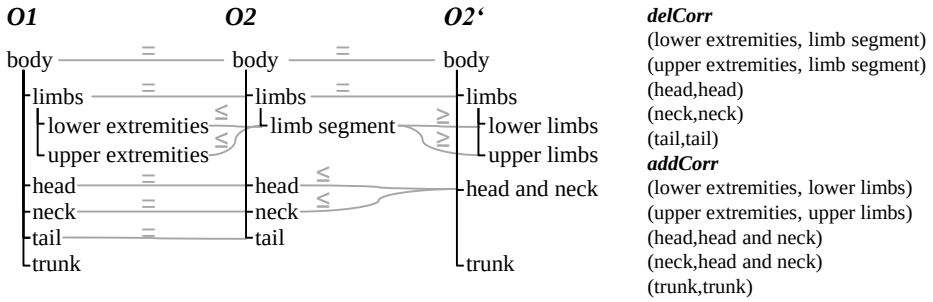


Fig. 2. Mapping evolution example.

are related with concepts in the new version. We generate these mappings by matching, *i.e.*, we match $O1$ with $O1'$ and $O2$ with $O2'$, respectively. The proposed composition-based approach (Sec. 4) uses the mappings $M_{O1,O2}$, $M_{O1,O1'}$ and $M_{O2,O2'}$ to create the adapted mapping $M_{O1',O2'}$ based on composition.

We further use so-called evolution mappings ($diff_{O1,O1'}$ and $diff_{O2,O2'}$) between the old and new ontology versions. These mappings integrate all changes that occurred during evolution from $O1$ to $O1'$ and $O2$ to $O2'$, respectively. An evolution mapping can be created using a Diff tool such as PromptDiff [26] or COnto-Diff [27] and contains different types of changes (Fig. 1b lists changes of COnto-Diff). For instance, there are concept changes such as add, delete, merge and split, or changes of attribute values. The proposed diff-based approach (Sec. 5) uses the diff evolution mappings $diff_{O1,O1'}$ and $diff_{O2,O2'}$ to create the adapted mapping $M_{O1',O2'}$.

4 Composition-based Adaptation

This section presents the composition-based approach for mapping adaptation. Its strength is the reuse of the previous, already validated ontology mapping to avoid an expensive re-computation of confirmed correspondences. Given that changes are typically limited to a small subset of ontologies, this promises that the largest part of the new mapping is easily determined. For illustration purpose, we use a running example shown in Fig. 2 with an evolution of an anatomy ontology ($O2 \mapsto O2'$). The ontology changes require an adaptation of the mapping $M_{O1,O2}$, in particular to delete the previous correspondence (*delCorr*) and to add the new correspondence (*addCorr*) shown on the right side. Our composition-based approach achieves the adaptation by composing the previous ontology mapping $M_{O1,O2}$ with the mapping $M_{O2,O2'}$, as well as by checking whether added concepts lead to new correspondences.

The composition of two mappings $M_{A,B}$ and $M_{B,C}$ generates a mapping $M_{A,C}$ between A and C. With mappings as introduced in Sec. 3.1, we define:

$$M_{A,C} = \mathsf{compose}(M_{A,B}, M_{B,C}) = M_{A,B} \circ M_{B,C} =$$
$$\{(c_1, c_2, aggSim(sim_1, sim_2), getNewType(semType_1, semType_2),$$

$$getNewStatus(semType_1, semType_2))|$$
$$c_1 \in A, c_2 \in C, b \in B : \exists(c_1, b, sim_1, semType_1, status_1) \in M_{A,B} \wedge$$
$$\exists(b, c_2, sim_2, semType_2, status_2) \in M_{B,C}\}$$

The generation of a correspondence (c_1, c_2) in $M_{A,C}$ requires the existence of two correspondences (c_1, b) and (b, c_2) connecting to the same concept $b \in B$. The attribute values of the new correspondence are derived from the values of the two 'connecting' correspondences. First, the new similarity is aggregated from the similarities sim_1 and sim_2 by computing, *e.g.*, their average or maximum (aggSim). Second, the new semantic type is derived from $semType_1$ and $semType_2$ (getNewType) based on the rule set presented in Fig. 4a. For example, the combination of '=' and '$\leq$' would lead to the new semantic type '$\leq$'. Third, the new correspondence is assigned the new status (getNewStatus, see Sec.5.2).

CompAdapt (Algorithm 1) shows how we perform composition-based mapping adaption for the general case when both ontologies evolve ($O1 \mapsto O1'$, $O2 \mapsto O2'$). The algorithm uses as input the previous ontology mapping $M_{O1,O2}$ as well as the two mappings $M_{O1,O1'}$ and $M_{O2,O2'}$.

Algorithm 1: CompAdapt($M_{O1,O2}$,$M_{O1,O1'}$,$M_{O2,O2'}$)

```
1 M_{O1',O1} ← inverse(M_{O1,O1'});
2 M_{O1',O2} ← compose(M_{O1',O1},M_{O1,O2});
3 M_{O1',O2'} ← compose(M_{O1',O2},M_{O2,O2'});
4 return M_{O1',O2'};
```

We first generate the inverse mapping $M_{O1',O1}$ (line 1) and compose it with $M_{O1,O2}$ to create an intermediate mapping between $O1'$ and $O2$ (line 2). We then transitively compose the intermediate mapping with $M_{O2,O2'}$ to produce the adapted mapping $M_{O1',O2'}$ between $O1'$ and $O2'$ (line 3). When exclusively one of the input ontologies evolve, we only need one of the two compositions. We perform the first two steps if $O1$ evolves to $O1'$, or only perform compose($M_{O1,O2}$,$M_{O2,O2'}$) if $O2$ evolves to $O2'$. For the running example (Fig. 2), we would create eight correspondences including retained correspondences such as ('limbs','limbs'). Unfortunately, the composition also creates the false correspondences (('lower extremities','upper limbs'), ('upper extremities','lower limbs')) since the concept 'limb segment' in the intermediate ontology is connected to several concepts in the ontologies to be composed. We will later see how our alternate solution (Sec. 5) can cope with such situations.

Composition alone is also unable to determine new correspondences due to added concepts in the ontologies, *e.g.*, 'trunk' in $O2'$. To address this shortcoming we apply an additional match step as shown in the CompAdaptMatch algorithm:

Algorithm 2: CompAdaptMatch($M_{O1,O2}$,$M_{O1,O1'}$,$M_{O2,O2'}$,$O1$,$O1'$,$O2$,$O2'$)

```
1 M_{O1',O2'} ← CompAdapt(M_{O1,O2},M_{O1,O1'},M_{O2,O2'});
2 Add_{O1} ← O1'\O1;
3 Add_{O2} ← O2'\O2;
4 M_{O1',O2'} ← M_{O1',O2'} ⋃ match(Add_{O1},O2') ⋃ match(O1',Add_{O2});
5 return M_{O1',O2'};
```

After adapting the mapping using composition (line 1) we identify the added concepts (Add_{O1},Add_{O2}) in both ontologies (lines 2–3). We match the added

concepts with the other ontology to find new correspondences (line 4) and include them in the adapted mapping. We can simplify the algorithm when exclusively one of the ontologies has changed by merely matching added concepts of the changed ontology with the unchanged ontology. In the running example, we would determine 'trunk' as an added concept in $O2'$ and matching would result in the additional correct correspondence ('trunk','trunk') in the adapted mapping.

5 Diff-based Adaptation

The Diff-based adaptation of ontology mappings considers the individual ontology changes, and so-called change handlers to adapt the ontology mapping. This modular approach is highly flexible and can accommodate different types of changes as well as distinct automatic or interactive approaches for mapping adaptation. For example, a concept deletion would lead to the deletion of all affected correspondences with the composition-based approach, while a change handler could try to keep a correspondence with a neighbor of the deleted concept. Furthermore, change handlers might request expert verification for proposed mapping changes.

We first explain Diff-based mapping adaptation for the frequent case when only one of two ontologies changes (Sec. 5.1). We then explain the different change handlers and their approaches for mapping adaptation (Sec. 5.2). Finally, we discuss Diff-based adaptation for the general case with two evolving ontologies (Sec. 5.3). Although the proposed approach is applicable for different diff techniques to determine ontology changes, we assume the use of our algorithm COnto-Diff [27] for concreteness. COnto-Diff is suited to identify a diff evolution mapping for two successive versions of an ontology containing typical change operations such as *merge*, *substitute*, *split*, *addC* or *delC* (see Fig. 1b).

5.1 Adaptation Algorithm for One Evolving Ontology

The input data of the algorithm DiffAdapt (Algorithm 3) are the ontology mapping to be adapted ($M_{O1,O2}$), the two versions of the domain ontology $O1$, $O1'$, a diff between them ($diff_{O1,O1'}$) as well as the current version of the range ontology $O2$. We assume that the change handlers are listed in the order in which they should be applied for mapping adaptation(CH). This ordering is feasible since COnto-Diff ensures that a concept is the subject of at most one of the considered change operations.

Algorithm 3: DiffAdapt($M_{O1,O2}$,$diff_{O1,O1'}$,$O1$,$O1'$,$O2$,CH)

```
1 M_infl ← getInfluencedCorrs(M_O1,O2, diff_O1,O1', CH);
2 M_O1',O2 ← M_O1,O2 \ M_infl; //reuse unaffected mapping part
3 foreach ch ∈ CH do
4 |   diffPart ← diff.filter (ch.getHandledOperations());
5 |   ch.handleChg(M_infl, diffPart_O1,O1', O1, O1', O2);
6 M_O1',O2 ← M_O1',O2 ∪ M_infl;
7 return M_O1',O2;
```

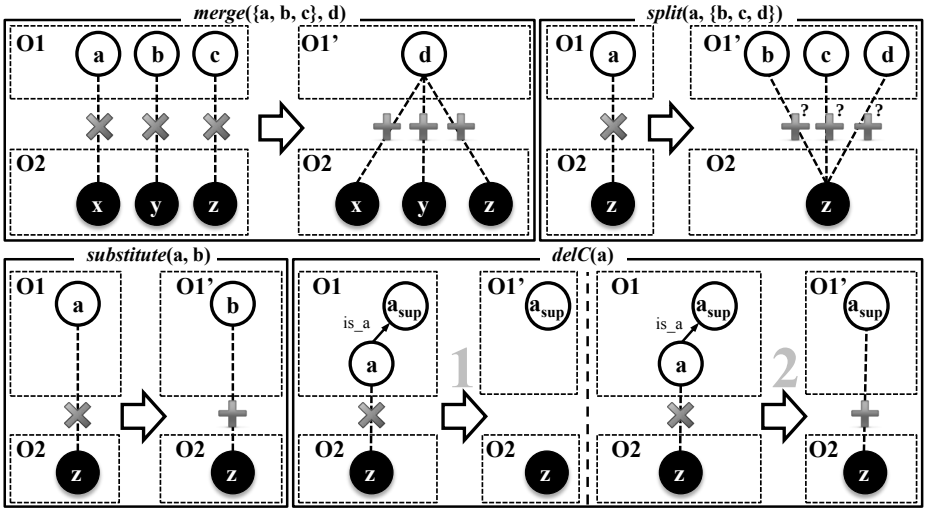


Fig. 3. Change handlers

We first identify all correspondences that are influenced by changes from the input diff. Therefore, we check if the domain concept of each correspondence was subject to a change operation listed in CH. All influenced correspondences in M_{infl} are initially set to status *to verify*, since they might require user verification. By contrast, we reuse unaffected correspondences (status *handled*) by adding them directly to the new mapping $M_{O1',O2}$ (line 2). For instance, in the running example (Fig. 2), 'limbs' and 'body' remain unchanged in $O2$ so that we keep the correspondences ('limbs','limbs') and ('body','body'). The influenced mapping part M_{infl} is then handled by the specified list of individual *Change Handlers* (lines 3-5). The mapping M_{infl} is iteratively adapted, *i.e.*, each change handler removes outdated correspondences from and adds new correspondences to M_{infl}. Depending on the used method in the change handler, the status of new correspondences is either set to *handled* or *to verify*. Finally, we take the union of the reused correspondences in $M_{O1',O2}$ and the adapted mapping part M_{infl} and then return the resulting mapping (lines 6-7).

5.2 Change Handlers

We provide a handler for each type of ontology change to implement appropriate approaches for mapping adaptation. These handlers can easily be adapted and extended to adjust mapping adaptation, request users' feedback in certain cases or deal with new types of ontology changes. Fig. 3 illustrates main adaptation choices for some major change operations namely *merge*, *substitute*, *split* and *delC*. It shows how correspondences from $M_{O1,O2}$ are adapted according to the evolution from $O1$ to $O1'$. In the following, we present the change handlers

in the order in which they are applied in the algorithm DiffAdapt: CH_{merge}, $CH_{substitute}$, CH_{split}, CH_{delC}, $CH_{toObsolete}$, CH_{addC} and $CH_{revokeObsolete}$.

In the *merge* operation, two or more source concepts from $O1$ are merged into one target concept in $O1'$. The merge handler migrates all correspondences once associated with any of the $O1$ concepts to the target concept in $O1'$. Thus, each correspondence from $M_{O1,O2}$ associated with concepts to be merged are removed and new correspondences to the target concept are added. In the running example (Fig. 2) 'head' and 'neck' concepts are merged as 'head and neck'. All correspondences once related to 'head' or 'neck' are assigned to the new concept 'head and neck'. Algorithm 4 details the sketched approach of the merge handler. It checks for each correspondence *corr* (line 1) and merge operation *merge* (line 2) if the domain concept of *corr* is equal to one of the source concepts in *merge* (lines 5-6). If so, the affected correspondence is adapted.

Algorithm 4: MergeHandler(M,$Merge$,$O1$,$O1'$,$O2$)

```
1  foreach corr ∈ M do
2      foreach merge ∈ Merge do
3          S ← merge.getSourceIDs();
4          t ← merge.getTargetID();
5          foreach s ∈ S do
6              if s = corr.getDomainID() then
7                  newType ← getNewType(corr.getType(), ≤);
8                  newStatus ← getNewStatus(corr.getType(), ≤);
9                  newCorr ← createCorr(t, corr.getRangeID(),
10                 corr.getSim(), newType, newStatus);
11                 M.remove(corr).add(newCorr);
```

The merge handler supports an adaptation of the semantic type of added correspondences. For example, for $merge(\{a, b, c\}, d)$ it usually holds that concepts a, b, c are less general ($\leq$) than d. Hence, we combine $\leq$ with the semantic type of the old correspondence ($=, \leq, \geq, \approx$) to derive the new semantic correspondence type.

Such an adaptation of the semantic correspondence type is needed for different types of changes and was also applied for mapping composition. To combine semantic types of correspondences (operation getNewType) and to determine the new correspondence status (operation getNewStatus) we currently use a set of combination rules as shown in Fig. 4a. The basic idea is that the semantic type with lower binding strength imposes the new semantic type. Following the definition of *semantic relation* in [28], $=$ has a higher binding strength than $\leq$ and $\geq$ which in turn are stronger than $\approx$. $\leq$ and $\geq$ are of equal binding strength such that the new semantic type of their combination can not be determined by rules (gray fields). The status *to verify* is set to $\approx$ since a user necessarily needs to check this correspondence and its semantic type. For all other combinations as shown in Fig. 4a, the status of the correspondence is *handled*.

For the $substitute(a, b)$ change operation, the applied strategy is similar to the one used for *merge*. In this case, the concept $a \in O1$ is substituted by the target concept $b \in O1'$. Since a is involved in a correspondence with z in $O2$, the correspondence between a and z is removed and the new correspondence from b

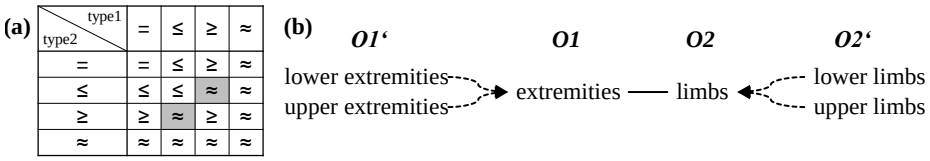


Fig. 4. (a) Combining semantic types (getNewType) and determine the new correspondence status (getNewStatus). (b) Example of conflicting changes for two evolving ontologies.

to z is added. We can assume $a = b$ as semantic type for substitute, and combine this with the old semantic type of the correspondence to derive the new one.

The adaptation of correspondences affected by split change operations is more complex. For example, $split(a, \{b, c, d\})$ caused a single source concept $a \in O1$ to be split into several target concepts b,c,$d \in O1'$. In the mapping adaptation, we first remove all correspondences associated with the split source concept a. We consider two strategies for adding new correspondences. First, one can add all possible combinations of correspondences between the split target concepts b, c, d and the unmodified range concept z in $O2$ ("take all"). Second, we can restrict the output result to the best correspondence(s), *i.e.*, the one(s) with the highest similarity based on a local match between b, c, d and z ("take best").

Also for split, new adapted correspondences obtain an individual new semantic type based on the rules in Fig. 4a and assuming that $d \geq a$, b, c holds for split. All correspondences get status *to verify* since these are only recommendations and an expert needs to decide about their validity. In the running example (Fig. 2) 'limb segment' was split into 'lower limbs' and 'upper limbs'. Using the "take all" strategy, we would present all four possible combinations between 'lower extremities', 'upper extremities' and 'lower limbs', 'upper limbs' to the user. Using the "take best" strategy, we can correctly identify the most adequate correspondences 'lower limbs' with 'lower extremities', and 'upper limbs' with 'upper extremities'.

For deletion of concepts ($delC(a)$) we also consider two strategies. First, all correspondences referencing deleted concepts in $O1'$ are removed (see Fig. 3) (strategy "del corr"). This is the case for 'tail' in the running example. Second, correspondences can be transferred to their parent concept, if possible ("keep corr"). Thus, correspondences related to the deleted concept a are removed, but new "more general" correspondences are created. In particular, the domain of the new correspondence is the first super concept (a_{sup}) of a. In case of multiple inheritance, the correspondence can be transferred to all parents. The status is set to *to verify* since a user has to check the adapted correspondences. The new semantic type is derived by following the $\leq$ parent relationship in $O1$ combined with the semantic type of the old deleted correspondence. For *toObsolete* changes we apply the same handler.

For all concept additions and *revokeObsolete* operations in $O1'$ we apply an automatic matching step with the whole range ontology $O2$. The status of the

new recommended correspondences is set to *to verify*. One can either apply a very restrictive selection of correspondences to show only the best matches to experts, and avoid many false positives, or to be less restrictive in order to get a perfect recall and let the selection up to the user. In the running example, $diff_{O2,O2'}$ contains an addition of the concept 'trunk' which is matched to $O1$ such that ('trunk','trunk') is correctly identified by selecting only the top result.

5.3 Adaptation Algorithm for Two Evolving Ontologies

In case where both ontologies change (domain and range of the correspondences), we can adapt the mapping by applying the DiffAdapt (Algorithm 3) twice as follows:

Algorithm 5: DiffAdaptBoth($M_{O1,O2}$,$diff_{O1,O1'}$,$diff_{O2,O2'}$,$O1$,$O1'$,$O2$,$O2'$,CH)

```
1 M_{O1',O2} ← DiffAdapt(M_{O1,O2}, diff_{O1,O1'}, O1, O1', O2, CH);
2 M_{O2,O1'} ← inverse(M_{O1',O2});
3 M_{O2',O1'} ← DiffAdapt(M_{O2,O1'}, diff_{O2,O2'}, O2, O2', O1', CH);
4 return inverse(M_{O2',O1'});
```

The input of algorithm DiffAdaptBoth (Algorithm 5) is similar as for DiffAdapt but requires two versions for both input ontologies $O1$, $O1'$, $O2$, $O2'$, as well as two diff mappings $diff_{O1,O1'}/diff_{O2,O2'}$. First, we adapt the given ontology mapping with respect to changes in the domain ontology to get $M_{O1',O2}$. To adapt the mapping regarding changes in the range ontology we call DiffAdapt with the inverse mapping $M_{O2,O1'}$ and the range diff $diff_{O2,O2'}$ (line 3). Finally, we invert the mapping again and return it (line 4).

When both ontologies change, some correspondences might be affected by changes of the domain and range concept at the same time. For instance, if both concepts of a correspondence are split into several concepts, we can produce wrong results by independently handling these changes one after the other. A possible problem scenario is shown in Fig. 4b. Applying the "take all" strategy twice would create too many correspondences, namely the local cross-product. By contrast, "take best" might lead to a wrong selection of ('lower extremities','limbs') in the first step, such that we can only find ('lower extremities','lower limbs') after the adaptation concerning the range ontology. To deal with such situations when both ontologies have evolved, we propose to handle these conflicting changes together in an extra step. We can first identify correspondences involved in conflicts and modify the input mapping before we run DiffAdaptBoth. In particular, we recommend to check conflicting change combinations as *split-split*, *merge-split* and *substitute-split* where it is helpful to do the migration on both sides in one step.

6 Evaluation

To evaluate the proposed approaches for mapping adaptation, we use three large life science ontologies: SNOMED-CT (SCT), NCI Thesaurus (NCI) and FMA. We use the integrated ontology UMLS to extract two mappings NCI-FMA and

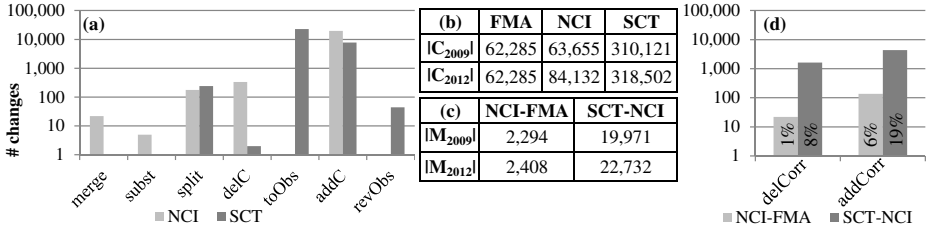


Fig. 5. (a) Ontology changes (b) Ontology size (c) Mapping size (d) Mapping changes.

SCT-NCI in two versions for 2009 and 2012 (see [29] for extraction details). We adapt the mapping versions from 2009 with the proposed algorithms, and use the 2012 versions as reference mappings for evaluating the quality of the mappings adapted. It is important to notice that such reference mappings can be considered as a 'silver standard', *i.e.*, these mappings are not complete, and curators manually correct them by modifying also correspondences associated with concepts that did not underlie changes. In this evaluation we eliminate such correspondences from the mappings since they do not change due to ontology modifications and can thus not be detected. To assess the quality of the adapted mappings with respect to the 2012 reference mappings, we calculate the standard metrics of Precision, Recall and F-Measure.

In the following we first analyze the used data sets (Sec. 6.1) and then evaluate the quality of the proposed mapping adaptation approaches (Sec. 6.2).

6.1 Ontology and Mapping Analysis

Fig. 5 gives an overview of changes in the considered ontology versions (a) and mapping versions (d) as well as of their sizes (b,c). From 2009 to 2012, FMA remains completely stable while NCI and SCT have been revised considerably. Besides some merge operations (22 for NCI) there was a notable number of ∼180 (240) concept splits for NCI (SCT). In SCT an enormous amount of >22.000 concepts has been set to obsolete while NCI has been extended by ∼20.000 concepts during 2009 and 2012. The 2009 mapping version of NCI-FMA is relatively small (∼2300) compared to SCT-NCI (∼20400) (Fig. 5c). During the considered time interval of three years, the NCI-FMA mapping grew by ∼5% and SCT-NCI by even 14%. The SCT-NCI mapping has been affected by more changes, namely 8% of the correspondences have been deleted from the old and 19% were added to the new mapping version. Thus, NCI-FMA has a higher rate of unchanged correspondences and might be easier to adapt than SCT-NCI.

6.2 Mapping Adaptation Results

Fig. 6 shows the quality of the mapping adaptation results for NCI-FMA (left) and SCT-NCI (right). To have a basic reference for analyzing how much each

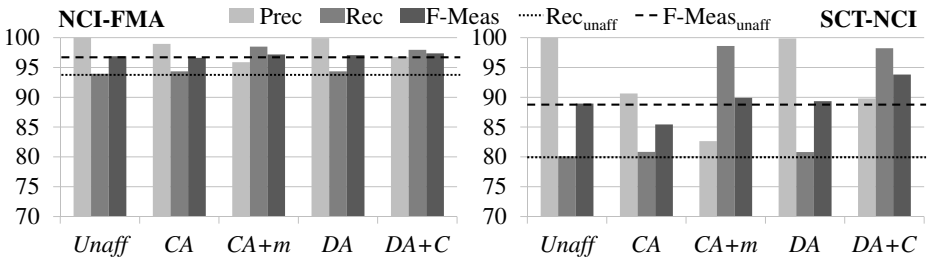


Fig. 6. Results on the Quality of Mapping Adaptation.

adaptation approach contributes, we mark the impact of unaffected (stable) correspondences in the adapted mapping (*Unaff*). The dotted and dashed lines highlight the recall (Rec_{unaff}) and F-Measure ($F\text{-}Meas_{unaff}$) of *Unaff*. We compare results with the composition-based adaptation (CA) and its match extension (CA+m). Moreover, we apply the diff-based adaptation (DA) using the major handlers CH_{merge}, $CH_{substitute}$, CH_{split} ("take best"), CH_{delC} and $CH_{toObsolete}$ ("del corr"), and as an extension (DA+C) the CH_{addC} and $CH_{revokeObsolete}$ handlers. Note that our approach is flexible and can be easily extended to handle also attribute and structural changes. In the evaluation scenario, this showed to have a negative impact on the quality of adapted mappings, such that we omit it in this study. We consider this an issue for future investigations.

For both cases analyzed, the basic quality of *Unaff* is already very high, since 94% (80%) of the NCI-FMA (SCT-NCI) mappings were unaffected and could be reused. For the adaptation of the relatively stable NCI-FMA mapping all considered approaches perform similarly well and achieve a very high F-Measure. SCT-NCI is a more challenging mapping adaptation scenario and helps to better differentiate the relative effectiveness of the proposed approaches. Compared to *Unaff*, CA is less precise and increases the recall only marginally. This is caused by the fact that the applied compose approach takes all possible combinations of existing correspondences, and no further selection takes place. An additional match of new concepts (CA+m) significantly increases the recall by 18.6% for SCT-NCI and slightly improve F-Measure compared to *Unaff* (despite a reduced precision for automatically generated match correspondences).

For SCT-NCI, the diff-based approaches clearly outperform the composition-based approaches. They not only reuse unaffected correspondences but can further improve recall with relatively high precision due to the individual change handling. DA+C performs best overall since it utilizes additional change handlers. In particular, it can find additional match correspondences for added concepts leading to a significant increase in recall and F-Measure. While this is similar to the high recall of CA+m, the precision and thus F-Measure remains higher for DA+C ($\sim$94% instead of $\sim$90%). The recall could even be further increased by using a lower match threshold than the applied 1.0, and let experts select the correct correspondences out of the recommended matches in DA+C.

Based on these results, we recommend that ontology mappings might be adapted in a semi-automatic manner as follows: (1) first, determine a consistent adapted mapping using the DA approach; (2) apply further strategies such as DA+C that provide recommendations of new correspondences; (3) apply expert knowledge based on the adaptation results to complete the mapping and validate those correspondences with *to verify* status.

7 Conclusion

Ontology evolution can potentially invalidate previously created mappings. We proposed a *composition- and a diff-based* approach for adapting ontology mappings as a consequence of ontology evolution. Both approaches can reuse unaffected correspondences from existing mappings and adapt only the changed parts in a (semi-)automatic way. The composition-based approach is conceptually simpler but can be already sufficient for ontologies that change only slightly. The diff-based approach is more powerful by supporting different change-specific approaches for mapping adaptation and by enabling experts to verify proposed correspondences. The conducted evaluation for large life science ontologies confirmed the high effectiveness of the proposed approaches. Both of them benefit from matching new concepts to produce a more complete mapping.

For future work, we plan to realize a more refined adaptation of semantic mappings. The techniques presented already support the migration of semantic mappings, but this has to be investigated in more detail and evaluated for real-world semantic mappings. Additionally, in further evaluation expert users should analyze the quality of mappings for the different adaptation strategies.

Acknowledgment

This work is funded by the German Research Foundation (DFG) (grant RA 497/18-1, "Evolution of Ontologies and Mappings"), by the National Research Fund (FNR) of Luxembourg (grant C10/IS/786147), by the European Social Fund and the Free State of Saxony.

References

1. Bodenreider, O., Stevens, R.: Bio-ontologies: current trends and future directions. Briefings in bioinformatics **7**(3) (2006)
2. Lambrix, P., Tan, H., Jakoniene, V., Strömbäck, L.: Biological Ontologies. In: Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences. (2007)
3. Smith, B., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature biotechnology **25**(11) (2007)
4. Mungall, C., et al.: Uberon, an integrative multi-species anatomy ontology. Genome Biol **13**(1) (2012)
5. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10**(4) (2001)

6. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag New York (2007)
7. Rahm, E.: Towards Large Scale Schema and Ontology Matching. In: Schema Matching and Mapping. Springer (2011)
8. Hartung, M., Kirsten, T., Rahm, E.: Analyzing the evolution of life science ontologies and mappings. In: Proc. DILS. (2008)
9. Malone, J., Stevens, R.: Measuring the level of activity in community built bio-ontologies. J Biomed Inform. **46**(1) (2013)
10. Dos Reis, J., Pruski, C., Da Silveira, M., Reynaud, C.: Analyzing and Supporting the Mapping Maintenance Problem in Biomedical Knowledge Organization Systems. In: Proc. SIMI Workshop at ESWC. (2012)
11. Groß, A., Hartung, M., Thor, A., Rahm, E.: How do computed ontology mappings evolve?-A case study for life science ontologies. Joint Workshop on Knowledge Evolution and Ontology Dynamics (2012)
12. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: Estimating the quality of ontology-based annotations by considering evolutionary changes. In: Proc. DILS. (2009)
13. Groß, A., Hartung, M., Prüfer, K., Kelso, J., Rahm, E.: Impact of Ontology Evolution on Functional Analyses. Bioinformatics **28**(20) (2012)
14. Kondylakis, H., Plexousakis, D.: Ontology Evolution: Assisting Query Migration. In: Proc. ER. (2012)
15. Liang, Y., Alani, H., Shadbolt, N.: Changing ontology breaks queries. In: Proc. ISWC. (2006)
16. Noy, N., et al.: BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic acids res. **37**(suppl 2) (2009)
17. Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research **32**(suppl 1) (2004)
18. Hartung, M., Terwilliger, J.F., Rahm, E.: Recent Advances in Schema and Ontology Evolution. In: Schema Matching and Mapping. Springer (2011)
19. Velegrakis, Y., Miller, J., Popa, L.: Mapping Adaptation under Evolving Schemas. In: Proc. VLDB. (2003)
20. Bernstein, P., Melnik, S.: Model management 2.0: manipulating richer mappings. In: Proc. SIGMOD. (2007)
21. Yu, C., Popa, L.: Semantic Adaptation of Schema Mappings when Schemas Evolve. In: Proc. VLDB. (2005)
22. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning Support for Mapping Revision. Journal of Logic and Computation **19**(5) (2008)
23. Khattak, A., Pervez, Z., Latif, K., Lee, S.: Time efficient reconciliation of mappings in dynamic web ontologies. Knowl.-Based Syst. **35** (2012)
24. Martins, H., Silva, N.: A User-Driven and a Semantic-Based Ontology Mapping Evolution Approach. In: Proc. Intl. Conf. on Enterprise Inform.Systems. (2009)
25. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA results for OAEI 2012. In: Proc. OM Workshop at ISWC. Volume 11. (2012)
26. Noy, N.F., Musen, M.A.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: Proc. of Nat. Conf. on Artificial Intelligence. (2002)
27. Hartung, M., Groß, A., Rahm, E.: COnto-Diff: generation of complex evolution mappings for life science ontologies. J Biomed Inform. **46**(1) (2013)
28. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an algorithm and an implementation of semantic matching. The semantic web: research and applications (2004)
29. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. J Biomed Sem. **2** (2011)