Repairing ABoxes through Active Integrity Constraints

Christos Rantsoudis, Guillaume Feuillade, Andreas Herzig

Repairing ABoxes through active integrity constraints

Christos Rantsoudis, Guillaume Feuillade, and Andreas Herzig

IRIT, CNRS, Univ. Toulouse

Abstract. In the literature of database repairing, active integrity constraints have provided a means of restoring integrity through a set of *preferred* update actions. While this is a known issue in the database community, it has not yet been directly applied to description logics. In this paper, we extend description logic TBoxes by a similar set of *preferred* actions and tackle the problem of ABox repairing by taking into account the new "active" TBox. For this, a mainly syntactic approach is explored and a further, dynamic logic oriented, semantic approach is suggested and briefly previewed.
Keywords: active integrity constraints, description logic, ABox repair

1 Introduction

In the database literature integrity constraints are usually considered to be universal conditions or rules that must hold in any situation. When a database fails to satisfy such constraints it has to be repaired in order to *restore integrity*. On a similar note, TBoxes in description logic are usually created by a long and careful procedure, rendering them of the highest priority for the ABoxes to abide by. In case of inconsistencies between an ABox and a TBox, the ABox is usually the one that should be updated to conform with the *rules* of the TBox [4, 16].

As a next step, active integrity constraints were introduced more recently in an effort to provide *preferred* ways for preserving integrity [11, 6, 7, 5, 8, 9]. This was done by extending the static integrity constraints by additional update actions, indicating to each constraint some preferred ways to repair out of all the possible ones. In this paper, we integrate the same idea to the TBoxes of description logic and investigate ways in which ABoxes could be repaired according to new "active" TBoxes. We do this by first defining an extension of the usual TBox, taking into account the operations $add(A)$ and $remove(A)$, where A is an atomic concept. The intuition is that whenever an inclusion of the form $A \sqcap B \sqsubseteq \bot$ appears inside the TBox, for atomic concepts A and B, then the TBox should also indicate the preferred way that this should be repaired when the constraint is violated in the ABox. While the problem is easier when an assertion of the form $a : A \sqcap B$ has to be updated to the assertion $a : A \sqcap \neg B$, difficulties start to arise when the conjunction of two concepts appears inside the scope of a quantifier. An example of the latter is when having to update the assertion $a : \forall R.(A \sqcap B)$ to the assertion $a : \forall R.(A \sqcap \neg B)$. The direction we pursue is clearly different from previous attempts to encode integrity constraints into TBoxes, which differentiate the Knowledge Base (the TBox and the ABox as a single set) from the set of integrity constraints, treating the latter with a closed-worlds assumption and using different dedicated semantics [19, 18].

$$
\begin{aligned}
\mathsf{Married} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hasSpouse.Person} \\
\mathsf{Divorced} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hadSpouse.Person} \sqcap \neg \exists \mathsf{hasSpouse.Person} \\
\mathsf{Bachelor} &\equiv \mathsf{Person} \sqcap \neg \exists \mathsf{hadSpouse.Person} \sqcap \neg \exists \mathsf{hasSpouse.Person} \\
\mathsf{Widowed} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hadSpouse.Deceased} \sqcap \neg \exists \mathsf{hasSpouse.Person} \\
\mathsf{Bachelor} \sqcap \mathsf{Married} &\sqsubseteq \bot \\
\mathsf{Divorced} \sqcap \mathsf{Widowed} &\sqsubseteq \bot \\
\mathsf{Person} &\sqsubseteq \mathsf{Married} \sqcup \mathsf{Divorced} \sqcup \mathsf{Bachelor} \sqcup \mathsf{Widowed}
\end{aligned}
$$

Fig. 1. Example of a TBox

We give a simple example of a TBox in Figure 1, expressing the different definitions of marital status. In this TBox it is clearly stated that someone cannot be identified as *bachelor* and *married* at the same time. Now an ABox containing the concept $\mathsf{Bachelor} \sqcap \mathsf{Married}$ in its assertions would clearly be inconsistent with respect to this TBox and would have to be repaired. Dropping one of the two would solve this, however one could argue that a person declared as both bachelor and married should only be identified as *married* everywhere. Whereas one can achieve married status from being a bachelor, a married person cannot 'go back' to being bachelor but can only become divorced or widowed instead. Thus the dropping of the concept $\mathsf{Bachelor}$ whenever the concept $\mathsf{Bachelor} \sqcap \mathsf{Married}$ appears in an ABox should be the preferred update action. In the same vein, since by the last constraint someone has to have a marital status, the preferred update action would be to add the concept $\mathsf{Bachelor}$ to an individual invalidating this constraint, as in any other case the person would have to declare that he got married, divorced or widowed. Finally, as a distinction between divorced and widowed cannot be made without further information, the constraint $\mathsf{Divorced} \sqcap \mathsf{Widowed} \sqsubseteq \bot$ should not give any preference between the concepts $\mathsf{Divorced}$ and $\mathsf{Widowed}$. Therefore, as we witness by this example, there are sufficient logical reasons behind the investigation on extensions of TBoxes, equipped with extra information on preferences and ways to make use of them.

The paper will be presented as follows. In Section 2 we assume familiarity with the basics of Description Logic and therefore only present a background on active integrity constraints. Section 3 is where we present the ideas discussed above. We start by defining the "active" TBox as an extension of the usual TBox in Section 3.1. We then continue by first taking a syntactic approach in Section 3.2, exploring ways in which we can reach a desired ABox that is repaired according to the preferences of the "active" TBox. A somewhat brief semantic approach then follows in Section 3.3, using the programs of Dynamic Logic, based on the ideas that are previously explored. Finally, we conclude in Section 4.

2 Background

We will use the basic description logic $\mathcal{ALC}$ and its extension $\mathcal{ALCO}$ with nominals [1]. Furthermore, throughout the paper we impose the following conventions:

- we suppose that all ABoxes are consistent and that all TBoxes are consistent
- a TBox will contain only concept definitions and concept inclusions

– all TBoxes we talk about will be considered to be acyclic
– given a TBox, a constraint is any inclusion of concepts appearing inside the TBox
– we only work on a 'simple' kind of constraints, which we will call *static* constraints and which will later be extended into *active* constraints

2.1 Active integrity constraints

In this subsection we present the basic notions around active integrity constraints mainly as presented in [8]. In the database literature, databases are sets of propositional variables denoted by V. A static integrity constraint r is a formula $L_1 \wedge \cdots \wedge L_n \rightarrow \bot$ where $L_1, \ldots, L_n$ are literals (i.e., propositional variables or negations of propositional variables) denoting that when a database satisfies all literals in the conjunction then it has to be repaired in order to satisfy r. A set of static integrity constraints is usually denoted by C. An update action is an expression of the form $+p$ or $-p$, where p is a propositional variable, denoting the insertion or deletion of p in a database V. A set of update actions U is consistent if it is not the case that both $+p \in U$ and $-p \in U$ for some propositional variable p. A consistent set of update actions U is called a *weak repair* of V achieving C if the update of V by U satisfies all the static constraints in C, formally: $V \diamond U \models \bigwedge C$. Usually there are several ways to repair a database V in order to satisfy a set of static integrity constraints C. A consistent set of update actions U is called a *repair* or *PMA repair* of V achieving C if U is a weak repair of V achieving C that is minimal with respect to set inclusion, i.e., there is no weak repair U' of V achieving C such that $U' \subset U$. The term *PMA repair* indicates the close relation of *repairs* and Winslett's *possible models approach* to updating a database [20, 21, 15].

As a next step, active integrity constraints are an extension of static constraints by additional update actions $+p$ and $-p$ for propositional variables p. They usually have the form $L_1 \wedge \cdots \wedge L_n \rightarrow a_1 \vee \cdots \vee a_m$, where $a_1, \ldots, a_m$ are update actions that indicate which from the literals $L_1, \ldots, L_n$ should change in case $L_1 \wedge \cdots \wedge L_n$ is satisfied. Of course in this form active integrity constraints are not formulas, as the right part of the expression is a disjunction of update actions rather than literals or propositional variables. If r is an active integrity constraint of this form, we denote by $body(r)$ the formula $L_1 \wedge \cdots \wedge L_n \rightarrow \bot$ and by $head(r)$ the set of update actions $\{a_1, \ldots, a_m\}$. A set of active integrity constraints is usually denoted by η.

Now let V be a database, η a set of active integrity constraints and U a consistent set of update actions. An update action $a \in U$ is *founded* if there is an active constraint $r \in \eta$ such that $a \in head(r)$, $V \diamond U \models body(r)$ and $V \diamond (U \setminus \{a\}) \not\models body(r)$. A consistent set of update actions U is founded if every update action $a \in U$ is founded. Intuitively, when a set of update actions U is founded then each $a \in U$ provides a unique repair action for the database V to satisfy a constraint r. Finally, let $body(\eta) = \{body(r) \mid r \in \eta\}$. A set of update actions U is a *founded weak repair* of V by η if U is a weak repair of V achieving $body(\eta)$ and U is founded. A set of update actions U is a *founded repair* of V by η if U is a PMA repair of V achieving $body(\eta)$ and U is founded. Although founded weak repairs and founded repairs do not necessarily exist and other forms of repairs have to be sought, they provide the basic means for repairing a database taking into account a set of active integrity constraints.

3 A new kind of ABox repairing

3.1 Integrating active constraints to the TBox

Similarly to how the active integrity constraints of Section 2.1 extend static constraints by adding additional update actions to the *head* of each constraint, we define "active" TBoxes to contain preferred update actions of the form $add(A)$ and $remove(A)$ for atomic concepts A. We start by defining what exactly is a static constraint in this setting.

Definition 1. *Let $\mathcal{T}$ be a TBox. A conjunctive constraint is any inclusion of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq \bot$ appearing inside the TBox, where $C_1, \ldots, C_n$ are either atomic or negation of atomic concepts. A static constraint is any inclusion appearing inside the TBox that is either a conjunctive constraint or equivalent to a conjunctive constraint.*

For example, in the TBox of Figure 1, all three of the inclusions

$$\mathsf{Bachelor} \sqcap \mathsf{Married} \sqsubseteq \bot, \quad \mathsf{Divorced} \sqcap \mathsf{Widowed} \sqsubseteq \bot$$

$$\mathsf{Person} \sqsubseteq \mathsf{Married} \sqcup \mathsf{Divorced} \sqcup \mathsf{Bachelor} \sqcup \mathsf{Widowed}$$

are static constraints since the first two are conjunctive constraints whereas the third is equivalent to the conjunctive constraint

$$\mathsf{Person} \sqcap \neg\mathsf{Married} \sqcap \neg\mathsf{Divorced} \sqcap \neg\mathsf{Bachelor} \sqcap \neg\mathsf{Widowed} \sqsubseteq \bot.$$

We continue with the definition of an active constraint.

Definition 2. *Let r be a static constraint. If r is not a conjunctive constraint, let r^* be the conjunctive constraint that is equivalent to r. An active constraint r' is the extension of r by either $add(A)$ or $remove(A)$ for some of the atomic concepts A appearing in r, according to the following rules:*

- *$add(A)$ can exist in r' whenever $\neg A$ is a literal on the conjunction of r (or r^*).*
- *$remove(A)$ can exist in r' whenever A is a literal on the conjunction of r (or r^*).*

We use the symbol $\rightarrow$ (being part of the metalanguage) to differentiate between the $body(r')$, which is r itself, and the $head(r')$, which is the set of update actions $add(A)$ and $remove(A)$ for atomic concepts A. For instance, for a static constraint r of the form $\neg A \sqcap B \sqsubseteq \bot$, the following are the possible active constraints extending it:

$r_1 : \neg A \sqcap B \sqsubseteq \bot \rightarrow add(A)$
$r_2 : \neg A \sqcap B \sqsubseteq \bot \rightarrow remove(B)$
$r_3 : \neg A \sqcap B \sqsubseteq \bot \rightarrow add(A), remove(B)$

The first two give a preference to one of the two concepts, while the third gives no preference to any of them. We formalize all the above by the relation $r \rightsquigarrow r'$, where r is a static constraint and r' an active constraint extending it as described in Definition 2. The next step is to generalize this construction to TBoxes. We extend the relation and define "active" TBoxes as follows.

$$
\begin{aligned}
\mathsf{Married} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hasSpouse.Person} \\
\mathsf{Divorced} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hadSpouse.Person} \sqcap \neg \exists \mathsf{hasSpouse.Person} \\
\mathsf{Bachelor} &\equiv \mathsf{Person} \sqcap \neg \exists \mathsf{hadSpouse.Person} \sqcap \neg \exists \mathsf{hasSpouse.Person} \\
\mathsf{Widowed} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hadSpouse.Deceased} \sqcap \neg \exists \mathsf{hasSpouse.Person} \\
\mathsf{Bachelor} \sqcap \mathsf{Married} &\sqsubseteq \bot \rightarrow \mathit{remove}(\mathsf{Bachelor}) \\
\mathsf{Divorced} \sqcap \mathsf{Widowed} &\sqsubseteq \bot \rightarrow \mathit{remove}(\mathsf{Divorced}), \mathit{remove}(\mathsf{Widowed}) \\
\mathsf{Person} &\sqsubseteq \mathsf{Married} \sqcup \mathsf{Divorced} \sqcup \mathsf{Bachelor} \sqcup \mathsf{Widowed} \rightarrow \mathit{add}(\mathsf{Bachelor})
\end{aligned}
$$

Fig. 2. Example of an active TBox, based on the TBox of Figure 1

Definition 3. *Let $\mathcal{T}$ be a TBox. $\mathcal{AT}$ is an active TBox extending $\mathcal{T}$, viz. $\mathcal{T}$ $\mathcal{AT}$, iff for each static constraint r in $\mathcal{T}$ there is an active constraint r' in $\mathcal{AT}$ such that r r'.*

In Figure 2 we see an example of an active TBox, based on the TBox of Figure 1 and the discussion about the preferred update actions in order to repair it. Finally, for any active TBox $\mathcal{AT}$ we denote with $\mathit{static}(\mathcal{AT})$ the TBox $\mathcal{T}$ for which $\mathcal{T} \rightsquigarrow \mathcal{AT}$ and say that an ABox $\mathcal{A}$ is consistent (respectively inconsistent) with respect to $\mathcal{AT}$ iff $\mathcal{A}$ is consistent (respectively inconsistent) with respect to $\mathit{static}(\mathcal{AT})$.

In what follows, we present a syntactic and briefly a semantic approach, based on Dynamic Logic, for the difficult task of repairing an ABox, inconsistent with respect to an active TBox, taking into account preferred update actions, especially when the inconsistencies appear inside the scope of a quantifier.

3.2 A syntactic approach

While updating to a simple ABox (i.e., an ABox whose assertions consist only of concept literals) is quite straightforward, the update to an ABox having complex concepts may not be easy (or even impossible) [12, 13, 17]. Consider for instance the active constraint $r : A \sqcap B \sqsubseteq \bot \rightarrow \mathit{remove}(B)$ and an ABox which includes only the assertions $a : \forall\, r.\,(A \sqcap B)$ and $r(a, b)$ for two individuals a and b. We would then like to repair this ABox with respect to r into the ABox having either $a : \forall\, r.\,A$ or $a : \forall\, r.\,(A \sqcap \neg B)$ as an assertion for the individual a[1]. From a semantic point of view, however, it is not clear what set of update actions would achieve this goal, especially when the update actions are defined only on the atomic level. In this subsection we investigate the direction of how we could transform an initial ABox, inconsistent with respect to the active TBox, to a repaired one conforming to the active constraints of the TBox. We mainly focus on the syntactic procedure that leads to a repaired ABox and what this resulting ABox would look like.

Consider that $\mathcal{AT}$ is the active TBox we are working with. We start by defining a relation between two ABoxes, so that the second ABox is the outcome of applying a

[1] Whereas the former seems like a better candidate for a repair (taking into account the open-world nature of DLs) we do not give a preference to any of them as long as the inconsistencies are eliminated. Regarding *minimality of change*, this will be defined syntactically to be the least amount of syntactic changes made in the ABox, once again providing no preference between the two.

small change to the first ABox. Define the set $S_{\mathcal{A}}$ to consist of all concept symbols in the ABox $\mathcal{A}$ and TBox $\mathcal{A}\mathcal{T}$. For $A \in S_{\mathcal{A}}$ define the following:

- $A_{\sqcup} = \{A \sqcup B \mid B \in S_{\mathcal{A}}\}$
- $A_{\sqcap} = \{A \sqcap B \mid B \in S_{\mathcal{A}}\}$
- $A_{\neg} = \{\neg A\}$

Furthermore, define:

- $\Gamma_{\mathcal{A}:A} = A_{\sqcup} \cup A_{\sqcap} \cup A_{\neg}$
- $\Gamma_{\mathcal{A}} = \bigcup_{A \in S_{\mathcal{A}}} \Gamma_{\mathcal{A}:A}$

Intuitively, $\Gamma_{\mathcal{A}}$ denotes the set of concepts that can be reached with one step from $S_{\mathcal{A}}$ using the three boolean constructors. Next, we write $\mathcal{A}[A \mid C]$ to denote the replacement in $\mathcal{A}$ of *some* instances of the atomic concept A with the concept C. Then we define the following relation.

Definition 4. *Let $\mathcal{A}$ and $\mathcal{A}'$ be ABoxes. Then $\mathcal{A} \sim_1 \mathcal{A}'$ iff:*

1. *there is an atomic concept $A \in S_{\mathcal{A}}$ and a concept $C \in \Gamma_{\mathcal{A}:A}$ such that $\mathcal{A}' = \mathcal{A}[A \mid C]$ or $\mathcal{A} = \mathcal{A}'[A \mid C]$.*
2. *$\mathcal{A}$ and $\mathcal{A}'$ are semantically different from one another, i.e., there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \not\models \mathcal{A}'$.*

The relation $\sim_1$ is clearly symmetric and irreflexive. The next step is to generalize this definition to an n-step relation between two ABoxes.

Definition 5. *Let $\mathcal{A}$ and $\mathcal{A}'$ be ABoxes and $n > 0$. Then $\mathcal{A} \sim_n \mathcal{A}'$ iff there are ABoxes $\mathcal{A}_1, ..., \mathcal{A}_{n+1}$ such that:*

1. *$\mathcal{A} = \mathcal{A}_1, \mathcal{A}_{n+1} = \mathcal{A}'$ and $\mathcal{A}_i \sim_1 \mathcal{A}_{i+1}, \forall i \in \{1, \ldots, n\}$.*
2. *$\mathcal{A}_1, ..., \mathcal{A}_{n+1}$ are semantically different from one another, i.e., for any two $\mathcal{A}_i$ and $\mathcal{A}_j$ there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{A}_i$ and $\mathcal{I} \not\models \mathcal{A}_j$.*
3. *there is no $n' < n$ with these two properties.*

When $\mathcal{A} \sim_n \mathcal{A}'$ we say that *at least n steps are needed* in order to reach the ABox $\mathcal{A}'$ from the ABox $\mathcal{A}$. Note that while we may have $\mathcal{A}_2 = \mathcal{A}_1[A_1 \mid C_1]$ and $\mathcal{A}_3 = \mathcal{A}_2[A_2 \mid C_2]$ and therefore $\mathcal{A}_1 \sim_1 \mathcal{A}_2 \sim_1 \mathcal{A}_3$, we do not have $\mathcal{A}_1 \sim_3 \mathcal{A}_3$ because of the last constraint. Finally, let us define the relation $\mathcal{A} \sim \mathcal{A}'$ to mean that $\mathcal{A}'$ can be reached from $\mathcal{A}$ by *an arbitrary number of steps*.

Definition 6. *Let $\mathcal{A}$ and $\mathcal{A}'$ be ABoxes. Then $\mathcal{A} \sim \mathcal{A}'$ iff there exists $n > 0$ such that $\mathcal{A} \sim_n \mathcal{A}'$.*

Note that by construction, $\sim$ is symmetric but it is neither reflexive ($\mathcal{A}$ cannot be semantically different from $\mathcal{A}$) nor transitive ($\mathcal{A} \sim \mathcal{A}'$ and $\mathcal{A}' \sim \mathcal{A}$ but $\mathcal{A} \nsim \mathcal{A}$). So we have a way to change an ABox syntactically to another one with the use of the set $\Gamma_{\mathcal{A}}$ by applying a finite number of times one-step changes to concept symbols on each subsequent ABox. Furthermore, by construction the two ABoxes are always

semantically different from each other and there is always a shortest path of $n > 0$ steps between them.

We can now utilize this construction in our effort of repairing an ABox with respect to an active TBox. Let $\mathcal{A}$ and $\mathcal{AT}$ be an ABox and an active TBox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathcal{AT}$ and let $\mathcal{R}_n^{\mathcal{A}} = \{\mathcal{A}' \mid \mathcal{A} \sim_n \mathcal{A}'\}$ be the set of ABoxes that can be reached from $\mathcal{A}$ by at least n steps. So for each $n > 0$ we have that the sets $\mathcal{R}_1^{\mathcal{A}}, \mathcal{R}_2^{\mathcal{A}}, \mathcal{R}_3^{\mathcal{A}}, \ldots$ are pairwise disjoint and their union is the set $\mathcal{R}^{\mathcal{A}} = \{\mathcal{A}' \mid \mathcal{A} \sim \mathcal{A}'\}$ of ABoxes that can be reached from $\mathcal{A}$ by an arbitrary number of steps. The next propositions give some important properties on the cardinality of these sets.

Proposition 1. *Let $\mathcal{A}$ and $\mathcal{AT}$ be an ABox and an active TBox respectively. Then for each $n > 0$ the set $\mathcal{R}_n^{\mathcal{A}}$ is finite.*

Proof. Let's start by noticing that since the ABox and the TBox are always finite, the set $S_{\mathcal{A}}$ containing their concept symbols is also finite. As a result, for each $A \in S_{\mathcal{A}}$ the sets $A_{\sqcup}, A_{\sqcap}$ and $A_{\neg}$ are also finite, since they are made up of disjunctions, conjunctions and negations between symbols of $S_{\mathcal{A}}$. Then the set $\Gamma_{\mathcal{A}:A}$ which is the union of the finite sets $A_{\sqcup}, A_{\sqcap}$ and $A_{\neg}$ is also finite, for all $A \in S_{\mathcal{A}}$. It follows that the set $\Gamma_{\mathcal{A}}$ of concepts that can be reached with one step from $S_{\mathcal{A}}$ is finite, since it comprises a finite union of finite sets.

Let's look initially at the set $\mathcal{R}_1^{\mathcal{A}}$. It comprises the ABoxes that are semantically different and can be reached with one step from $\mathcal{A}$. So by the definition, $\mathcal{A}' \in \mathcal{R}_1^{\mathcal{A}}$ iff $\mathcal{A}' = \mathcal{A}[A \mid C]$ or $\mathcal{A} = \mathcal{A}'[A \mid C]$ for some $A \in S_{\mathcal{A}}$, where $C \in \Gamma_{\mathcal{A}:A}$. But as the $A \in S_{\mathcal{A}}$ are finite and for each A the set $\Gamma_{\mathcal{A}:A}$ is also finite, there is a finite number of ABoxes such that $\mathcal{A}' = \mathcal{A}[A \mid C]$ or $\mathcal{A} = \mathcal{A}'[A \mid C]$. As a result the set $\mathcal{R}_1^{\mathcal{A}}$ is also finite. Next, consider that the set $\mathcal{R}_n^{\mathcal{A}}$ is finite for an arbitrary $n > 0$. It suffices to show that the set $\mathcal{R}_{n+1}^{\mathcal{A}}$ is also finite. Let's take an ABox $\mathcal{A}' \in \mathcal{R}_n^{\mathcal{A}}$ and create the set $\mathcal{R}_1^{\mathcal{A}'}$ of ABoxes that are semantically different and can be reached with one step from $\mathcal{A}'$. We already know that this set is finite. But by hypothesis, the set of ABoxes $\mathcal{A}' \in \mathcal{R}_n^{\mathcal{A}}$ is also finite and thus the union $\bigcup_{\mathcal{A}' \in \mathcal{R}_n^{\mathcal{A}}} \mathcal{R}_1^{\mathcal{A}'}$ is finite as well. It's easy to see that $\mathcal{R}_{n+1}^{\mathcal{A}} \subseteq \bigcup_{\mathcal{A}' \in \mathcal{R}_n^{\mathcal{A}}} \mathcal{R}_1^{\mathcal{A}'}$ since for each ABox $\mathcal{A}''$ which is at least $n+1$ steps away from $\mathcal{A}$ there is an ABox $\mathcal{A}'$ which is at least n steps away from $\mathcal{A}$ such that $\mathcal{A} \sim_n \mathcal{A}'$ and $\mathcal{A}' \sim_1 \mathcal{A}''$. Thus the set $\mathcal{R}_{n+1}^{\mathcal{A}}$ is also finite and the induction is complete.

Proposition 2. *Let $\mathcal{A}$ and $\mathcal{AT}$ be an ABox and an active TBox respectively. Then the set $\mathcal{R}^{\mathcal{A}}$ is finite.*

Proof. We will only provide a sketch of the proof. It suffices to show that $\mathcal{R}^{\mathcal{A}} = \bigcup_{n=1}^{m} \mathcal{R}_n^{\mathcal{A}}$ for some $m > 0$. Since we have a finite number of concept symbols, there is only a finite number of semantically different concepts that can be expressed by these symbols using the three boolean constructors. Furthermore, using these concepts in combination with the role symbols of $\mathcal{A}$ there is a finite number of semantically different concepts that can reach a specific role depth. But since for all concepts the role depth never changes between the ABox $\mathcal{A}$ and the ABoxes $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$, there will be a set of ABoxes $\mathcal{R}_n^{\mathcal{A}}$ for which each subsequent ABox constructed by the relation $\sim_1$ will have a semantically equivalent ABox belonging in a set $\mathcal{R}_m^{\mathcal{A}}$ for $m < n$. In other words, there is an $m > 0$ such that $\mathcal{R}_n^{\mathcal{A}} = \emptyset$ for all $n > m$ and $\mathcal{R}^{\mathcal{A}} = \bigcup_{n=1}^{m} \mathcal{R}_n^{\mathcal{A}}$.

Next we define a *syntactic modification* to be the update action needed in order to reach an ABox $\mathcal{A}'$ from an ABox $\mathcal{A}$ in one step using the set $\Gamma_{\mathcal{A}}$.

Definition 7. *Let $\mathcal{A}, \mathcal{A}'$ and $\mathcal{AT}$ be two ABoxes and an active TBox respectively such that $\mathcal{A} \sim_1 \mathcal{A}'$. The syntactic modification from $\mathcal{A}$ to $\mathcal{A}'$ is the singleton set*

$$U_{\mathcal{A}}^{\mathcal{A}'} = \{A \rightarrow C\} \quad \textit{if} \quad \mathcal{A}' = \mathcal{A}\,[A \,|\, C]$$

$$U_{\mathcal{A}}^{\mathcal{A}'} = \{C \rightarrow A\} \quad \textit{if} \quad \mathcal{A} = \mathcal{A}'\,[A \,|\, C]$$

where $C \in \Gamma_{\mathcal{A}:A}$.

Using this definition we can now define an *update sequence* to be the sequence of syntactic modifications needed in order to reach an ABox $\mathcal{A}'$ from an ABox $\mathcal{A}$ in n steps using the set $\Gamma_{\mathcal{A}}$.

Definition 8. *Let $\mathcal{A}, \mathcal{A}'$ and $\mathcal{AT}$ be two ABoxes and an active TBox respectively such that $\mathcal{A} \sim_n \mathcal{A}'$. The update sequence from $\mathcal{A}$ to $\mathcal{A}'$ is the sequence*

$$S_{\mathcal{A}}^{\mathcal{A}'} = (U_{\mathcal{A}_1}^{\mathcal{A}_2}, U_{\mathcal{A}_2}^{\mathcal{A}_3}, \ldots, U_{\mathcal{A}_{n-1}}^{\mathcal{A}_n}, U_{\mathcal{A}_n}^{\mathcal{A}_{n+1}})$$

where $\mathcal{A}_1, \ldots, \mathcal{A}_{n+1}$ are the semantically different ABoxes as in Definition 5.

Finally, if an ABox $\mathcal{A}$ can be syntactically modified to the semantically different ABox $\mathcal{A}'$ in at least n steps (i.e., if $\mathcal{A} \sim_n \mathcal{A}'$) through the update sequence $S_{\mathcal{A}}^{\mathcal{A}'}$, we write $\mathcal{A} \diamond S_{\mathcal{A}}^{\mathcal{A}'} = \mathcal{A}'$.

Notice that up until now we have not made use of the 'active' part of the TBox and only investigated the different ways to construct new ABoxes. The next step is to indicate what it means for an ABox to be repaired with respect to the active TBox. We will make use of similar notions that we already presented in Section 2.1 about active integrity constraints to show the relation between the two settings. We start by the definitions of *weak repair* and *PMA repair*.

Definition 9. *Let $\mathcal{A}$ and $\mathcal{T}$ be an ABox and a TBox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathcal{T}$.*

1. *A* weak repair *of $\mathcal{A}$ achieving $\mathcal{T}$ is an update sequence $S_{\mathcal{A}}^{\mathcal{A}'}$ such that $\mathcal{A} \diamond S_{\mathcal{A}}^{\mathcal{A}'}$ is consistent with respect to $\mathcal{T}$.*
2. *A* PMA repair *of $\mathcal{A}$ achieving $\mathcal{T}$ is a weak repair of $\mathcal{A}$ achieving $\mathcal{T}$ that is minimal with respect to the number of steps needed, i.e., there is no weak repair of $\mathcal{A}$ achieving $\mathcal{T}$ in fewer steps.*

Next we define the notion of *foundedness* on the level of syntactic modifications and on the level of update sequences.

Definition 10. *Let $\mathcal{A}$ and $\mathcal{AT}$ be an ABox and an active TBox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathcal{AT}$. A syntactic modification $U_{\mathcal{A}}^{\mathcal{A}'}$ is* founded *if there is an active constraint r on $\mathcal{AT}$ such that:*

1. *$\mathcal{A}$ is not consistent with respect to body(r).*

2. $\mathcal{A}'$ *is consistent with respect to body*(r).
3. $U_{\mathcal{A}}^{\mathcal{A}'}$ *either adds or removes a concept according to the update actions in head*(r).

Furthermore, an update sequence $S_{\mathcal{A}}^{\mathcal{A}'}$ *is founded if for every* $U \in S_{\mathcal{A}}^{\mathcal{A}'}$ *there is an active constraint* r *on* $\mathcal{AT}$ *such that* U *is founded.*

Finally, using the above definitions, we define *founded weak repairs* and *founded repairs* as follows.

Definition 11. *Let* $\mathcal{A}$ *and* $\mathcal{AT}$ *be an ABox and an active TBox respectively such that* $\mathcal{A}$ *is inconsistent with respect to* $\mathcal{AT}$.

1. *An update sequence* $S_{\mathcal{A}}^{\mathcal{A}'}$ *is a founded weak repair of* $\mathcal{A}$ *by* $\mathcal{AT}$ *if* $S_{\mathcal{A}}^{\mathcal{A}'}$ *is a weak repair of* $\mathcal{A}$ *achieving static*$(\mathcal{AT})$ *and* $S_{\mathcal{A}}^{\mathcal{A}'}$ *is founded.*
2. *An update sequence* $S_{\mathcal{A}}^{\mathcal{A}'}$ *is a founded repair of* $\mathcal{A}$ *by* $\mathcal{AT}$ *if* $S_{\mathcal{A}}^{\mathcal{A}'}$ *is a PMA repair of* $\mathcal{A}$ *achieving static*$(\mathcal{AT})$ *and* $S_{\mathcal{A}}^{\mathcal{A}'}$ *is founded.*

Summing up, let $\mathcal{A}$ be an ABox and $\mathcal{AT}$ an active TBox such that $\mathcal{A}$ is inconsistent with respect to $\mathcal{AT}$. A repaired ABox with respect to $\mathcal{AT}$ is any ABox $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$ such that $S_{\mathcal{A}}^{\mathcal{A}'}$ is a founded weak repair of $\mathcal{A}$ by $\mathcal{AT}$. A minimally repaired ABox with respect to $\mathcal{AT}$ is any ABox $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$ such that $S_{\mathcal{A}}^{\mathcal{A}'}$ is a founded repair of $\mathcal{A}$ by $\mathcal{AT}$. Since by Proposition 2 there is a finite number of ABoxes that we can construct step by step from the initial ABox using the set $\Gamma_{\mathcal{A}}$, the sets of repaired and minimally repaired ABoxes with respect to $\mathcal{AT}$ are also finite. This means that we can start from the set of ABoxes $\mathcal{R}_1^{\mathcal{A}}$ and continue searching all the sets $\mathcal{R}_n^{\mathcal{A}}$ for $n > 0$ until we find a minimally repaired ABox with respect to $\mathcal{AT}$.

Remark 1. If we are working on $\mathcal{ALCO}$, let $S_{\mathcal{A}}^*$ be the extension of $S_{\mathcal{A}}$ such that it also includes the nominals of the ABox $\mathcal{A}$ and TBox $\mathcal{AT}$ and define $A_{\sqcup}^*, A_{\sqcap}^*, A_{\neg}^*, \Gamma_{\mathcal{A}:A}^*$ and $\Gamma_{\mathcal{A}}^*$ accordingly. We can then also extend all the definitions of this subsection to incorporate nominals and Propositions 1 and 2 are still valid.

We now return to the original example of Figures 1 and 2 and examine an ABox (which is inconsistent with respect to this TBox) and one of its possible repairs. Let $\mathcal{AT}$ be the active TBox of Figure 2. Let the ABox $\mathcal{A}$, written in the language of $\mathcal{ALC}$, consist of the following assertions:

$John$: Person $\sqcap$ Married $\sqcap$ Bachelor $\sqcap \exists$ hasChild. (Divorced $\sqcap$ Widowed)

$Mary$: Person $\sqcap \neg$Married $\sqcap \neg$Divorced $\sqcap \neg$Bachelor $\sqcap \neg$Widowed

A minimally repaired ABox $\mathcal{A}'$ with respect to $\mathcal{AT}$ is the following:

$John$: Person $\sqcap$ Married $\sqcap \exists$ hasChild. Divorced

$Mary$: Person $\sqcap \neg$Married $\sqcap \neg$Divorced $\sqcap$ Bachelor $\sqcap \neg$Widowed

A founded repair of $\mathcal{A}$ by $\mathcal{AT}$ then is the update sequence $S_{\mathcal{A}}^{\mathcal{A}'} = (U_1, U_2, U_3)$ where $U_1 = \{\text{Married} \sqcap \text{Bachelor} \rightarrow \text{Married}\}, U_2 = \{\text{Divorced} \sqcap \text{Widowed} \rightarrow \text{Divorced}\}$ and $U_3 = \{\neg\text{Bachelor} \rightarrow \text{Bachelor}\}$. Notice also that if we replace U_1 by

$U_1' = \{\mathsf{Bachelor} \rightarrow \neg\mathsf{Bachelor}\}$ and U_2 by $U_2' = \{\mathsf{Widowed} \rightarrow \neg\mathsf{Widowed}\}$ in $S_{\mathcal{A}}^{\mathcal{A}'}$, the new update sequence $S_{\mathcal{A}}^{\mathcal{A}''} = (U_1', U_2', U_3)$ is also a founded repair of $\mathcal{A}$ by $\mathcal{A}\mathcal{T}$.

Finally, let us note that it may not be possible to acquire a founded repair in cases where a concept is defined in the TBox and does not explicitly appear inside the ABox but is inferred, as shown in the table below. Using the active TBox of the first column we would not be able to provide a founded repair of any of the two ABoxes in the last row. Given a more precise active TBox, however, this would not pose a problem. We can witness this with the active TBox of the second column, which provides a founded repair for the second assertion, and even more with the active TBox of the third column, which provides a founded repair for both assertions.

$A \sqcap B \sqsubseteq \bot \rightarrow remove(B)$	$A \sqcap B \sqsubseteq \bot \rightarrow remove(B)$	$A \sqcap E \sqcap F \sqsubseteq \bot \rightarrow remove(E)$
$B \equiv E \sqcap F$	$B \sqsubseteq E \sqcap F \rightarrow remove(B)$	$A \sqcap B \sqsubseteq \bot \rightarrow remove(B)$
	$E \sqcap F \sqsubseteq B \rightarrow remove(E)$	$B \equiv E \sqcap F$
$\alpha : A \sqcap E \sqcap F$ or $\alpha : A \sqcap B \sqcap E \sqcap F$		

3.3 A semantic approach

In Dynamic Logic the most prominent role is that of the *programs*, which are usually denoted by π. Formulas of the form $\langle\pi\rangle\varphi$ express the fact that "there is a possible execution of the program π after which φ is the case". Many extensions and variants have been proposed in the literature, with DL-PA (standing for Dynamic Logic of Propositional Assignments [14, 3, 2]) being recently utilised in the study of more dynamic ways to repair databases taking into account a set of active integrity constraints [10]. In that setting, atomic programs are update actions of the form $p \leftarrow \top$ and $p \leftarrow \bot$ denoting the insertion and the deletion of the propositional variable p.

Looking in a somewhat similar direction, but on the setting of Description Logic, a logical next step would be to define the atomic programs to be of the form $A(a) \leftarrow \top$ and $A(a) \leftarrow \bot$, where A is an atomic concept and a an individual. Similar to DL-PA, the atomic program $A(a) \leftarrow \top$ denotes the update of an ABox with the assertion $a : A$, while $A(a) \leftarrow \bot$ denotes the update with the assertion $a : \neg A$. For an ABox $\mathcal{A}$, we denote by $|\mathcal{A}|$ the set of interpretations $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{A}$. Note that, although an ABox may have several different syntactic forms, all of them are semantically equivalent and are represented by the unique set $|\mathcal{A}|$.

We use the following grammar in order to define arbitrary programs and formulas:

$$\varphi ::= C(a) \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi$$
$$\pi ::= \alpha \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi^- \mid \varphi?$$

where $C(a)$ is a concept assertion and α an atomic program of the form $A(a) \leftarrow \top$ or $A(a) \leftarrow \bot$ for atomic concepts A and individuals a. The operators $;, \cup, ^*, ^-$ and $?$ are considered familiar from Propositional Dynamic Logic. The semantics of these formulas and programs are once again equivalent to the semantics of Propositional Dynamic Logic with exceptions being the following:

$|\mathcal{A}| \in \|C(a)\|$ iff $\mathcal{A} \models C(a)$
$|\mathcal{A}| \in \|\langle\pi\rangle\varphi\|$ iff there exists $\mathcal{A}'$ such that $\langle|\mathcal{A}|, |\mathcal{A}'|\rangle \in \|\pi\|$ and $\mathcal{A}' \models \varphi$

$\langle |\mathcal{A}|, |\mathcal{A}'| \rangle \in \|\alpha\|$ iff $|\mathcal{A}'| = |\mathcal{A}| \diamond \{\alpha\}$
$\langle |\mathcal{A}|, |\mathcal{A}| \rangle \in \|\varphi?\|$ iff $\mathcal{A} \models \varphi$

where $|\mathcal{A}| \diamond \{\alpha\} = \{\mathcal{I} \diamond \{\alpha\} \mid \mathcal{I} \models \mathcal{A}\}$ as defined in [17]. Using these semantics now we can define the following programs, denoting the n-step and arbitrary-step syntactic modifications of Section 3.2:

$$\|mod_n\| = \{ < |\mathcal{A}|, |\mathcal{A}'| > \text{ such that } \mathcal{A} \sim_n \mathcal{A}' \}$$

$$\|mod\| = \{ < |\mathcal{A}|, |\mathcal{A}'| > \text{ such that } \mathcal{A} \sim \mathcal{A}' \}$$

In other words:

$$< \bigcup I_i, \bigcup I'_i > \in \|mod_n\| \text{ iff } \bigcup I_i \models \mathcal{A}, \bigcup I'_i \models \mathcal{A}' \text{ and } \mathcal{A} \sim_n \mathcal{A}'$$

$$< \bigcup I_i, \bigcup I'_i > \in \|mod\| \text{ iff } \bigcup I_i \models \mathcal{A}, \bigcup I'_i \models \mathcal{A}' \text{ and } \mathcal{A} \sim \mathcal{A}'$$

Now, given a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ define $left[r]$ to be the left side of the constraint r in $\mathcal{T}$, $right[r]$ to be the right side of the constraint r in $\mathcal{T}$ and $Ind(\mathcal{A})$ the set of individuals in $\mathcal{A}$. Then we define $consistent(\mathcal{A}, \mathcal{T})$ to be the formula:

$$\bigwedge_{\substack{r \in \mathcal{T} \\ a \in Ind(\mathcal{A})}} (left[r](a) \rightarrow right[r](a))$$

denoting the fact that $\mathcal{A} \models consistent(\mathcal{A}, \mathcal{T})$ iff the ABox $\mathcal{A}$ is consistent with respect to the TBox $\mathcal{T}$.

Finally, let $\mathcal{T}$ and $\mathcal{A}$ be a TBox and an ABox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathcal{T}$. A repaired ABox $\mathcal{A}'$ satisfying the constraints r in $\mathcal{T}$ could be computed by the following semantic procedure:

$$\langle |\mathcal{A}|, |\mathcal{A}'| \rangle \in \| mod ; consistent(\mathcal{A}', \mathcal{T}) ? \|$$

The next obvious step would be to continue this semantic investigation by taking into account the active constraints of an active TBox and providing minimally repaired ABoxes as well as their founded repairs through programs of Dynamic Logic.

4 Conclusion

In this paper we explored the ways in which active constraints, which originate from the database community, could be integrated into the TBoxes of Description Logic. Based on these "active" TBoxes, we then investigated a mainly syntactic approach of transforming an ABox (inconsistent with an active TBox) step-by-step by syntactic modifications to a repaired one, conforming to the preferred update actions found in the active constraints of the extended TBox.

A semantic approach was also previewed, showing the ways in which programs of Dynamic Logic could be useful in providing a semantic solution to this problem. Although a brief one, it laid some foundations for future work, motivated by the way DL-PA was utilised to provide new kinds of repairs in the database literature.

The most crucial thing though would be to make the landscape of active TBoxes and the associated repairs more clear and intuitive. In this work, we provided a first (small) step into this direction and believe that it is one that the description logic community would find interesting enough to delve into.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Balbiani, P., Herzig, A., Schwarzentruber, F., Troquard, N.: DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE. CoRR abs/1411.7825 (2014), http://arxiv.org/abs/1411.7825
3. Balbiani, P., Herzig, A., Troquard, N.: Dynamic logic of propositional assignments: a well-behaved variant of PDL. In: Kupferman, O. (ed.) Logic in Computer Science (LICS). IEEE (2013)
4. Bienvenu, M., Bourgaux, C., Goasdoué, F.: Querying inconsistent description logic knowledge bases under preferred repair semantics. In: Brodley, C.E., Stone, P. (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. pp. 996–1002. AAAI Press (2014), http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8231
5. Caroprese, L., Greco, S., Zumpano, E.: Active integrity constraints for database consistency maintenance. IEEE Trans. Knowl. Data Eng. 21(7), 1042–1058 (2009)
6. Caroprese, L., Trubitsyna, I., Zumpano, E.: View updating through active integrity constraints. In: Dahl, V., Niemelä, I. (eds.) ICLP. Lecture Notes in Computer Science, vol. 4670, pp. 430–431. Springer (2007)
7. Caroprese, L., Truszczynski, M.: Declarative semantics for active integrity constraints. In: de la Banda, M.G., Pontelli, E. (eds.) ICLP. Lecture Notes in Computer Science, vol. 5366, pp. 269–283. Springer (2008)
8. Caroprese, L., Truszczynski, M.: Active integrity constraints and revision programming. TPLP 11(6), 905–952 (2011)
9. Cruz-Filipe, L.: Optimizing computation of repairs from active integrity constraints. In: Beierle, C., Meghini, C. (eds.) FoIKS. Lecture Notes in Computer Science, vol. 8367, pp. 361–380. Springer (2014)
10. Feuillade, G., Herzig, A.: A dynamic view of active integrity constraints. In: Ferme, E., Leite, J. (eds.) European Conference on Logics in Artificial Intelligence (JELIA), Madeira, 24/09/2014-26/09/2014. pp. 486–499. Springer, http://www.springerlink.com (septembre 2014), http://oatao.univ-toulouse.fr/13171/
11. Flesca, S., Greco, S., Zumpano, E.: Active integrity constraints. In: Moggi, E., Warren, D.S. (eds.) PPDP. pp. 98–107. ACM (2004)
12. Flouris, G., d'Aquin, M., Antoniou, G., Pan, J.Z., Plexousakis, D.: Special issue on ontology dynamics. J. Log. Comput. 19(5), 717–719 (2009), http://dx.doi.org/10.1093/logcom/exn046
13. Flouris, G., Plexousakis, D., Antoniou, G.: Updating dls using the AGM theory: A preliminary study. In: Horrocks, I., Sattler, U., Wolter, F. (eds.) Proceedings of the 2005 International Workshop on Description Logics (DL2005), Edinburgh, Scotland, UK, July 26-28, 2005. CEUR Workshop Proceedings, vol. 147. CEUR-WS.org (2005), http://ceur-ws.org/Vol-147/26-Flouris.pdf
14. Herzig, A., Lorini, E., Moisan, F., Troquard, N.: A dynamic logic of normative systems. In: Walsh, T. (ed.) International Joint Conference on Artificial Intelligence (IJCAI). pp. 228–233. IJCAI/AAAI, Barcelona (2011), http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html
15. Herzig, A., Rifi, O.: Propositional belief base update and minimal change. Artificial Intelligence Journal 115(1), 107–138 (Nov 1999), ../Herzig/P/aij99.html

16. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Hitzler, P., Lukasiewicz, T. (eds.) Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6333, pp. 103–117. Springer (2010), http://dx.doi.org/10.1007/978-3-642-15918-3_9
17. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Foundations of instance level updates in expressive description logics. Artificial Intelligence 175(18), 2170–2197 (2011)
18. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Sem. 7(2), 74–89 (2009), https://doi.org/10.1016/j.websem.2009.02.001
19. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Fox, M., Poole, D. (eds.) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press (2010), http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1931
20. Winslett, M.A.: Reasoning about action using a possible models approach. In: Proc. 7th Conf. on Artificial Intelligence (AAAI'88). pp. 89–93. St. Paul (1988)
21. Winslett, M.A.: Updating Logical Databases. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (1990)