THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Christos RANTSOUDIS

Le 10 décembre 2018

Bases de connaissance et actions de mise à jour préférées : à la recherche de consistance au travers des programmes de la logique dynamique

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Olivier GASQUET et Guillaume FEUILLADE

Jury

Mme Laura GIORDANO, Rapporteur
Mme Renate SCHMIDT, Rapporteur
M. Andreas HERZIG, Examinateur
Mme Meghyn BIENVENU, Examinateur
M. Olivier GASQUET, Directeur de thèse
Mme Laure VIEU, Président

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :
Christos RANTSOUDIS
le lundi 10 décembre 2018

Titre :

Knowledge Bases and Preferred Update Actions:
Searching for Consistency through Dynamic Logic Programs

École doctorale et discipline ou spécialité :
ED MITT : Domaine STIC : Intelligence Artificielle

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Directeur/trice(s) de Thèse :

Olivier GASQUET et Guillaume FEUILLADE

Jury :
Mme Laura GIORDANO, Rapporteur
Mme Renate SCHMIDT, Rapporteur
M. Andreas HERZIG, Examinateur
Mme Meghyn BIENVENU, Examinateur
M. Olivier GASQUET, Directeur de thèse
Mme Laure VIEU, Président
M. Guillaume FEUILLADE, Invité
M. Martin STRECKER, Invité

Abstract

In the database literature it has been proposed to resort to active integrity constraints in order to restore database integrity. Such active integrity constraints consist of a classical constraint together with a set of preferred update actions that can be triggered when the constraint is violated. In the first part of this thesis, we review the main repairing routes that have been proposed in the literature and capture them by means of Dynamic Logic programs. The main tool we employ for our investigations is the recently introduced logic DL-PA, which constitutes a variant of PDL. We then go on to explore a new, dynamic kind of database repairing whose computational complexity and general properties are compared to the previous established approaches.

In the second part of the thesis we leave the propositional setting and pursue to adapt the aforementioned ideas to higher level languages. More specifically, we venture into Description Logics and investigate extensions of TBox axioms by update actions that denote the preferred ways an ABox should be repaired in case of inconsistency with the axioms of the TBox. The extension of the TBox axioms with these update actions constitute new, *active* TBoxes. We tackle the problem of repairing an ABox with respect to such an active TBox both from a syntactic as well as a semantic perspective. Given an initial ABox, the syntactic approach allows us to construct a set of new ABoxes out of which we then identify the most suitable repairs. On the other hand, for the semantic approach we once again resort to a dynamic logic framework and view update actions, active inclusion axioms and repairs as programs. Given an active TBox $\mathsf{a}\mathcal{T}$, the framework allows to check (1) whether a set of update actions is able to repair an ABox according to the active axioms of $\mathsf{a}\mathcal{T}$ by interpreting the update actions *locally* and (2) whether an ABox $\mathcal{A}'$ is the repair of a given ABox $\mathcal{A}$ under the active axioms of $\mathsf{a}\mathcal{T}$ using a bounded number of computations by interpreting the update actions *globally*. After discussing the strong points of each direction, we conclude by combining the syntactic and semantic investigations into a cohesive approach.

Keywords: Active Integrity Constraints, Dynamic Logic, Description Logic, Inconsistency Management, Database Repairing

Résumé

Dans la littérature sur les bases de données, il a été proposé d'utiliser des contraintes d'intégrité actives afin de restaurer l'intégrité de la base. Ces contraintes d'intégrité actives consistent en une contrainte classique augmentée d'un ensemble d'actions de mise à jour préférées qui peuvent être déclenchées quand la contrainte est violée. Dans la première partie de cette thèse, nous passons en revue les principales stratégies de réparation qui ont été proposées dans la littérature et proposons une formalisation par des programmes de la Logique Dynamique. L'outil principal que nous employons dans notre recherche est la logique DL-PA, une variante de PDL récemment introduite. Nous explorons ensuite une nouvelle façon dynamique de réparer les bases de données et comparons sa complexité calculatoire et ses propriétés générales aux approches classiques.

Dans la seconde partie de la thèse nous abandonnons le cadre propositionnel et adaptons les idées susmentionnées à des langages d'un niveau supérieur. Plus précisément, nous nous intéressons aux Logiques de Description, et étudions des extensions des axiomes d'une TBox par des actions de mise à jour donnant les manières préférées par lesquelles une ABox doit être réparée dans le cas d'une inconsistance avec les axiomes de la TBox. L'extension des axiomes d'une TBox avec de telles actions de mise à jour constitue une nouvelle TBox, qui est *active*. Nous nous intéressons à la manière de réparer une ABox en rapport avec une telle TBox active, du point de vue syntaxique ainsi que du point de vue sémantique. Étant donnée une ABox initiale, l'approche syntaxique nous permet de construire un nouvel ensemble d'ABox dans lequel nous identifions les réparations les mieux adaptées. D'autre part, pour l'approche sémantique, nous faisons de nouveau appel au cadre de la logique dynamique et considérons les actions de mise à jour, les axiomes d'inclusion actives et les réparations comme des programmes. Étant donné une TBox active $\mathsf{a}\mathcal{T}$, ce cadre nous permet de vérifier (1) si un ensemble d'actions de mise à jour est capable de réparer une ABox en accord avec les axiomes actifs d'$\mathsf{a}\mathcal{T}$ en effectuant une interprétation *locale* des actions de mise à jour et (2) si une ABox $\mathcal{A}'$ est la réparation d'une ABox donnée $\mathcal{A}$ sous les axiomes actifs d'$\mathsf{a}\mathcal{T}$ moyennant un nombre borné de calculs, en utilisant une interprétation *globale* des actions de mise à jour. Après une discussion des avantages de chaque approche nous concluons en proposant une intégration des approches syntaxiques et sémantiques dans une approche cohésive.

Mots clefs: Contraintes d'Intégrité Active, Logique Dynamique, Logique de Description, Gestion de l'Inconsistance, Réparation de Bases de Données

Acknowledgements

During my stay in Toulouse there were a number of people who played an instrumental and pivotal role in the successful writing of this PhD thesis. Coming from all parts of the world they were my friends, advisors, colleagues and mentors, and without their support and guidance writing this thesis would never be truly possible.

First and foremost, I am truly grateful to my supervisors Guillaume Feuillade, Olivier Gasquet and Andreas Herzig for helping make this thesis a reality and shaping me as a person as well as a researcher. Their everyday guidance, both scientific and nonscientific, in combination with their encouragement and advice inspired me in many, important ways. More specifically, I would like to thank Andreas for his immense patience and knowledge, his always positive attitude, and his constant willingness to help, all traits that kept me going no matter the difficulties we faced. I'll also be forever indebted to Guillaume for his critical help, his friendly nature and all the invaluable conversations we had. I could not have asked for better advisors and will always strive to follow their example.

I'm also grateful to my friends and colleagues in office, the people who made my everyday life extremely pleasant and joyful. Going forward we may not see each other as often, but these people became part of my life, inspired and influenced me without their knowledge and made my time in Toulouse an absolute delight. Huge thank you to Zhanhao Xiao, Maël Valais, Usman Younus, Martin Dieguez, Ezgi Iraz Su, Julien Hay, Julien Vianey, Arianna Novaro, Maryam Rostamigiv, David Fernández-Duque, Petar Iliev, Victor David, Saul Gonzalez, Elise Perrotin and Dragan Doder.

Next, I would like to express my gratitude to the people of the lab, from the researchers on our team to the always friendly and helpful administrative staff. It was a real pleasure to meet and interact with them on a daily basis. A special mention to Stergos Afantenos, whose valuable advice and conversations during break time was something to always look forward to. Outside the lab, I would like to mention and thank Costas Koutras for keeping me extra busy and giving me extra advice on all things work and life related.

Last but not least, I would like to thank my family and friends in my home country for their unconditional love and support during these three years, most of which we had to regrettably be afar. Special thanks to those who visited me and gave me something to look forward to in times of need. I truly wouldn't have achieved anything if it weren't for the people closest to me, albeit the farthest away.

Contents

List of Figures

CHAPTER 1

Introduction

Contents

1.1 Motivation

One of the most important (but notoriously difficult) issues in the database and AI literature is the problem of updating a database under a set of *integrity constraints*. The latter are usually expressed by logical formulas and their role is to impose conditions that every state of the database must satisfy. In the course of database maintenance several changes are applied to the databases and checking whether these constraints are still satisfied is of the highest priority. When a database fails to satisfy the integrity constraints, it has to be repaired in order to *restore integrity*. Given a database, the procedure of repairing and restoring its consistency with respect to a set of integrity constraints has been extensively studied in the last decades [Abiteboul, 1988, Ceri et al., 1994, Bertossi, 2011]. These approaches propose several possible repairs as candidates for integrity maintenance and it seems essential to identify which types of repairs are more suitable, given the fact that the number of all possible repairs can be remarkably large. Given this, the most preva-

lent have become those that are based on the *minimality of change* principle [Winslett, 1990, Herzig and Rifi, 1999, Chomicki and Marcinkowski, 2005]. Despite this, however, the need to have 'more informed' ways of maintaining database integrity arose.

In light of this, *active integrity constraints* were proposed as an extension of integrity constraints (or *static* constraints) with update actions, each one suggesting the preferred update method when an inconsistency arises between the database and a constraint [Flesca et al., 2004,Caroprese et al., 2009,Caroprese and Truszczynski, 2011,Cruz-Filipe, 2014]. For example, the integrity constraint $(\forall X)[\mathsf{Bachelor}(X) \wedge \mathsf{Married}(X) \rightarrow \bot]$ which says that no one should have the property of being a bachelor and married at the same time can be turned into the active constraint $(\forall X)[\mathsf{Bachelor}(X) \wedge \mathsf{Married}(X) \rightarrow \bot, \{-\mathsf{Bachelor}(X)\}]$, whose meaning is that when there is a person in the database who has both the status of being a bachelor and the status of being married then the preferred repair is to remove from the database the bachelor status (as opposed to removing the married status) since married status can be achieved from being bachelor but not the other way. In this way, the possible repairs are narrowed down as well as better match designer preferences when maintaining the database. In the propositional case, an active integrity constraint can be represented as a couple $r = \langle \mathsf{C}(r), \mathsf{R}(r) \rangle$ where $\mathsf{C}(r)$ is a boolean formula (called the static part of r and denoting a static constraint) and $\mathsf{R}(r)$ is a set of update actions, each of which is of the form $+p$ or $-p$ for some atomic formula p. The idea is that (1) when $\mathsf{C}(r)$ is false then the constraint r is violated and (2) a violated constraint can be repaired by performing one or more of the update actions in $\mathsf{R}(r)$. The two most prevalent types of repairs w.r.t. a set of active integrity constraints are the *founded* and the *justified* repairs. Note that while with these methods one can greatly reduce the number of possible repairs, different choices between update actions in $\mathsf{R}(r)$ can still lead to different repairing routes or even no repairs at all (for example when $\mathsf{R}(r)$ is the empty set).

Applying the same idea to the Description Logic setting would result in the extension of the TBox axioms with *preferred* update actions that 'dictate' the changes that should be imposed on an ABox in order to achieve consistency with a TBox and this has not yet been directly pursued in the DL literature. It seems very natural though that sometimes ontology engineers should be given the means to easily express such preferences without having to worry that unwanted 'repairing routes' will be followed in case there is an

inconsistency between an ABox and a TBox. Consider for instance the following TBox:

$$\mathcal{T} = \{\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}, \mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot\}$$

and an ontology engineer being able to 'enhance' it to two possible *active* TBoxes, viz. $\mathsf{a}\mathcal{T}_1 = \{\eta_1, \eta_2\}$ or $\mathsf{a}\mathcal{T}_2 = \{\eta_3, \eta_4\}$ where:

$$\eta_1 : \langle \mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}, \{+\mathsf{Male}, +\mathsf{Parent}\}\rangle$$
$$\eta_2 : \langle \mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot, \{-\mathsf{OnlyChild}\}\rangle$$
$$\eta_3 : \langle \mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}, \{-\mathsf{Father}\}\rangle$$
$$\eta_4 : \langle \mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot, \{-\mathsf{hasSibling}.\top\}\rangle$$

We can witness how, through these enhanced concept inclusions, one can be more specific in the update actions that s/he prefers when repairing an ABox that is inconsistent with $\mathcal{T}$: the active axioms of $\mathsf{a}\mathcal{T}_1$ dictate that an individual who is a father should remain a father in case of inconsistency, whereas an individual who has siblings should change its status and not be an only child anymore. Similarly for $\mathsf{a}\mathcal{T}_2$, where '$-\mathsf{hasSibling}.\top$' removes all relations between individuals that violate the axiom and individuals satisfying $\top$ (i.e., all individuals), thus stating that an only child who has siblings should drop its 'sibling' relationship with everyone and stay an only child. In practice, consider the following ABox:

$$\mathcal{A} = \{\mathsf{John} : \mathsf{Male} \sqcap \mathsf{Father} \sqcap \neg\mathsf{Parent}, \mathsf{Mary} : \mathsf{OnlyChild}, \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})\}$$

A repaired ABox then according to $\mathsf{a}\mathcal{T}_1$ should be the following:

$$\mathcal{A}_1 = \{\mathsf{John} : \mathsf{Male} \sqcap \mathsf{Father} \sqcap \mathsf{Parent}, \mathsf{Mary} : \neg\mathsf{OnlyChild}, \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})\}$$

whereas a repaired ABox according to $\mathsf{a}\mathcal{T}_2$ should be the following:

$$\mathcal{A}_2 = \{\mathsf{John} : \mathsf{Male} \sqcap \neg\mathsf{Father} \sqcap \neg\mathsf{Parent}, \mathsf{Mary} : \mathsf{OnlyChild}\}$$

During the course of this thesis, we will attempt to materialize the above intuitions in a number of different formats and examine how the various definitions of repairs from the

database literature translate and behave in the DL setting.

1.2 Background and State of the Art

We begin by presenting some background that is needed for the later chapters. Meanwhile, we discuss the relevant state of the art of the areas that this thesis is based on. More specifically, we give the basics of DL-PA, a variant of Propositional Dynamic Logic that Chapter 2 is based on, give a survey of the literature on Active Integrity Constraints from their origins to later developments, as well as recall the fundamental notions of Description Logics and the most prevalent methods that exist on repairing inconsistent DL knowledge bases.

1.2.1 Dynamic Logic of Propositional Assignments

Dynamic Logic [Pratt, 1976, Harel et al., 2000] extends modal logic with *dynamic* operators that are designed to reason about the behavior of (computer) programs: each modal operator has as argument a (computer) program and allows to express facts which hold after that program is executed. Propositional Dynamic Logic PDL [Fischer and Ladner, 1979] is the 'fragment' of Dynamic Logic that deals with propositional statements.

The first studies of assignments in the context of Dynamic Logic are due, among others, to Tiomkin and Makowski and van Eijck [Tiomkin and Makowsky, 1985, van Eijck, 2000]. Dynamic Logic of Propositional Assignments DL-PA is a variant of PDL that was introduced in [Herzig et al., 2011] and was further studied in [Balbiani et al., 2013, Balbiani et al., 2014]. Evidence for its widespread applicability was provided in several recent publications, including belief update and belief revision, argumentation, planning and reasoning about knowledge [Herzig, 2014, Doutre et al., 2014, Herzig et al., 2014, Herzig et al., 2016, Cooper et al., 2016, Charrier and Schwarzentruber, 2017]. We briefly recall syntax and semantics.

Language

Consider a countable set of propositional variables (alias atomic formulas) $\mathbb{P} = \{p, q, \ldots\}$. An *update action* is of the form $p \leftarrow \top$ and $p \leftarrow \bot$, for $p \in \mathbb{P}$. The former is the insertion of p and the latter is the deletion of p. We denote the set of all update actions by $\mathbb{U}$. A set of update actions $U \subseteq \mathbb{U}$ is *consistent* if it does not contain both $p \leftarrow \top$ and $p \leftarrow \bot$, for some p. The language of DL-PA is defined by the following grammar:

$$\varphi ::= p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi$$
$$\pi ::= \alpha \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi^{\mathsf{c}} \mid \varphi?$$

where p ranges over the set of atomic formulas $\mathbb{P}$ and α ranges over the set of update actions $\mathbb{U}$. The other boolean connectives $\wedge$, $\rightarrow$, and $\leftrightarrow$ are abbreviated in the usual way. In DL-PA, update actions are singletons and are called *atomic assignments*. The operators of sequential composition ";", nondeterministic composition "$\cup$", finite iteration (the so-called Kleene star) "$(.)^*$" and test "$(.)?$" are the familiar operators of PDL. The operator "$(.)^{\mathsf{c}}$" is the converse operator. The formula $\langle\pi\rangle\varphi$ is read "there is an execution of π after which φ is true". So e.g. $\langle p \leftarrow \bot^{\mathsf{c}}\rangle(p \wedge q)$ expresses that $p \wedge q$ is true after some execution of $p \leftarrow \bot^{\mathsf{c}}$, i.e., $p \wedge q$ was true before p was set to false. The star-free fragment of DL-PA is the subset of the language made up of formulas without the Kleene star.

Let $\mathbb{P}_\varphi$ denote the set of variables from $\mathbb{P}$ occurring in formula φ, and let $\mathbb{P}_\pi$ denote the set of variables from $\mathbb{P}$ occurring in program π. For example, $\mathbb{P}_{p \leftarrow \top \cup q \leftarrow \bot} = \{p, q\} = \mathbb{P}_{\langle p \leftarrow \bot\rangle q}$. Moreover, a *literal* is an element of $\mathbb{P}$ or the negation of an element of $\mathbb{P}$ and a *clause* is a disjunction of literals.

Several program abbreviations are familiar from PDL. First, $\mathbf{skip}$ abbreviates $\top?$ and $\mathbf{fail}$ abbreviates $\bot?$. Second, $\mathbf{if}\ \varphi\ \mathbf{then}\ \pi_1\ \mathbf{else}\ \pi_2$ is expressed by $(\varphi?;\pi_1) \cup (\neg\varphi?;\pi_2)$. Third, the loop $\mathbf{while}\ \varphi\ \mathbf{do}\ \pi$ is expressed by $(\varphi?;\pi)^*;\neg\varphi?$. The nondeterministic composition $\bigcup_{\alpha\in U} \alpha$ equals $\mathbf{fail}$ when U is empty. Furthermore, let π^+ abbreviate the program $\pi;\pi^*$. Assignments of literals to variables are introduced by means of the following two abbreviations:

$$p \leftarrow q = \mathbf{if}\ q\ \mathbf{then}\ p \leftarrow \top\ \mathbf{else}\ p \leftarrow \bot \qquad p \leftarrow \neg q = \mathbf{if}\ q\ \mathbf{then}\ p \leftarrow \bot\ \mathbf{else}\ p \leftarrow \top$$

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. In particular, the program $p \leftarrow \neg p$ flips the truth value of p. Note that both abbreviations have constant length, namely 14. Finally and as usual in modal logic, $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$.

Semantics

Valuations are subsets of $\mathbb{P}$ and are denoted by V, V_1, V_2, etc. The set of all valuations is therefore $\mathbb{V} = 2^{\mathbb{P}}$. It will sometimes be convenient to write $V(p) = \top$ instead of $p \in V$ and $V(p) = \bot$ instead of $p \notin V$. A valuation determines the truth value of every boolean formula. The set of valuations where A is true is noted $||A||$. We sometimes write $V \models A$ instead of $V \in ||A||$. The *update* of a valuation V by a set of update actions U is defined as:

$$V \diamond U = \Big(V \setminus \{p \ : \ p \leftarrow \bot \in U\}\Big) \cup \{p \ : \ p \leftarrow \top \in U\}$$

So all the deletions are applied in parallel first, followed by the parallel application of all insertions. We could as well have chosen another order of application. When U is consistent then all of them lead to the same result. In particular:

Proposition 1.1. *Let* $\{\alpha_1, \ldots, \alpha_n\}$ *be a consistent set of update actions. Let* $\langle k_1 \ldots k_n\rangle$ *be some permutation of* $\langle 1 \ldots n\rangle$. *Then* $V \diamond \{\alpha_1, \ldots, \alpha_n\} = \Big(\ldots (V \diamond \{\alpha_{k_1}\}) \ldots\Big) \diamond \{\alpha_{k_n}\}$.

DL-PA programs are interpreted as relations between valuations. The atomic programs α update valuations in the aforementioned way and complex programs are interpreted just as in PDL by mutual recursion. Figure 1.1 gives the interpretation of formulas and programs, where $\circ$ is relation composition and $(.)^{-1}$ is relation inverse.

A formula φ is DL-PA *valid* iff $||\varphi|| = 2^{\mathbb{P}} = \mathbb{V}$. It is DL-PA *satisfiable* iff $||\varphi|| \neq \emptyset$. For example, the formulas $\langle p \leftarrow \bot\rangle \top$, $\langle p \leftarrow \top\rangle \varphi \ \leftrightarrow \ \neg\langle p \leftarrow \top\rangle\neg\varphi$, $\langle p \leftarrow \top\rangle p$ and $\langle p \leftarrow \bot\rangle\neg p$ are all valid.

Observe that if p does not occur in φ then $\varphi \rightarrow \langle p \leftarrow \top\rangle \varphi$ and $\varphi \rightarrow \langle p \leftarrow \bot\rangle \varphi$ are valid. This is due to the following semantical property that is instrumental in the proof of several results involving DL-PA.

$$
\begin{array}{rcl}
||p|| & = & \{V \,:\, p \in V\} \\
||\top|| & = & \mathbb{V} = 2^{\mathbb{P}} \\
||\bot|| & = & \emptyset \\
||\neg\varphi|| & = & 2^{\mathbb{P}} \setminus ||\varphi|| \\
||\varphi \vee \psi|| & = & ||\varphi|| \cup ||\psi|| \\
||\langle\pi\rangle\varphi|| & = & \big\{V \,:\, \exists V_1 \text{ s.t. } \langle V, V_1\rangle \in ||\pi|| \\
 & & \quad \text{and } V_1 \in ||\varphi||\big\}
\end{array}
\qquad
\begin{array}{rcl}
||\alpha|| & = & \big\{\langle V_1, V_2\rangle \,:\, V_2 = V_1 \diamond \{\alpha\}\big\} \\
||\pi;\pi'|| & = & ||\pi|| \circ ||\pi'|| \\
||\pi \cup \pi'|| & = & ||\pi|| \cup ||\pi'|| \\
||\pi^*|| & = & \big(||\pi||\big)^* \\
||\pi^{\mathsf{c}}|| & = & \big(||\pi||\big)^{-1} \\
||\varphi?|| & = & \big\{\langle V, V\rangle \,:\, V \in ||\varphi||\big\}
\end{array}
$$

Figure 1.1: Interpretation of formulas and programs.

Proposition 1.2. *Let P be a subset of $\mathbb{P}$. Suppose that $\mathbb{P}_\varphi \cap P = \emptyset$, i.e., none of the variables of P occurs in φ. Then $V \cup P \in ||\varphi||$ iff $V \setminus P \in ||\varphi||$.*

The most distinguishing feature of DL-PA though is that its dynamic operators can be eliminated (which is not possible in PDL). Just as for QBF, the resulting formula may be exponentially longer than the original formula.

Theorem 1.1 ([Balbiani et al., 2013], Theorem 1)**.** *For every* DL-PA *formula there is an equivalent boolean formula.*

For example, the DL-PA formula $\langle p \leftarrow \bot\rangle(\neg p \wedge \neg q)$ is equivalent to the formula $\langle p \leftarrow \bot\rangle\neg p \wedge \langle p \leftarrow \bot\rangle\neg q$, which is in turn equivalent to $\top \wedge \neg q$. Therefore, the DL-PA formula $\langle p \leftarrow \bot\rangle(\neg p \wedge \neg q)$ reduces to the boolean formula $\neg q$.

Every assignment sequence $\alpha_1; \ldots; \alpha_n$ is a deterministic program that is always executable: for a given V, there is exactly one V' such that $\langle V, V'\rangle \in ||\alpha_1; \ldots; \alpha_n||$. Moreover, the order of the α_i in a sequential composition is irrelevant when the set of update actions $\{\alpha_1, \ldots, \alpha_n\}$ is consistent. The following can be viewed as a reformulation of Proposition 1.1 in terms of the DL-PA operator of sequential composition.

Proposition 1.3. *Let $\{\alpha_1, \ldots, \alpha_n\}$ be a consistent set of update actions. Let $\langle k_1 \ldots k_n\rangle$ be some permutation of $\langle 1 \ldots n\rangle$. Then $V \diamond \{\alpha_1, \ldots, \alpha_n\}$ equals the single V' such that $\langle V, V'\rangle \in ||\alpha_{k_1}; \ldots; \alpha_{k_n}||$.*

This entitles us to use consistent sets of update actions as programs: one may suppose that this stands for a sequential composition in some predefined order (based e.g. on the enumeration of the set of propositional variables).

Finally, we end by noting that both satisfiability and model checking problems of DL-PA are PSPACE-complete [Balbiani et al., 2014]. We will make extensive use of DL-PA in Chapter 2 whose results are based on the definitions and contents of this Background section. Furthermore, almost all of the logics defined in the remaining chapters are influenced and inspired by DL-PA.

1.2.2 Active Integrity Constraints

Active Integrity Constraints ($\mathcal{AIC}$s) originate from the observation that, in the course of database maintenance, a designer may want to express *preferences* between the various repairs that a database may be the subject of. Therefore, $\mathcal{AIC}$s were introduced in [Flesca et al., 2004] as an extension of (static) integrity constraints with *preferred* update actions that enforce specific repairing routes when updating a database. For instance, the integrity constraint:

$$\forall(E, S_1, S_2)[Employee(E, S_1) \wedge Employee(E, S_2) \rightarrow S_1 = S_2]$$

which denotes the fact that every employee should only have one salary can be extended into the active constraint:

$$\forall(E, S_1, S_2)[Employee(E, S_1) \wedge Employee(E, S_2) \wedge S_1 > S_2 \rightarrow -Employee(E, S_1)]$$

which states that if there is an employee with two salaries then the preference is to remove the highest salary (instead of removing one randomly). In the paper the authors define *founded* repairs for the first time. The idea is that any update action applied to the database should be *supported* by the 'active part' of an $\mathcal{AIC}$, i.e., by a preferred update action of a violated constraint. They also obtain complexity results for the problem of existence of founded repairs, both in the general case (Σ_P^2-complete) as well as in the case that $\mathcal{AIC}$s comprise 'single heads', i.e., only one preferred update action is allowed in each constraint (NP-complete). Recognizing that the existence of founded repairs is not always guaranteed though, they go on to define *preferred* repairs as an intermediate repairing route between founded and standard repairs that always exist. The complexity for the problem of existence of preferred repairs is shown to be Σ_P^2-complete. Further research on $\mathcal{AIC}$s also ensued [Caroprese et al., 2006, Caroprese et al., 2009] that reviewed and

expanded upon the aforementioned results.

The first new definitions on repairing procedures that are based on preferences between update routes came in [Caroprese and Truszczynski, 2011], an attempt to relate the seemingly different approaches to $\mathcal{AIC}$s and *revision programming*. There, the authors distinguish between the various repairs that they propose (standard repairs, founded repairs, justified repairs) and their *weaker* versions (weak repairs, founded weak repairs, justified weak repairs), with only the former complying to the *minimality of change* principle that the previous approaches took by default. The definitions of justified weak repairs and justified repairs were introduced as a response to the so-called *circularity of support* defect that founded repairs cannot evade and which the authors argue against. Furthermore, they leave the first-order setting of the previous papers and use a propositional one. We will have a close and thorough look at all of these definitions in Chapter 2, which we'll examine from the perspective of DL-PA, and the propositional setting of [Caroprese and Truszczynski, 2011] will provide a valuable stepping stone to present and discuss our own approach. Another distinguishing feature in their work is that they investigate properties of *normalization*, i.e., 'breaking' all active integrity constraints into many copies such that each one has at most one preferred update action. They denote the differences that exist in these two different classes of $\mathcal{AIC}$s, both in practice (resulting in more or fewer repairs) as well as in the complexity of deciding the existence of repairs under *normal* $\mathcal{AIC}$s. The consensus on complexity results is very interesting, with all of the different kinds of repairs, being either weak or minimal, and being applied on either normal $\mathcal{AIC}$s or standard $\mathcal{AIC}$s, to fall either on the NP-complete or Σ_P^2-complete territory.

Last but not least, further approaches to refine or extend active integrity constraints have been investigated in [Cruz-Filipe et al., 2013, Cruz-Filipe, 2014] with analyses of algorithms on trees, extensions to knowledge bases outside databases, as well as independence/precedence relations among active integrity constraints.

1.2.3 Description Logic

Description Logic (DL) is an important subfield of *Logic-based Knowledge Representation* (KR) that provides delicate ways to represent and reason about knowledge through

the use of expressive but decidable families of languages [Baader et al., 2003]. These logics lie between basic propositional logic and first-order logic, providing a balance between expressiveness of the language and complexity of the various reasoning tasks. The decidable nature of DLs and the efficient algorithms that have been developed for reasoning with many of them play a vital role in plenty of applications that take advantage of their more expressive languages to deal with important real-life tasks, with an emphasis on medical ontologies [Baader et al., 2006, Rector and Rogers, 2006] and the Semantic Web [Baader et al., 2005]. Although not a new area of research, DLs enjoy an ever-increasing attention by researchers in the field of KR with new promising directions emerging and improving the state of the art continuously.

In Description Logics, a *knowledge base* KB $= (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The TBox, also called the *terminology*, contains intensional knowledge, i.e, general knowledge, in the form of (inclusion) axioms that describe general properties of the concepts and their relationship. The ABox, also called the *assertional box*, contains extensional knowledge, i.e., specific knowledge, in the form of (assertional) axioms that describe specific properties of the individuals of the domain. It is important to note early from this stage that the axioms of the TBox are intended to describe non-contingent knowledge: knowledge that is typically stable in time. The ABoxes on the other hand contain contingent knowledge that typically changes more frequently than that of the TBox. Modifications of the ABox may lead to inconsistencies with regard to TBoxes and repairing an ABox that is inconsistent with a TBox is a standard task [Lembo et al., 2010, Bienvenu et al., 2014, Bienvenu et al., 2016].

We go on to present the syntax and semantics of some of the most important and well-studied families of DL-languages, four of which will be the basis of our investigations in Chapters 3, 4 and 5. In DLs, the atomic elements from which the languages are built are the *atomic concept names*, the *atomic role names* and the *individual names*. Using these together with the *concept* and *role constructors* that each description language deploys, we can inductively build all complex concept and role descriptions available in the language. We begin with the basic description language $\mathcal{AL}$, whose concept descriptions are built inductively by the following grammar:

$$C ::= A \mid \top \mid \bot \mid \neg A \mid C \sqcap C \mid \forall r.C \mid \exists r.\top$$

where A is an atomic concept name and r is an atomic role name. The concept constructor $\forall r.C$ is called *value restriction* while the concept constructor $\exists r.\top$ is called *limited existential quantification*. Note that there are no individual names, no role constructors and that negation is only applied to atomic concept names. When it is applied to any arbitrary concept description, then the (more expressive) description language is called $\mathcal{ALC}$ (where $\mathcal{C}$ stands for *complement*). In $\mathcal{ALC}$ the *union* of concepts ($C \sqcup C$) as well as *full existential quantification* ($\exists r.C$) become expressible, since the former is an abbreviation of $\neg(\neg C \sqcap \neg C)$ while the latter is an abbreviation of $\neg\forall r.\neg C$. Further concept constructors include the *number restrictions* $\geq nr$ (at-least restriction) and $\leq nr$ (at-most restriction) for any natural number n and role r, the *one-of* constructor $\{a_1, \ldots, a_n\}$ where $a_1, \ldots, a_n$ are individual names, as well as the *fills* constructor $r : a$ where r is a role and a an individual name. The language $\mathcal{ALC}$ extended with the *one-of* constructor is called $\mathcal{ALCO}$ and concepts of the form $\{a\}$ for any individual name a are called *nominals*. Last but not least, important role constructors comprise the *role inverse*, usually written as r^- but we will denote it by r^{c} so as to keep the notation uniform with the converse operator of DL-PA throughout the chapters, as well as the *intersection* of roles $r \sqcap r$, the *union* of roles $r \sqcup r$, the *complement* of roles $\neg r$, the *composition* of roles $r \circ r$ and finally the *transitive closure* of roles r^+, where r is any atomic role name. We also mention the *universal role* r_U which we will come across and use in later chapters. The languages $\mathcal{ALC}$ and $\mathcal{ALCO}$ extended with the *role inverse* constructor are called $\mathcal{ALCI}$ and $\mathcal{ALCIO}$, respectively, whereas $\mathcal{L}(\mathcal{U})$ denotes any language $\mathcal{L}$ which also includes the universal role. For example, the concept descriptions of $\mathcal{ALCIO}(\mathcal{U})$ are built inductively by the following grammar:

$$C ::= A \mid \top \mid \bot \mid \{a\} \mid \neg C \mid C \sqcup C \mid C \sqcap C \mid \forall r.C \mid \exists r.\top$$

where A is an atomic concept name, a is an individual name and r is a (possibly inverse) atomic role name or the universal role r_U.

Having given an overview of the various concept and role constructors that are used to build complex concept and role descriptions, the next step is to present the *semantics* of each one which provides a formal description of the job each constructor is intended to do. We begin by defining the *interpretations* $\mathcal{I}$ to comprise a non-empty set $\Delta^{\mathcal{I}}$, which denotes the *domain* or *universe* of the interpretation, and an *interpretation function* $\cdot^{\mathcal{I}}$ that

maps each atomic concept A to a subset of the domain, each role r to a binary relation on the domain and each individual a to an element of the domain, i.e., $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and $\alpha^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. We say that $\mathcal{I}$ has the *unique name assumption* (UNA) if $\alpha^{\mathcal{I}} \neq b^{\mathcal{I}}$ whenever a and b are distinct individual names. The extension of $\cdot^{\mathcal{I}}$ to complex concept descriptions for the basic description language $\mathcal{AL}$ is defined inductively as follows:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{ for all } b \in \Delta^{\mathcal{I}}, \text{ if } (a,b) \in r^{\mathcal{I}} \text{ then } b \in C^{\mathcal{I}}\} \\
(\exists r.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{ there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}}\}
\end{aligned}
$$

whereas the remaining concept and role constructors have the following semantics:

$$
\begin{aligned}
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{ there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\} \\
(\geq nr)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \mathsf{card}(\{b \mid (a,b) \in r^{\mathcal{I}}\}) \geq n\} \\
(\leq nr)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \mathsf{card}(\{b \mid (a,b) \in r^{\mathcal{I}}\}) \leq n\} \\
\{a_1, \ldots, a_n\}^{\mathcal{I}} &= \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\} \\
(r : a)^{\mathcal{I}} &= \{b \in \Delta^{\mathcal{I}} \mid (b, a^{\mathcal{I}}) \in r^{\mathcal{I}}\} \\
(r^{\mathsf{c}})^{\mathcal{I}} &= \{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}}\} \\
(r \sqcap s)^{\mathcal{I}} &= r^{\mathcal{I}} \cap s^{\mathcal{I}} \\
(r \sqcup s)^{\mathcal{I}} &= r^{\mathcal{I}} \cup s^{\mathcal{I}} \\
(\neg r)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus r^{\mathcal{I}} \\
(r \circ s)^{\mathcal{I}} &= \{(a,c) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \text{ there is } b \in \Delta^{\mathcal{I}} \text{ such that} \\
&\qquad (a,b) \in r^{\mathcal{I}} \text{ and } (b,c) \in s^{\mathcal{I}}\} \\
(r^{+})^{\mathcal{I}} &= \bigcup_{i \geq 1} (r^{\mathcal{I}})^i \\
r_U^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}
\end{aligned}
$$

The TBoxes are composed of *terminological axioms* which have two forms: either $C \sqsubseteq D$ and $r \sqsubseteq s$ or $C \equiv D$ and $r \equiv s$, where C, D are concept descriptions and r, s are role descriptions of the DL language. Axioms of the first kind are called *concept* and *role inclusions* whereas axioms of the second kind are called *concept* and *role equalities*. Furthermore, equalities whose left-hand side is an atomic concept or role name are called *definitions*. For any two atomic concept names A and B, if B appears on the right-hand side of the definition of A then we say that A *directly sees* B. We also say that A *sees* B to describe the transitive closure of the relation *directly sees*. If the TBox contains an atomic concept name that sees itself then the TBox contains a *cycle*. If it doesn't contain any such atomic concept name then the TBox is *acyclic*. In the course of this thesis we will mainly use TBoxes which contain concept inclusion axioms and they can be either cyclic or acyclic. Finally, the ABoxes are constructed from a finite set of *assertional axioms* of two types: the *concept assertions* $a : C$ and the *role assertions* $r(a, b)$ where a, b are individuals, C is a concept description and r is a role description which again depend on the DL language at hand.

In Chapter 3 we use the language of the description logic $\mathcal{ALC}$ for both TBoxes and ABoxes. In Chapter 4 we stay on $\mathcal{ALC}$ for TBoxes, but we use its extension $\mathcal{ALCO}$ with nominals for ABoxes. In Chapter 5 the language of the slightly more expressive description logic $\mathcal{ALCI}$ is used for TBoxes and its extension $\mathcal{ALCIO}$ is used for ABoxes.

Moving on, an interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and, respectively, a role inclusion $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. Similarly for concept and role equalities: $\mathcal{I}$ *satisfies* $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ and $\mathcal{I}$ *satisfies* $r \equiv s$ if $r^{\mathcal{I}} = s^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* a concept assertion $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I}$ *satisfies* a role assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. An interpretation that satisfies all terminological axioms of a TBox $\mathcal{T}$ is called a *model* of $\mathcal{T}$. Similarly, an interpretation that satisfies all assertional axioms of an ABox $\mathcal{A}$ is called a *model* of $\mathcal{A}$. Last but not least, an interpretation is a model of a KB $= (\mathcal{T}, \mathcal{A})$ if it is both a model of $\mathcal{T}$ and a model of $\mathcal{A}$. If there is a model of a KB we say that the KB is *consistent*, or equivalently, that $\mathcal{A}$ is *consistent with respect to* $\mathcal{T}$.

We end the presentation on the fundamental notions of DLs by introducing the most important reasoning tasks for TBoxes and ABoxes. The *subsumption problem* is the problem of whether a concept is more (or less) general than another, i.e., given the concept

descriptions C and D, whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$. If the problem is relative to a TBox $\mathcal{T}$ then the question transforms into whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. When this is the case then we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$. The *satisfiability* problem for a concept description C is the problem of whether there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. When this is the case then we say that $\mathcal{I}$ is a *model* of C. Once again, if the problem is relative to a TBox $\mathcal{T}$ then the question transforms into whether $C^{\mathcal{I}} \neq \emptyset$ for a model $\mathcal{I}$ of $\mathcal{T}$. Another reasoning task is the *instance checking* problem, which is the problem of whether an assertion *follows* from a specific ABox $\mathcal{A}$, i.e., given the concept description C, the role description r and the individuals a, b, whether $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{A}$. When this is the case then we write $\mathcal{A} \models a : C$ and $\mathcal{A} \models r(a, b)$, respectively. It is worth mentioning that most reasoning tasks in DLs are *reducible* to one another.

Apart from the syntax, semantics and reasoning tasks of the various DL languages, we now review some of the most widespread methods to repair inconsistencies in DL knowledge bases. As we will see in later chapters, our approach to the problem of repairing inconsistent KBs is by updating ABox assertions in order to comply with (preferred update actions indicated by) the TBox axioms, which is a research avenue that has not been directly pursued in the DL literature. The related (and most prominent) approaches to repairing KBs are mainly based on the so-called *justifications*, which are minimal subsets of the KB containing the terminological and assertional axioms from which an undesirable consequence is inferred. Axiom pinpointing through justifications became an important topic of research within the DL community and several results were quickly established [Schlobach and Cornet, 2003, Kalyanpur et al., 2007, Suntisrivaraporn et al., 2008]. Both black-box [Schlobach et al., 2007, Baader and Suntisrivaraporn, 2008] and glass-box [Parsia et al., 2005, Meyer et al., 2006] methods emerged for computing justifications. The former have a more universal approach and are used independently of the reasoner at hand, while the latter have a more delicate construction that is tied to specific reasoners and usually require less calls. After computing all justifications of an undesirable consequence, the next step is to obtain a minimal *hitting set* [Reiter, 1987] made up of one axiom per justification and remove it from the knowledge base. More recent approaches though have focused on providing methods for *weakening* the axioms instead of removing them, since the latter can prove to be too big of a change [Troquard et al.,

2018].

On another front, since we will be using dynamic procedures to obtain results in the DL setting, it is worth recalling the interplay between PDL (and various extensions) with Description Logics admitting regular expressions over roles. The mapping that established a translation from one to the other helped both disciplines in obtaining decidability and complexity results that were until then non-existent. For an overview on the matter see the handbook [Baader et al., 2003]. More recent related work in reducing dynamic problems to static DL reasoning includes [Ahmetaj et al., 2017] which also makes use of integrity constraints in rich DLs. Speaking of integrity constraints, we end by mentioning that approaches to encode integrity constraints into TBoxes already exist, with the most prominent ones being based on extending the KBs with *constraint axioms*. These axioms however are used for validation purposes only and do not share the same semantics as regular TBox axioms [Motik et al., 2009, Tao et al., 2010].

1.3 Contributions and Outline of the Thesis

As we have mentioned before, the main goal in this thesis is to apply the idea behind active integrity constraints in the Description Logic setting and, in particular, to provide extensions of TBox axioms so that they are able to suggest *preferred* repairing routes in case of inconsistencies. Before attempting this though, we begin in Chapter 2 with an overview of static and active integrity constraints, investigated through DL-PA in the propositional setting. We give the details of *weak*, *founded* and *justified* repairs that we skipped in Section 1.2.2 and present embeddings of each one into DL-PA. The most important contribution of the chapter though is the definition of *dynamic repairs* which constitute a new, dynamic way of integrity maintenance and which was recently proposed in [Feuillade and Herzig, 2014]. After an analysis of their properties and a comparison to the other established repairs of the literature, we provide complexity results for the problem of existence of these new repairs. We then take advantage of the dynamic framework that we use (the logic DL-PA) in order to explore an extension on databases with history and adjust the behavior of the various repairs so that they work in this setting. Finally, for all these definitions we provide DL-PA counterparts of reasoning and decision problems,

such as the existence of a repair or the existence of a unique repair. Chapter 2 is based on [Feuillade et al., 2019].

Lifting the idea behind $\mathcal{ALC}$s to Description Logics starts in Chapter 3, where the first definition of '*active*' TBoxes is introduced. After a brief discussion on the differences and difficulties of leaving the propositional setting, we examine preliminary steps into repairing ABoxes *syntactically* so that they conform to the preferences denoted by the active axioms. These syntactic repairs are inspired by (and correspond to) the weak and founded repairs of the database literature. Proving to be quite impractical though, we suggest that a *semantic* approach seems more viable and venture into tackling the limitations of such a syntactic approach in the subsequent chapters. Chapter 3 is based on [Rantsoudis et al., 2017].

In the following two chapters we go on to pursue this semantic approach and, more specifically, investigate how the dynamic logic-based framework and methods used in Chapter 2 behave in the DL level. Chapter 4 uses a more *local* approach, where preferred update actions in the active axioms behave similarly to the update actions introduced before, i.e., they have the form $\pm A$ for an *atomic* concept A denoting either the *addition* or the *removal* of an individual from the set of individuals that have property A. This approach, although more expressive and well-behaved than the syntactic one of Chapter 3, still leaves a lot of repairing scenarios unattainable, mainly because of its boolean nature and close similarity to the repairs of the database literature. On the other hand, Chapter 5 introduces and discusses a more elaborate logic and techniques which apply changes *globally*, in the sense that preferred update actions of the form $\pm C$ can be applied to *all* individuals violating an axiom and C is not necessarily atomic. Again, both of these semantic approaches use a *dynamic logic* framework, influenced by the logic DL-PA already showcased in Chapter 2. Furthermore, although the resulting logics are extensions of $\mathcal{ALCO}$ and $\mathcal{ALCIO}$ respectively with dynamic operators, they are shown to be as expressive as their static counterparts (with the addition of the universal role). The results on Chapter 4 are based on [Feuillade et al., 2018], whereas the results on Chapter 5 have not been published at the time of writing.

The thesis concludes in Chapter 6 with a summary of the work presented and a proposal to connect the approaches of the (syntactic) Chapter 3 and the (semantic) Chapter 5,

thus completing the picture on active TBox-based ABox repairs. We also briefly discuss future work, with possible applications of the proposed repairing methods on nonmonotonic scenarios in an attempt to combine the results of this thesis with different areas of research that I've also been involved in (see the next section).

1.4 Other Contributions

In this section I briefly discuss research that I've conducted in parallel to what is reported in this thesis. The main theme of several research areas and applications, as well as of the work presented here, revolves around the search for *consistency*. Although we generally want consistency, it is true that in real life we tend to live with inconsistencies in the sense that a big part of the conclusions we draw is *defeasible*. The second line of work I was involved in deals with notions of *defeasibility* since one of my main interests lies on *nonmonotonic reasoning* and in particular on *conditional logics of normality*. Before delving into any details though, a general description of this research filed immediately follows.

Nonmonotonic Logics. One of the major goals in AI is to develop tools that bridge the gap between human *commonsense reasoning* and artificial reasoning of *rational agents* (like computers or robots). In this regard, *nonmonotonic* ways to reason about knowledge are extensively studied and developed since their connection with commonsense reasoning is apparent: it is true for a human agent that a seemingly plausible conclusion can be later retracted in the light of new information (*defeasible inferences*) and in most real-life scenarios reasoning is conducted with only incomplete information. Classical (Mathematical) Logic cannot deal with these situations since it is inherently monotonic. Thus, *nonmonotonic logics* (like *autoepistemic* or *conditional* logics) were devised to capture *default reasoning* in the way described above, extending classical logic and providing a formal mathematical framework through which knowledge can be represented and inferred using nonmonotonic procedures. An exciting and mature research field, it always remains relevant in the era that AI flourishes and its connection to emerging areas of research (like machine learning) looks promising and essential.

Conditional Logic is primarily concerned with the logical and semantic analysis of the rich class of *conditional statements*, identified with the sentences conforming with the '*if* A *then* B' structure. The topic has roots in antiquity and the medieval times but its contemporary development seems to start with F. Ramsey in the '30s and has blossomed after the late '60s [Arlo-Costa, 2014]. There exist various conditionals of interest in Philosophy, Logic, Computer Science and Artificial Intelligence, including *counterfactual conditionals, open conditionals, causal conditionals, deontic conditionals, normality conditionals* (see [Crocco et al., 1996] for a broad overview of applications); they represent different linguistic constructions with a common structural form ('*if ... then*') and the aim of the field is to provide a unifying formal logical account that accurately captures their essential meaning.

Conditional Logics of Normality. Artificial Intelligence has been interested in *conditional logics* for *default reasoning* already from the '80s (see the work of J. Delgrande [Delgrande, 1987, Delgrande, 1988]), in *counterfactual conditionals* (M. Ginsberg, [Ginsberg, 1986]) and in the '*normality conditionals*' in nonmonotonic reasoning [Bell, 1990, Lamarre, 1991, Boutilier, 1992]. The reader is referred to the handbook article of J. Delgrande [Delgrande, 1998] for a broad overview of conditional logics for defeasible reasoning. The investigations on the intimate relation of conditional logics to nonmonotonic reasoning have been further triggered by the seminal work of S. Kraus, D. Lehmann and M. Magidor [Kraus et al., 1990, Lehmann and Magidor, 1992], whose framework (KLM) has become the 'industry standard' for nonmonotonic consequence relations. There exist various possible-worlds semantics for conditional logics (see [Nute, 1980, Delgrande, 1998]) and a connection to modal logic (known from D. Lewis' work [Lewis, 1973]) which has been further explored by the modal construction of '*normality conditionals*' [Lamarre, 1991, Boutilier, 1992].

A logic of '*normality conditionals*' for default reasoning attempts to pin down the principles governing the statements of the form '*if* A*, then normally* B *is the case*'. '*Normally*' is susceptible to a variety of interpretations. One is based on a '*normality*' ordering between possible worlds: $(A \Rightarrow B)$ is true if it happens that in the most '*normal*' (least *exceptional*) A-worlds, B is also true [Lamarre, 1991, Boutilier, 1992]. Another, more recent one [Jauregui, 2008] interprets '*normally*' as a '*majority*' quantifier: $(A \Rightarrow B)$ is

true iff B is true in '*most*' A-worlds. Questions of '*size*' in preferential nonmonotonic reasoning have been firstly introduced by K. Schlechta [Schlechta, 1995, Schlechta, 1997]; the notion of '*weak filter*' that emerged (as a '*core*' definition of a collection of '*large*' subsets) has been also employed in modal epistemic logics [Askounis et al., 2016].

A majority-based account of default conditionals depends heavily on what counts as a '*majority*' of alternative situations, what is a '*large*' set of possible worlds. It is difficult to state a good definition that would work for both the finite and the infinite case; the notions of *weak filters* and *weak ultrafilters* that have been used capture the minimum requirements of such a notion [Schlechta, 1997, Jauregui, 2008]. In [Koutras and Rantsoudis, 2015, Koutras and Rantsoudis, 2017], we experimented with a notion of '*overwhelming majority*', combined with the widely accepted intuition that $(A \Rightarrow B)$ should mean that $(A \wedge B)$ is more plausible than $(A \wedge \neg B)$. We defined conditionals of this form to (essentially) mean that $(A \wedge B)$ is true in '*almost all*' (in the mathematical sense, i.e., in '*all but finitely many*') points in the countable modal frame $(\omega, <)$ (the first infinite ordinal, strictly ordered under $<$), whose modal axiomatization (the normal modal logic $\mathbf{K4DLZ}$) is known as the 'future' fragment of the temporal logic of discrete linear time [Goldblatt, 1987, Segerberg, 1970]. This majority conditional is modally defined and this readily provides a decision procedure, as a modal translation of conditional formulas can be checked for validity in $(\omega, <)$ using any of the proof procedures known for $\mathbf{K4DLZ}$. We examined the properties of this conditional, in particular with respect to the (conditional incarnation of the) '*conservative core*' of defeasible reasoning set by the KLM framework. The paradigm of '*overwhelming majority*' in our work is consistently represented with cofinite subsets of ω. En route, we discuss variants: trying *cofinal* (rather than *cofinite*) subsets of ω, and/or varying the modal definition of the conditional connective. Then, we discuss the possibility of defining conditionals over cofinite subsets of ω in the neighborhood semantics for conditional logics; we prove that the conditionals defined can be very weak, even compared to the conditionals introduced in [Delgrande, 2006].

Our main goal in these papers was to identify a set of '*normality*' principles that would possibly characterize a majority-based account of default conditionals, given that the '*cofinite vs. finite*' intuition is undeniably a widely acceptable case in the '*big vs.*

small' question, and to delineate the limits of such an approach. In addition, we decided to exploit the well-known (and well-behaved) machinery of modal logic which allowed us to obtain definitions of cofinite sets of possible worlds in $(\omega, <)$, resulting in logics with an NP-complete satisfiability problem. We have also experimented with logics defined on appropriately populated Scott-Montague frames obtaining relatively weak logics, something not entirely unexpected in this area (see [Delgrande, 2006] for weak conditionals constructed under a rule-based interpretation of defaults). A more fine-grained and elaborate approach came in [Koutras et al., 2018] by exploiting tools from Mathematical Analysis and Topology to investigate, model-theoretically, this time more delicate and refined ways to obtain size-oriented approaches to normality statements. A 'spiritual successor' of the previous works on 'overwhelming majority' interpretations for defaults, the 'most' generalised quantifiers this time provide better readings of 'normality' than the previous finite-cofinite intuition. The logics are obtained through the notions of *clear majority* on finite sets of worlds, *asymptotic density* of integer sequences, *dense* (and *nowhere dense*) sets of topological spaces and *measure* on measure spaces, examined and compared against the 'conservative core' of nonmonotonic reasoning.

Apart from the size-oriented approaches to normality, an area where I have contributed is on defining conditional logics of normality directly within Epistemic Logic. It is true that default statements admit various readings and a fundamental one corresponds to their principal use of ascribing default properties to individuals ('*Tweety flies since birds normally fly*'), a function accomplished elegantly also in McCarthy's *Circumscription* (via classical first-order logic) and Reiter's *Default Logic* (via the rules of inference adjoined to first-order logic). Other readings of defeasible conditionals seem closer to statements about (mostly qualitative but also quantitative) probability: '*birds generally* (*typically, mostly*) *fly*'. It has been noticed however that "*the reading* 'a bird that can be consistently assumed to fly does fly' *is clearly epistemic in nature*" [Delgrande, 2012, p. 95]. The reader should consult the recent paper [Delgrande, 2012] for a deep technical and philosophical discussion on the content, the nature and the role of default statements in Commonsense Reasoning and the pros and cons of using conditional logic to capture their content.

That the defeasible conditionals (or at least, some of them) clearly have an epistemic

interpretation, has been noticed early enough, implicitly or explicitly. A 'normality statement' of the form '*every Tuesday afternoon, you can find Jimmy taking a beer in the corner pub*' allows one to infer that on a 'regular' Tuesday s/he can meet Jimmy there and this default inference involves facts known ('it is a Tuesday'), facts observed and believed ('Jimmy frequents this place on Tuesday afternoon'), facts considered to be consistent with the belief base ('there is no reason to believe this is an 'irregular' Tuesday') and facts plausibly inferred ('most probably I will meet him there'). Although it is difficult to agree on the subtle details of the epistemic attitudes involved, it seems that there is an agreement on the fact that such a description is quite reasonable. Such cognitive statements are implicit in Reiter's normal defaults [Reiter, 1980] and the conditional entailment of H. Geffner and J. Pearl [Geffner and Pearl, 1992]: '*a rule* $\frac{a : b}{b}$ *may be seen as a soft constraint for believing* b *when* a *is known, while a conditional rule* $a \Rightarrow b$ *can be viewed as a hard constraint to believe b in a limited context defined by a and possibly some background knowledge*' [Eiter and Lukasiewicz, 2000, p. 220].

The study of the connection of defeasible (and perhaps other sorts of) conditionals with the area of Epistemic and Doxastic Logic has not been hitherto pursued in its full entirety. In general, the '*conditionals-via-modal-logic*' technique is known and quite successful [Lewis, 1973, Chellas, 1975]; yet, the technical and philosophical step to the construction of conditionals via Epistemic Logic—or the integration of epistemic logic and conditional logic—has not been fully taken. The relation of Epistemic Logic to conditionals mainly revolves around the famous *Ramsey test* and this is also apparent in the earlier works of P. Lamarre & Y. Shoham [Lamarre and Shoham, 1994] or N. Friedman & J. Halpern [Friedman and Halpern, 1997] where an interesting notion of *conditional belief* is based on the semantics of default conditionals (see also [Aucher, 2014, p. 107]). Modal approaches to defeasible conditionals [Lamarre, 1991, Boutilier, 1992, Boutilier, 1994, Delgrande, 2006] are mostly based on the model-theoretic intuition of 'preference' among possible worlds or propositions (overall, the '*preferential approach*' to nonmonotonic logics and to logics of 'typicality' has been very influential, see [Britz et al., 2011] for a recent application). The conditional $\varphi \Rightarrow \psi$ is true in a possible world if ψ is true in the most 'normal' or 'preferred' accessible φ-worlds; equivalently, given the context of φ, the proposition expressed by $\varphi \wedge \psi$ is preferred over the one expressed by $\varphi \wedge \neg\psi$ [Delgrande, 2006]. It is natural to consider that normality orderings are preorders (reflexive

and transitive relations) and thus the modal approaches to defeasible conditionals usually employ the logic $\mathbf{S4}$ (or its extension $\mathbf{S4.3}$) within which the defeasible conditional is modally defined [Boutilier, 1992].

In [Koutras et al., 2017a,Koutras et al., 2019], we amplify the epistemic interpretation of defeasible conditionals and proceed to define them directly within Epistemic Logic. We work inside $\mathbf{KBE}$, a recently introduced epistemic logic [Koutras et al., 2017b] accounting for *knowledge, belief* and *estimation* (as a form of weak, complete belief, interpreted as '*truth in most epistemic alternatives*'). $\mathbf{KBE}$ comprises a $\mathbf{S4.2}$ framework for knowledge and belief, following the fundamental investigations of W. Lenzen [Lenzen, 1978] who advocated it as the 'correct' logic of knowledge; note that R. Stalnaker arrived at a similar proposal via different assumptions [Stalnaker, 2006]. The non-normal modal operator for *estimation* is interpreted as a majority quantifier over the set of epistemic alternatives of a given possible world. The formal apparatus is that of a *weak ultrafilter*, which is an upwards-closed collection of sets, with pairwise non-disjoint members and such that exactly one out of a set and its complement occurs in the collection; the notion extends the *weak filters* introduced in [Schlechta, 1997] and later, independently, in [Jauregui, 2008]. We define three nonmonotonic conditionals by capturing a size-oriented version of the fundamental intuition of normality conditionals: $\varphi \Rightarrow \psi$ is set to mean that $\varphi \wedge \psi$ is more normal compared to $\varphi \wedge \neg\psi$, as it holds in 'most' epistemically alternative worlds; this is achieved by exploiting the nature of $\mathbf{KBE}$'s 'estimation' operator as a majority quantifier. The logics emerging are rather weak compared to the 'conservative core' of default reasoning (the system $\mathbf{P}$, [Kraus et al., 1990]) but this is neither surprising nor discouraging: weak conditionals of this kind have been also introduced in [Delgrande, 2006, system $\mathbf{C}$ and system $\mathbf{\Lambda}$] under a rule-based interpretation of defaults and it is well-known that conditionals based on the plausibility structures of N. Friedman & J. Halpern do not generally satisfy all the KLM properties [Friedman and Halpern, 1997, p. 266]. Another, very 'natural' (but rather strong in epistemic assumptions) translation leads to a weak monotonic conditional, and two other epistemic definitions give rise to nonmonotonic conditional logics which do not satisfy the axiom $\mathbf{ID}$ (*reflexivity*), but they capture very interesting conditional principles and one of them comes close to the 'overwhelming majority' conditional defined in [Koutras and Rantsoudis, 2017]. Note that for all these definitions a recursive translation in the language of $\mathbf{KBE}$ provides direct access to the

tableaux proof procedure for this logic [Koutras et al., 2017b], and thus a machinery for testing theoremhood is readily available. It should also be noted that analytic tableaux exist for KLM logics and the tableaux procedure is constructed via a modal encoding of nonmonotonic conditionals [Giordano et al., 2009].

There exist two main directions in employing Conditional Logic in Knowledge Representation. The first - and perhaps more influential one - asks for devising mechanisms which will reveal the contingent conclusions that can be plausibly extracted given a background conditional (default) theory. The second responds to the objective of introducing axiomatic ways to account for a precise notion of default conditional and devise theories that capture the basic properties of a defeasible conditional, in as much the same way it has been achieved for indicative or counterfactual conditionals; see J. Delgrande's handbook article [Delgrande, 1998] for further analysis. In these papers, we took the second direction. We focused on the 'epistemic connection' of defeasible conditionals and investigated the possibility of a direct syntactic definition within Epistemic Logic; the intension was to check the potential of such an approach, whose rewarding benefits are more than obvious.

We will come back to the realm of nonmonotonic reasoning in Chapter 6, where we briefly mention possible future connections between the results reported in this thesis and defeasible inferences.

CHAPTER 2

A Dynamic Logic Account of Active Integrity Constraints

Contents

2.1 Introduction

As we have seen in Section 1.2.2, the setting of active integrity constraints is well established and thoroughly explored. Despite this however, there exist cases where even founded and justified repairs cannot provide a satisfactory solution to the problem of repairing a database under a set of $\mathcal{AIC}$s. A more *dynamic* procedure could provide solutions to inconsistencies that arise between a database and a set of active constraints that build upon and extend one another. We showcase such an example in the following

(propositional) scenario. Consider a company with two departments, D_1 and D_2, where temporary employees (e.g. interns) are assigned to D_1 and permanent employees work at D_2 (so D_1 has no permanent members and D_2 has no temporary members). Every employee must be in a department, i.e., we have the integrity constraint $D_1 \vee D_2$. Furthermore, consider that every person working on D_2 has previously worked on D_1, i.e., D_1 is like a 'training ground' for becoming a permanent member. Consider now a database for permanent members which keeps track of their status. Every employee should be declared in the database as starting to work on D_1 and we can express this by the active constraint $\langle D_1 \vee D_2, +D_1 \rangle$ which declares that if $D_1 \vee D_2$ is violated and an employee is not assigned to any department then s/he should be assigned to D_1. Furthermore, since the database is for permanent employees, if an employee is assigned only to D_1 then this should be rectified by assigning him/her to D_2 as well. This is expressed by the active constraint $\langle \neg D_1 \vee D_2, +D_2 \rangle$. Finally, consider that the database loses track of an employee, i.e., a permanent employee is declared as working at neither D_1 nor D_2. How would the status of this employee, which is inconsistent w.r.t. the active constraints, be repaired according to the available repair procedures? Whereas founded and justified repairs cannot provide a solution to this problem (see Section 2.2.2 for more details), a dynamic procedure which would check each active constraint at a time and apply an update action before repeating seems able to do so. Indeed, as we will witness in Section 2.5.2, the set $\{+D_1, +D_2\}$ will be the only solution using such a dynamic procedure. Note also that a repair which conforms to the minimal change principle would suggest that $\{+D_2\}$ should be the only repair. While this is indeed the only minimal solution, it can be argued that it should not be the case that this employee was assigned to D_2 directly, without having worked on D_1 first.

Just as in [Caroprese and Truszczynski, 2011, Cruz-Filipe, 2014] we only consider ground constraints in the current chapter, i.e., we work with a propositional language. Due to this, static constraints will be represented by boolean formulas and, as we have mentioned in Section 1.1, an active integrity constraint will be a couple $r = \langle \mathsf{C}(r), \mathsf{R}(r) \rangle$ where $\mathsf{C}(r)$ is the static constraint and $\mathsf{R}(r)$ is the set of preferred update actions. In this chapter we examine active integrity constraints in the framework of dynamic logic and argue that they can be viewed as particular programs: the sequential composition of the test of $\neg\mathsf{C}(r)$ and the nondeterministic choice of an action in $\mathsf{R}(r)$. Repairing a database

can then be done by means of a complex program that combines active integrity constraints. We use DL-PA, the simple yet powerful dialect of dynamic logic we introduced in Section 1.2.1. We recall that, instead of PDL's abstract atomic programs, the atomic programs of DL-PA are update actions: assignments of propositional variables to either true or false, written $p \leftarrow \top$ and $p \leftarrow \bot$. Just as in PDL, these atomic programs can be combined by means of program operators: sequential and nondeterministic composition, finite iteration and test. The language of DL-PA has not only programs, but also formulas. While DL-PA programs describe the evolution of the world, DL-PA formulas describe the state of the world. In particular, formulas of the form $\langle \pi \rangle \varphi$ express that φ is true after *some* possible execution of π, and $[\pi]\varphi$ expresses that φ is true after *every* possible execution of π. The models of DL-PA are considerably simpler than PDL's Kripke models: valuations of classical propositional logic are enough. The assignment $p \leftarrow \top$ inserts p, while the assignment $p \leftarrow \bot$ deletes p. Apart from being simple yet quite expressive, its biggest computational advantage over PDL comes in the form of the elimination of the Kleene star: it is shown in [Herzig et al., 2011, Balbiani et al., 2013] that every DL-PA formula can be reduced to an equivalent boolean formula (something that is not possible in PDL). This is an important attribute and a very useful tool, as it will allow us to construct repaired databases syntactically. The most significant advantage of using a dynamic logic framework though is the fact that we can easily study extensions (like the history-based repairs of Section 2.7) that are expressible in the language by simply extending the formulas in the appropriate ways.

The chapter is organized as follows. In Section 2.2 we provide a thorough background on static and active constraints, as well as the associated repairs for both (weak repairs, PMA repairs, founded and justified repairs). In Section 2.3 we provide an embedding of the associated repairs of static constraints (weak repairs and PMA repairs) into DL-PA. In Section 2.4 we do the same for the associated repairs of active constraints (founded and justified repairs). Section 2.5 comprises the main contribution of the chapter: we propose some new definitions of repairs in terms of **while** programs and compare them with the aforementioned founded and justified repairs in various aspects. Their computational complexity is investigated in Section 2.6. In Section 2.7 we push the envelope of the active constraint paradigm and examine databases with history as well as how the various repairs are integrated in their framework. Finally, Section 2.8 concludes with some examples of

related reasoning problems and a brief discussion on future work.

This chapter extends [Feuillade and Herzig, 2014] by the analysis of justified repairs, a thorough discussion on the dynamic repairs introduced as well as how they compare with the available repair procedures from the literature, a complexity analysis and a look into databases with history.

2.2 Background

The basic definitions and properties of DL-PA have already been introduced in Section 1.2.1. We only note that:

(1) In the context of the present chapter a valuation is called a *database*.

(2) We sometimes use X as a metavariable for $\top$ and $\bot$ and write $p \leftarrow \mathsf{X}$.

(3) For subsets P of $\mathbb{P}$ it will be convenient to write $P \leftarrow \top$ to denote the set of update actions $\{p \leftarrow \top \ : \ p \in P\}$, and likewise for $P \leftarrow \bot$.

In the following subsections, we recall the definitions of the various repair procedures w.r.t. static and active integrity constraints.

2.2.1 Static Constraints and the Associated Repairs

In this subsection we consider the classical notion of database integrity that is defined in terms of *static integrity constraints*, or *static constraints* for short. In our propositional language they are nothing but boolean formulas. Two ways of repairing databases based on such constraints can be found in the literature [Caroprese and Truszczynski, 2011]. Both consist in first finding an appropriate set of update actions U and then building the update $V \diamond U$ of V by U as defined in Section 1.2.1. We relate them to well-known operations in belief revision and update [Katsuno and Mendelzon, 1992], which allows us to reuse their embeddings into DL-PA [Herzig, 2014].

Weak Repairs and Drastic Updates

Let V be a database, U a set of update actions and C a set of static constraints, i.e., a set of boolean formulas. In the rest of the chapter we will only consider finite sets of static constraints. We say that U is *relevant* w.r.t. V iff $p \leftarrow \top \in U$ implies $p \notin V$ and $p \leftarrow \bot \in U$ implies $p \in V$. The definition of a *weak repair* immediately follows.

Definition 2.1. *Let V be a database and let C be a set of static constraints. A weak repair of V achieving C is a consistent set of update actions $U \subseteq \mathbb{U}$ such that $V \diamond U \models \bigwedge C$ and such that U is relevant w.r.t. V.*

The next example illustrates that weak repairs are indeed very weak.

Example 2.1. *Let $V = \emptyset$ and $C = \{p \vee q\}$. The weak repairs of V achieving C are all those subsets of the set of positive update actions $\{r \leftarrow \top \ : \ r \in \mathbb{P}\}$ that contain either $p \leftarrow \top$, or $q \leftarrow \top$, or both.*

As the following result shows, if we consider what is true in all possible weak repairs then we obtain what is called a *drastic update* in the literature on belief revision and update.[1]

Proposition 2.1. *Let V be a database and let C be a set of static constraints. Then:*

$$\Big\{ V \diamond U \ : \ U \text{ is a weak repair of } V \text{ achieving } C \Big\} = \Big|\Big| \bigwedge C \Big|\Big|$$

Note that a weak repair may contain assignments of variables that do not occur in C. To witness, in the above example $\{p \leftarrow \top, q \leftarrow \top, r \leftarrow \top\}$ is a weak repair of V achieving C. To remedy this we consider every weak repair U from now on to be such that if $p \leftarrow \top$ or $p \leftarrow \bot$ occurs in U then $p \in \mathbb{P}_C$. This corresponds to a very basic update semantics that is sometimes called Winslett's standard semantics [Winslett, 1990, Herzig and Rifi, 1999].

[1] It is actually also a drastic revision because V is a complete database and update and revision coincide in that case [Peppas et al., 1996].

Repairs *Tout Court* and their Relation to Winslett's PMA

We now present the definition of a *repair* which, as we already mentioned, uses the principle of minimal change to produce repair solutions that are considered more practical in contrast to the more general weak repairs.

Definition 2.2. *Let V be a database and let C be a set of static constraints. A repair of V achieving C is a weak repair of V achieving C that is minimal w.r.t. set inclusion: there is no weak repair of V achieving C that is strictly contained in it.*

The next example is a follow-up to Example 2.1.

Example 2.2. *Let $V = \emptyset$ and $C = \{p \vee q\}$. There are exactly two repairs of V achieving C, viz. $\{p \leftarrow \top\}$ and $\{q \leftarrow \top\}$.*

We are now going to relate repairs to Winslett's possible models approach PMA [Winslett, 1988, Winslett, 1990]. Remember that the update of a database V by a boolean formula A according to the PMA is the set of valuations V' such that $V' \models A$ and such that the symmetric difference between V and V' is minimal w.r.t. set inclusion. Formally, the symmetric difference is defined as $\mathsf{D}(V, V') = \{p \ : \ V(p) \neq V'(p)\}$ and the PMA update of V by A is the set:

$$\Big\{ V' \ : \ V' \models A \text{ and there is no } V'' \in ||A|| \text{ such that } \mathsf{D}(V, V'') \subset \mathsf{D}(V, V') \Big\}$$

For example, the PMA update of $\emptyset$ by $p \vee q$ is $\Big\{\{p\}, \{q\}\Big\}$ and the PMA update of $\emptyset$ by $(p \wedge q) \vee r$ is $\Big\{\{p, q\}, \{r\}\Big\}$.

Proposition 2.2. *Let V be a database and let C be a set of static constraints. Then:*

$$\Big\{ V \diamond U \ : \ U \textit{ is a repair of } V \textit{ by } C \Big\} \textit{ is the PMA update of } V \textit{ by } \Big(\bigwedge C \Big)$$

The above result justifies the term *PMA repair* that we are going to employ henceforth (because the mere term 'repair' might lead to confusions).

2.2.2 Active Constraints and the Associated Repairs

Active integrity constraints were proposed more than ten years ago [Flesca et al., 2004] and various ways of repairing a database V by such constraints were studied in the literature. Just as for static constraints, all definitions are based on the notion of a *repair set*: an appropriate set of update actions U such that $V \diamond U$ no longer violates the integrity constraints. $V \diamond U$ is once again the result of updating V by U as defined in Section 1.2.1 and is called the *repaired database*.

In the present subsection we recall syntax and semantics of the two main routes that have been explored in the literature.

Active Integrity Constraints

An *active integrity constraint* (or *active constraint* for short) combines a static integrity constraint with a set of preferred repair actions.

Definition 2.3. *An active constraint is a couple* $r = \big\langle \mathsf{C}(r), \mathsf{R}(r) \big\rangle$*, where* $\mathsf{C}(r)$ *is a boolean formula and* $\mathsf{R}(r)$ *is a finite set of update actions that is consistent.*

As before, $\mathsf{C}(r)$ is a static integrity constraint that is violated when $\mathsf{C}(r)$ is false. If this is the case and $\mathsf{R}(r) \neq \emptyset$ then r is *applicable* and $\mathsf{R}(r)$ indicates how to get rid of the violation and restore integrity. The elements of $\mathsf{R}(r)$ are viewed as *permitted* update actions: when $\mathsf{C}(r)$ is violated then each of the actions in $\mathsf{R}(r)$ gets a 'license to update'. This is a rather imprecise description of the job the update actions in $\mathsf{R}(r)$ are expected to do and in the literature various semantics were proposed. One of the most prominent of them are *founded* repairs which make use of the *foundedness* condition in order to apply the correct update actions, while *justified* repairs build upon and refine this condition in order to tackle the so-called *circularity of support* issue that can be witnessed by founded repairs.

We say that an active constraint $r = \langle \mathsf{C}(r), \mathsf{R}(r) \rangle$ is *standard* if $\mathsf{C}(r)$ is a clause and each update action in $\mathsf{R}(r)$ makes one of the literals of $\mathsf{C}(r)$ true: if $p \leftarrow \top \in \mathsf{R}(r)$ then p has to be one of the literals of $\mathsf{C}(r)$ and if $p \leftarrow \bot \in \mathsf{R}(r)$ then $\neg p$ has to be one of the

literals of $\mathsf{C}(r)$.

Remark 1. *The definition in the literature differs in several respects from ours here. First,* $\mathsf{C}(r)$ *is usually not viewed as a static integrity constraint but as the negation of a static integrity constraint:* r *is violated when the first argument of* r *is true. Second, active constraints are noted* $\mathsf{C}(r) \rightarrow \mathsf{R}(r)$*, which makes them look like formulas. However, "*$\rightarrow$*" is different from material implication as the right hand side of the implication is not a formula but a set of programs. So their semantics remains to be given: in the literature this is typically done by means of disjunctive logic programs under a non-monotonic semantics. Third, all active constraints have to be standard.*

We denote finite sets of active constraints by η, η_1, etc. The set of static integrity constraints associated with η is defined as $\mathsf{C}(\eta) = \{\mathsf{C}(r) \ : \ r \in \eta\}$. Furthermore, the *size* of $\mathsf{C}(\eta)$, denoted by $|\, \mathsf{C}(\eta)\, |$, is the sum of the lengths of each $\mathsf{C}(r)$ for all $r \in \eta$, i.e., $|\, \mathsf{C}(\eta)\, | = \sum_{r \in \eta} |\, \mathsf{C}(r)\, |$, where $|\, \mathsf{C}(r)\, |$ is the length of the boolean formula $\mathsf{C}(r)$ as defined in propositional logic.

It remains to give a semantics to active constraints. In the rest of this subsection we discuss the two main existing directions, viz. founded and justified repairs. We later propose a new one in Section 2.5 using the programs of DL-PA.

Founded Weak Repairs and Founded Repairs

In the literature, founded repairs are considered to be a basic semantics of active constraints. They provide a basis for further refinements. The key notion they rely on is the *foundedness* condition.[2]

Definition 2.4. *Let* V *be a database and let* η *be a set of active constraints. A consistent set of update actions* U *is founded if for every* $\alpha \in U$ *there is an* $r \in \eta$ *such that:*

(a) $\alpha \in \mathsf{R}(r)$

(b) $V \diamond U \models \mathsf{C}(r)$

[2]We have reformulated the original definition so that it applies to our more general definition of active constraints. Both are equivalent as far as standard active constraints are concerned.

(c) $V \diamond (U \setminus \{\alpha\}) \not\models \mathsf{C}(r)$

Given this condition, the definitions of a *founded weak repair* and a *founded repair* immediately follow.

Definition 2.5. *Let V be a database and let η be a set of active constraints. A set of update actions U is a founded weak repair of V by η if U is a weak repair of V achieving $\mathsf{C}(\eta)$ and U is founded. Moreover, if U is also a PMA repair of V achieving $\mathsf{C}(\eta)$, then U is a founded repair of V by η.*

The following simple example showcases this definition.

Example 2.3. *Let $V = \emptyset$ and $\eta = \Big\{\langle p, \{p \leftarrow \top\}\rangle, \langle p \vee q, \{q \leftarrow \top\}\rangle\Big\}$. The set $\{p \leftarrow \top\}$ is the only founded weak repair of V by η. Indeed, the second update action in $\{p \leftarrow \top, q \leftarrow \top\}$ cannot be founded on the second active constraint of η. It is also the only founded repair.*

There are sets of active constraints for which there is no founded repair, although there is a founded weak repair [Caroprese and Truszczynski, 2011, Example 2]. The next example, which is adapted from the example of the introductory Section 2.1, shows that there are sets of active constraints for which there is not even a founded weak repair.

Example 2.4. *Let $V = \emptyset$ and $\eta = \Big\{\langle p \vee q, \{p \leftarrow \top\}\rangle, \langle \neg p \vee q, \{q \leftarrow \top\}\rangle\Big\}$. The set $\{q \leftarrow \top\}$ is a PMA repair of V achieving $\mathsf{C}(\eta)$. However, there is no founded weak repair (and thus no founded repair either).*

Last but not least, the next example illustrates *circularity of support*: each update action is individually founded because the others happen to be in the repair.

Example 2.5 ([Caroprese and Truszczynski, 2011], Example 3)**.** *Let $V = \emptyset$ and $\eta = \Big\{\langle p \vee q,\ \{p \leftarrow \top\}\rangle, \langle \neg p \vee q, \{q \leftarrow \top\}\rangle, \langle p \vee \neg q, \{p \leftarrow \top\}\rangle\Big\}$. The set $\{p \leftarrow \top, q \leftarrow \top\}$ is the only founded weak repair of V by η: $p \leftarrow \top$ is founded on $\langle p \vee \neg q, \{p \leftarrow \top\}\rangle$ and $q \leftarrow \top$ is founded on $\langle \neg p \vee q, \{q \leftarrow \top\}\rangle$. It is also a founded repair.*

Such repairs were considered to be unintended in [Caroprese and Truszczynski, 2011] and the notion of *justified repair* was proposed to overcome the problem. We discuss this issue further in Section 2.5.3.

Justified Weak Repairs and Justified Repairs

Justified repairs use a stronger condition than foundedness in order to avoid the aforementioned circular dependencies. We start with the definition of a *closed* set of update actions.

Definition 2.6. *Let η be a set of standard active constraints. For $r \in \eta$, all the literals in $\mathsf{C}(r)$ which have no preferred update actions in $\mathsf{R}(r)$ are called* non-updatable. *A set of update actions U is closed under an $r \in \eta$ when the following holds: if the update actions in U* falsify *all the non-updatable literals in $\mathsf{C}(r)$, then U must contain an update action from $\mathsf{R}(r)$. Furthermore, U is closed under η if it is closed under every $r \in \eta$.*

For example, $\{p \leftarrow \top, q \leftarrow \top\}$ is closed under $\langle p \vee \neg q, \{p \leftarrow \top\}\rangle$, $\langle \neg p \vee q, \{q \leftarrow \top\}\rangle$ and $\langle p \vee q,\ \{p \leftarrow \top\}\rangle$, while it is neither closed under $\langle \neg p \vee \neg q, \{p \leftarrow \bot\}\rangle$ nor under $\langle \neg p \vee r,\ \{r \leftarrow \top\}\rangle$. The second step is to define the *no-effect actions* associated with an initial database V and an updated database V'.

Definition 2.7. *Let V and V' be two databases. The update action $p \leftarrow \top$ is a no-effect action of (V, V') if $p \in V \cap V'$ and the update action $p \leftarrow \bot$ is a no-effect action of (V, V') if $p \notin V \cup V'$. The set $\mathsf{ne}(V, V')$ denotes the set of all no-effect actions of (V, V').*

Clearly, for given V and U, we have that $V \diamond U = V \diamond (U \setminus U')$ for every $U' \subseteq \mathsf{ne}(V, V \diamond U)$. Returning to our initial aim now, the definitions of a *justified weak repair* and a *justified repair* are the following.

Definition 2.8. *Let V be a database and let η be a set of standard active constraints. A consistent set of update actions U is a justified weak repair of V by η iff:*[3]

(a) $U \cap \mathsf{ne}(V, V \diamond U) = \emptyset$ *(no 'no-effect' actions)*

(b) $U \cup \mathsf{ne}(V, V \diamond U)$ *is closed under* η

(c) there is no $U' \subset U \cup \mathsf{ne}(V, V \diamond U)$ *such that:*

[3]The original definition of justified weak repairs is slightly different than the one given here. However it is shown that the two are equivalent in [Caroprese and Truszczynski, 2011, Theorem 1].

(1) U' *contains* $\mathsf{ne}(V, V \diamond U)$

(2) U' *is closed under* η

Finally, if U *is also a PMA repair of* V *achieving* $\mathsf{C}(\eta)$*, then* U *is a justified repair of* V *by* η*.*

The next theorem shows the relationship between founded and justified repairs.

Theorem 2.1 ([Caroprese and Truszczynski, 2011], Corollaries 1 and 2)**.** *Let* V *be a database,* U *a consistent set of update actions and* η *a set of standard active constraints. If* U *is a justified weak repair of* V *by* η*, then* U *is also a founded weak repair of* V *by* η *(and likewise, if it is a justified repair of* V *by* η*, then it is also a founded repair of* V *by* η*).*

The next example shows that the converse does not hold. Furthermore, it illustrates that for justified repairs circularity of support is no longer an issue.

Example 2.6 ([Caroprese and Truszczynski, 2011], Example 5)**.** *Consider again* $V = \emptyset$ *and* $\eta = \Big\{\langle p \vee q, \{p \leftarrow \top\}\rangle,\ \langle \neg p \vee q, \{q \leftarrow \top\}\rangle,\ \langle p \vee \neg q,\ \{p \leftarrow \top\}\rangle\Big\}$*. In contrast with Example 2.5 and its founded repair, there is no justified weak repair of* V *by* η*. As justified weak repairs are also founded weak repairs, we only have to check whether* $U = \{p \leftarrow \top, q \leftarrow \top\}$ *is a justified weak repair of* V *by* η*. Supposing that* $\mathbb{P} = \{p, q\}$*, we have* $\mathsf{ne}(V, V \diamond U) = \mathsf{ne}(\emptyset, \{p, q\}) = \emptyset$ *and* $U \cup \mathsf{ne}(V, V \diamond U) = U$ *is not a minimal set of update actions containing* $\mathsf{ne}(V, V \diamond U)$ *and closed under* η*, as* $\emptyset$ *also has these properties.*

However if we replace η *with* $\eta' = \Big\{\langle p \vee q, \{p \leftarrow \top, q \leftarrow \top\}\rangle, \langle \neg p \vee q, \{q \leftarrow \top\}\rangle, \langle p \vee \neg q, \{p \leftarrow \top\}\rangle\Big\}$ *then the set* $\{p \leftarrow \top, q \leftarrow \top\}$ *is both a justified and a founded repair of* V *by* η' *(and both are the only ones).*

In the next two sections, we show that weak, PMA, founded and justified repairs can be captured in DL-PA.

2.3 Repairs and Weak Repairs in DL-PA

We now embed Winslett's standard semantics (and thereby weak repairs) and the PMA (and thereby repairs *tout court*) into DL-PA. This was already done in [Herzig, 2014], but our embeddings are slightly more elegant and are presented in a more uniform and streamlined way. We start with some auxiliary definitions.

To each propositional variable p we associate a *fresh* propositional variable $p^{\pm}$. Each $p^{\pm}$ will register whether or not the proposition p has been modified along the update.[4] This is necessary to ensure that every variable is modified at most once during a repair. We extend the definition to sets of variables $P \subseteq \mathbb{P}$: $P^{\pm} = \{p^{\pm} \mid p \in P\}$. With the information stored in the fresh variables $\mathbb{P}^{\pm}$, we can retrieve the initial valuation from a valuation $V \subseteq \mathbb{P} \cup \mathbb{P}^{\pm}$ through the set:

$$\{p \in V \,:\, p^{\pm} \notin V\} \cup \{p \notin V \,:\, p^{\pm} \in V\}$$

First, we need a program that sets all the propositions in a given set P to $\bot$: $P \leftarrow \bot$ is the sequence of assignments $p \leftarrow \bot$ for all $p \in P$ (whose order does not matter, see Proposition 1.3). Second, the following two DL-PA programs (1) modify a single proposition and store this and (2) undo that modification:

$$\begin{array}{ll} \mathsf{toggle}(p) = \textbf{if } \neg p^{\pm} \textbf{ then } p \leftarrow \neg p; p^{\pm} \leftarrow \top \textbf{ else fail} & = \neg p^{\pm}?; p \leftarrow \neg p; p^{\pm} \leftarrow \top \\ \mathsf{undo}(p) \;\; = \textbf{if } p^{\pm} \textbf{ then } p \leftarrow \neg p; p^{\pm} \leftarrow \bot \textbf{ else fail} & = p^{\pm}?; p \leftarrow \neg p; p^{\pm} \leftarrow \bot \end{array}$$

As announced above, $p^{\pm}$ keeps track of the modifications of p: we are going to ensure that it is true only once p has been modified during the current update. The program $\mathsf{toggle}(p)$ flips the truth value of p if this value has not been modified yet and records the modification by setting $p^{\pm}$ to $\top$; if p has already been made true then $\mathsf{toggle}(p)$ fails. The program $\mathsf{undo}(p)$ undoes this.

It is easy to see then that starting from a database V that contains none of the variables $p^{\pm}$, a weak repair of V achieving C can be obtained through the following DL-PA program:

[4] The difference with [Herzig, 2014] is that our programs memorise that a variable has been flipped instead of storing its previous value.

$$\mathsf{weakRepair}(C) = \left(\bigcup_{p \in \mathbb{P}_C} \mathsf{toggle}(p) \right)^* ; \left(\bigwedge C \right)?$$

Since each variable can be updated at most once and since the order of the updates does not matter, this can be rewritten without the Kleene star as a sequence:

$$\left(\mathsf{toggle}(p_1) \cup \mathbf{skip}\right) ; \ldots ; \left(\mathsf{toggle}(p_k) \cup \mathbf{skip}\right) ; \left(\bigwedge C \right)?$$

where $p_1, \ldots, p_k$ are the variables in $\mathbb{P}_C$. Furthermore, given that none of the variables $p^{\pm}$ occur in the database, the program $\mathsf{toggle}(p)$ simplifies to just: $p \leftarrow \neg p; p^{\pm} \leftarrow \top$.

Finally, we define the following DL-PA formula:

$$\mathsf{Minimal}(C) = \neg \left\langle \left(\bigcup_{p \in \mathbb{P}_C} \mathsf{undo}(p) \right)^+ \right\rangle \bigwedge C$$

The program in this formula undoes a nonempty set of $\mathsf{toggle}(p)$ actions (and nondeterministically so, failing when there was no change at all). Thus, the formula $\mathsf{Minimal}(C)$ says that there is no execution of that program leading to a database closer to the actual database that satisfies the constraints. Hence the actual database corresponds to a minimal change of the initial database. We sum up all the above in the following theorem.

Theorem 2.2. *Let C be a set of static integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database (i.e., no $p^{\pm}$ occurs in either of them). Let U be a consistent set of update actions that is relevant w.r.t. V.*

- *U is a weak repair of V achieving C iff:*

$$\langle V, V \diamond U \rangle \in ||\mathsf{weakRepair}(C) \; ; \; {\mathbb{P}_C}^{\pm} \leftarrow \bot||$$

- *U is a PMA repair of V achieving C iff:*

$$\langle V, V \diamond U \rangle \in ||\mathsf{weakRepair}(C) \; ; \; \mathsf{Minimal}(C)? \; ; {\mathbb{P}_C}^{\pm} \leftarrow \bot||$$

Proof. For the first item, when $V \subseteq \mathbb{P}$ observe that $\langle V, V' \rangle \in ||\mathsf{weakRepair}(C)||$ iff $V' \in || \bigwedge C||$ and the following holds for all variables $p \in \mathbb{P}$ (i.e., excluding the $p^{\pm}$):

$$\text{(a) } p^{\pm} \in V' \text{ iff } V(p) \neq V'(p)$$

$$\text{(b) if } p \notin \mathbb{P}_C \text{ then } V(p) = V'(p)$$

which means that only p's from C and the associated $p^{\pm}$ were modified.
So $\langle V, V \diamond U\rangle \in ||\mathsf{weakRepair}(C); {\mathbb{P}_C}^{\pm} \leftarrow \bot||$ iff $(V \diamond U) \in ||\bigwedge C||$ and no $p^{\pm}$ exists in U. By Definition 2.1 then, this is equivalent to U being a weak repair of V achieving C.

For the second item, given some actual database V', define the initial database as:

$$V = \{p \in \mathbb{P} \,:\, p \in V' \text{ and } p^{\pm} \notin V'\} \cup \{p \in \mathbb{P} \,:\, p \notin V' \text{ and } p^{\pm} \in V'\}$$

Then $V' \in ||\mathsf{Minimal}(C)||$ iff there is no $V'' \in ||\bigwedge C||$ such that $\mathsf{D}(V, V'') \subset \mathsf{D}(V, V')$.[5] Again then, starting from an initial database $V \subseteq \mathbb{P}$, $\langle V, V \diamond U\rangle \in ||\mathsf{weakRepair}(C); \mathsf{Minimal}(C)?; {\mathbb{P}_C}^{\pm} \leftarrow \bot||$ iff $(V \diamond U) \in ||\bigwedge C||$, there is no $V'' \in ||\bigwedge C||$ such that $\mathsf{D}(V, V'') \subset \mathsf{D}(V, V \diamond U)$ and no $p^{\pm}$ exists in U, which according to Definition 2.8 is equivalent to U being a PMA repair of V achieving C. □

2.4 Founded and Justified Repairs in DL-PA

We will now move on to the embedding of the notions of founded and justified repairs into DL-PA. For this, we will re-use the programs defined in the previous section for finding a weak repair and checking minimality, as well as the set of fresh variables $\mathbb{P}^{\pm}$ we had at our disposal in order to keep track of modifications. Moreover, we will need to define a program for checking the foundedness condition in order to generate the founded repairs as well as programs for adding the no-effect actions to a database and checking if a set of update actions is closed under a set of active constraints.

We start with the embedding of founded repairs into DL-PA, for which we will need the following formula:

$$\mathsf{Founded}(\eta) = \bigwedge_{p \in \mathbb{P}_{\mathsf{C}(\eta)}} \left(p^{\pm} \rightarrow \bigvee_{\substack{r \in \eta \\ p \leftarrow \mathsf{X} \in \mathsf{R}(r)}} \langle p \leftarrow \neg p \rangle \neg \mathsf{C}(r) \right)$$

[5] Note that by definition of $\mathsf{toggle}(p), p \in \mathsf{D}(V, V')$ is equivalent to $p^{\pm} \in \mathsf{D}(V, V')$ thus the inclusion $\mathsf{D}(V, V'') \subset \mathsf{D}(V, V')$ is not affected by the variables in ${\mathbb{P}_C}^{\pm}$.

where X ranges over $\{\top, \bot\}$. As we will see, the formula is true if and only if all current update actions (encoded in the current valuation by means of the fresh variables $p^{\pm}$) are founded.

The embedding of justified repairs into DL-PA is a bit more complex. Firstly, we use the set $\mathsf{nup}(r)$ of all the *non-updatable* literals in r that we saw in Definition 2.6, i.e., all the literals in $\mathsf{C}(r)$ for which there is no preferred update action in $\mathsf{R}(r)$. By $\mathsf{nup}(r)^+$ we denote the set of propositional variables of the form p in $\mathsf{nup}(r)$, i.e., $\mathsf{nup}(r)^+ = \mathsf{nup}(r) \cap \mathbb{P}$. Similarly, $\mathsf{nup}(r)^-$ comprises the propositional variables of the form $\neg p$ in $\mathsf{nup}(r)$, i.e., $\mathsf{nup}(r)^- = \{\neg p \mid p \in \mathsf{nup}(r) \setminus \mathbb{P}\}$. Furthermore, we introduce two new sets of *fresh* propositional variables, P^+ and P^-, defined similarly to $P^{\pm}$ as follows: $P^+ = \{p^+ \mid p \in P\}$ and $P^- = \{p^- \mid p \in P\}$. Intuitively, the proposition p^+ will keep track of the no-effect action of the form $p \leftarrow \top$, while the proposition p^- will keep track of the no-effect action of the form $p \leftarrow \bot$. This is realized by the following two programs:

$$\mathsf{ne}^+(p) = \left(p \wedge \neg p^{\pm}\right)?\,; p^+ \leftarrow \top$$
$$\mathsf{ne}^-(p) = \left(\neg p \wedge \neg p^{\pm}\right)?\,; p^- \leftarrow \top$$

Moreover, we will need a program that skips when neither $p \leftarrow \top$ nor $p \leftarrow \bot$ is a no-effect action and fails otherwise (where 'ea' stands for 'effect action'):

$$\mathsf{ea}(p) = p^{\pm}?$$

Then we associate with the current database all the no-effect actions between the initial database and the current state through the following program:

$$\mathsf{ne}(\eta) = \left(\mathsf{ne}^+(p_1) \cup \mathsf{ne}^-(p_1) \cup \mathsf{ea}(p_1)\right); \ldots; \left(\mathsf{ne}^+(p_k) \cup \mathsf{ne}^-(p_k) \cup \mathsf{ea}(p_k)\right)$$

where $p_1, \ldots, p_k$ are the variables in $\mathbb{P}_{\mathsf{C}(\eta)}$ (so the no-effect actions that are added are only those that are relevant w.r.t. η). The next lemma will be helpful in the proof of Theorem 2.4 below.

Lemma 2.1. *Let η be a set of standard active constraints in the language of $\mathbb{P}$ and let $V_0 \subseteq \mathbb{P}$ be a database (so no $p^{\pm}, p^+$ and p^- occurs in either of them). Let V be some repaired database (containing variables $p^{\pm}$) and let $V' = V \cap \mathbb{P}$ be the database V*

without the variables $p^{\pm}$*. Furthermore, let* U *be a consistent set of update actions and let* $U' = \{p^+ \leftarrow \top \mid p \leftarrow \top \in U\} \cup \{p^- \leftarrow \top \mid p \leftarrow \bot \in U\}$*. Then* U *contains all the no-effect actions between* V_0 *and* V' *that are relevant w.r.t.* η *iff* $\langle V, V \diamond U' \rangle \in \Big|\Big|\mathsf{ne}(\eta)\Big|\Big|$.

Proof. For the left-to-right direction, $\langle V, V \diamond U' \rangle \in \Big|\Big|\mathsf{ne}(\eta)\Big|\Big|$ if all the extra variables in $(V \diamond U') \setminus V$ were added through the programs $\mathsf{ne}^+(p)$ and $\mathsf{ne}^-(p)$ for all $p \in \mathbb{P}_{\mathsf{C}(\eta)}$ (note that $V \subseteq V \diamond U'$). But by definition, if $p \leftarrow \top$ is a no-effect action of (V_0, V') then $p \in V_0 \cap V'$, which means that $p \wedge \neg p^{\pm} \in V$. Similarly, if $p \leftarrow \bot$ is a no-effect action of (V_0, V') then $p \notin V_0 \cup V'$, which means that $\neg p \wedge \neg p^{\pm} \in V$. But these are exactly the test programs in $\mathsf{ne}^+(p)$ and $\mathsf{ne}^-(p)$.

For the right-to-left direction, let $p^+ \in V \diamond U'$. Since V contains no p^+ variables, this means that $p^+ \leftarrow \top \in U'$ and consequently $p \leftarrow \top \in U$. Furthermore, since $\langle V, V \diamond U' \rangle \in \Big|\Big|\mathsf{ne}(\eta)\Big|\Big|$, the only way for $p^+ \in V \diamond U'$ is through the program $\mathsf{ne}^+(p)$ in $\mathsf{ne}(\eta)$. So this means that $p \wedge \neg p^{\pm} \in V$, which makes $p \leftarrow \top$ a no-effect action between V_0 and V'. Similarly for $p^- \in V \diamond U'$. Finally, since the program $\mathsf{ne}(\eta)$ spans across all variables in $\mathbb{P}_{\mathsf{C}(\eta)}$, the only way for both $p^+ \notin V \diamond U'$ and $p^- \notin V \diamond U'$ is through the program $\mathsf{ea}(p)$. This means then that $p^{\pm} \in V$ and p is not a no-effect action between V_0 and V'. Thus U contains all the no-effect actions between V_0 and V' that are relevant w.r.t. η. □

Next, we define $\mathsf{A}(\mathsf{r})$ and $\mathsf{B}(\mathsf{r})$ to be the following formulas:

$$\mathsf{A}(\mathsf{r}) = \Big(\bigwedge_{p \in \mathsf{nup}(r)^+} (\neg p \wedge p^{\pm}) \vee (\neg p \wedge p^-)\Big) \wedge \Big(\bigwedge_{p \in \mathsf{nup}(r)^-} (p \wedge p^{\pm}) \vee (p \wedge p^+)\Big)$$

$$\mathsf{B}(\mathsf{r}) = \Big(\bigvee_{p \leftarrow \top \in \mathsf{R}(r)} p\Big) \vee \Big(\bigvee_{p \leftarrow \bot \in \mathsf{R}(r)} \neg p\Big)$$

Using these then we define:

$$\mathsf{Closed}(\eta) = \bigwedge_{r \in \eta} \Big(\mathsf{A}(\mathsf{r}) \rightarrow \mathsf{B}(\mathsf{r})\Big)$$

$$\mathsf{MinimallyClosed}(\eta) = \neg \Big\langle \Big(\bigcup_{p \in \mathbb{P}_{\mathsf{C}(\eta)}} \mathsf{undo}(p)\Big)^+ \Big\rangle \mathsf{Closed}(\eta)$$

The next lemma will be once again used in the proof of Theorem 2.4 below.

Lemma 2.2. *Let* η *be a set of standard active constraints in the language of* $\mathbb{P}$ *and let*

$V_0 \subseteq \mathbb{P}$ be a database (so no $p^{\pm}, p^{+}$ and p^{-} occurs in either of them). Let V_1 be some repaired database (containing variables $p^{\pm}$) and let $V_1' = V_1 \cap \mathbb{P}$ be the database V_1 without the variables $p^{\pm}$. Furthermore, let

$$V_2 = V_1 \cup \{p^{+} \mid p \leftarrow \top \in \mathsf{ne}(V_0, V_1')\} \cup \{p^{-} \mid p \leftarrow \bot \in \mathsf{ne}(V_0, V_1')\}$$

be the extension of V_1 containing all no-effect actions between V_0 and V_1' encoded through the variables p^{+} and p^{-}. Finally, let U comprise the update actions that repaired V_0 to V_1', plus all no-effect actions between V_0 and V_1'. Then U is closed under η iff $V_2 \in \big|\!\big|\mathsf{Closed}(\eta)\big|\!\big|$.

Proof. For the left-to-right direction, it is easy to see that if $V_2 \models \mathsf{A}(\mathsf{r})$ for some $r \in \eta$ then V_2 falsifies all non-updatable literals in r. But by construction the variables in these literals either come from the repairing of V_0 to V_1 (through $p^{\pm}$), or from adding the no-effect actions to V_1 (through p^{+} and p^{-}). This means that they come from update actions in U, which by hypothesis is closed under η. Thus U must contain an update action from $\mathsf{R}(r)$ and $V_2 \models \mathsf{B}(\mathsf{r})$. Since $r \in \eta$ was arbitrary, $V_2 \in \big|\!\big|\mathsf{Closed}(\eta)\big|\!\big|$.

For the right-to-left direction, let the update actions in U falsify all non-updatable literals in $\mathsf{C}(r)$ for some $r \in \eta$. By construction again, this means that $V_2 \models \mathsf{A}(\mathsf{r})$. But by hypothesis $V_2 \in \big|\!\big|\mathsf{Closed}(\eta)\big|\!\big|$, so $V_2 \models \mathsf{B}(\mathsf{r})$ as well. If also $V_0 \models \mathsf{B}(\mathsf{r})$, then either the constraint $r \in \eta$ was already satisfied from the beginning and the update action $p \leftarrow \mathsf{X} \in \mathsf{R}(r)$ such that $V_0 \models \mathsf{B}(\mathsf{r})$ is a no-effect action, or it got satisfied by another update action in U. In all cases U contains an update action from $\mathsf{R}(r)$, which means U is closed under r. Since $r \in \eta$ was arbitrary, U is closed under η. □

So given a set of standard active constraints η, the first formula is true exactly when all current update actions (again, encoded in the current valuation through the set $P^{\pm}$) plus all the no-effect actions (encoded in the current valuation through the sets P^{+} and P^{-}) are closed under η, while the second formula is true if and only if the set comprising of all current update actions is the minimal set of update actions that (together with the no-effect actions) has this property. Finally, we define the following abbreviations:

$$\mathsf{Justified}(\eta)? = \mathsf{ne}(\eta)\ ;\ \mathsf{Closed}(\eta)?\ ;\ \mathsf{MinimallyClosed}(\eta)?$$

$$\mathsf{ClearAll} = {\mathbb{P}_C}^{\pm} \leftarrow \bot \,;\, {\mathbb{P}_C}^{+} \leftarrow \bot \,;\, {\mathbb{P}_C}^{-} \leftarrow \bot$$

The following two theorems now give a complete characterisation of founded and justified repairs in terms of DL-PA programs.

Theorem 2.3. *Let η be a set of active integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database (i.e., no $p^{\pm}$ occurs in either of them). Let U be a consistent set of update actions that is relevant w.r.t. V.*

- *U is a founded weak repair of V by η iff:*

$$\langle V, V \diamond U \rangle \in \Big|\Big|\mathsf{weakRepair}(\mathsf{C}(\eta)) \,;\, \mathsf{Founded}(\eta)? \,;\, {\mathbb{P}_C}^{\pm} \leftarrow \bot\Big|\Big|$$

- *U is a founded repair of V by η iff:*

$$\langle V, V \diamond U \rangle \in \Big|\Big|\mathsf{weakRepair}(\mathsf{C}(\eta)) \,;\, \mathsf{Founded}(\eta)? \,;\, \mathsf{Minimal}(\mathsf{C}(\eta))? \,;\, {\mathbb{P}_C}^{\pm} \leftarrow \bot\Big|\Big|$$

Proof. Let V_0 be the initial database and suppose V is some repaired database (containing variables $p^{\pm}$). Define the set of update actions:

$$U_{V,\eta} = \{p \leftarrow \top \;:\; p^{\pm} \in V \text{ and } p \in V\} \cup \{p \leftarrow \bot \;:\; p^{\pm} \in V \text{ and } p \notin V\}$$

Let us prove that $\langle V_0, V \rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta)); \mathsf{Founded}(\eta)?||$ iff $U_{V,\eta}$ is a founded weak repair of V_0 by η. The latter means that for every $\alpha \in U_{V,\eta}$ the three conditions (a) $\alpha \in \mathsf{R}(r)$, (b) $V_0 \diamond U_{V,\eta} \models \mathsf{C}(r)$ and (c) $V_0 \diamond (U_{V,\eta} \setminus \{\alpha\}) \not\models \mathsf{C}(r)$ of Definition 2.4 are satisfied.

For the left-to-right direction consider some $p \leftarrow \top \in U_{V,\eta}$. Then $p^{\pm} \in V$. First of all, $\langle V_0, V \rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta))||$ so $U_{V,\eta}$ is a weak repair of V_0 by η. Condition (b) is satisfied from the definition $\mathsf{weakRepair}(\mathsf{C}(\eta))$ and Theorem 2.2. Condition (a) is satisfied by the existence of a candidate constraint in the definition of $\mathsf{Founded}(\eta)$; remark that we are guaranteed that the constraint contains indeed $p \leftarrow \top$, as opposed to $p \leftarrow \bot$, because undoing the change on p changes $\mathsf{C}(r)$ to false (so X has to be $\top$). Condition (c) is satisfied because $V_0 \diamond (U \setminus \{p \leftarrow \top\}) \not\models \mathsf{C}(r)$ is equivalent to $V_0 \diamond U \models \neg\langle p \leftarrow \bot\rangle \mathsf{C}(r)$. Similarly for some $p \leftarrow \bot \in U_{V,\eta}$. So $U_{V,\eta}$ is a weak repair and satisfies the three conditions, making

it a founded weak repair of V_0 by η.

For the right-to-left direction, since $U_{V,\eta}$ is also a weak repair of V_0 by η, Theorem 2.2 ensures that $\langle V_0, V\rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta))||$ (remember that we chose the repaired database V to contain variables $p^{\pm}$). To prove that $V \in ||\mathsf{Founded}(\eta)||$, consider some $p^{\pm} \in V$. By definition, it entails $p \leftarrow \mathsf{X} \in U_{V,\eta}$ for some $\mathsf{X} \in \{\top, \bot\}$. Condition (a) ensures that there is a constraint $r \in \eta$ with $p \leftarrow \mathsf{X} \in \mathsf{R}(r)$. Condition (c) implies $V \models \neg\langle p \leftarrow \neg\mathsf{X}\rangle \mathsf{C}(r)$. So $\langle V, V\rangle \in ||\mathsf{Founded}(\eta)?||$ and thus $\langle V_0, V\rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta)); \mathsf{Founded}(\eta)?||$.

The rest of the proof (regarding minimality and non-existence of any $p^{\pm}$ in U) is similar to that of Theorem 2.2. □

Theorem 2.4. *Let η be a set of standard active constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database (so no $p^{\pm}, p^{+}$ and p^{-} occurs in either of them). Let U be a consistent set of update actions that is relevant w.r.t. V.*

- *U is a justified weak repair of V by η iff:*

$$\langle V, V \diamond U\rangle \in \Big|\Big|\mathsf{weakRepair}(\mathsf{C}(\eta)) \,;\, \mathsf{Justified}(\eta)? \,;\, \mathsf{ClearAll}\Big|\Big|$$

- *U is a justified repair of V by η iff:*

$$\langle V, V \diamond U\rangle \in \Big|\Big|\mathsf{weakRepair}(\mathsf{C}(\eta)) \,;\, \mathsf{Justified}(\eta)? \,;\, \mathsf{Minimal}(\mathsf{C}(\eta))? \,;\, \mathsf{ClearAll}\Big|\Big|$$

Proof. First of all, notice that we can reduce the set of no-effect actions to those only concerning the set of active integrity constraints η, since the rest have no impact on the repairing procedure and produce the same results. Without loss of generality, we can assume that the set $\mathsf{ne}(V, V \diamond U)$ only contains those no-effect actions that are relevant w.r.t. η, for all V and U. Similarly to the previous proof, let V_0 be the initial database, V_1 some repaired database (containing variables $p^{\pm}$) and define the set of update actions:

$$U = \{p \leftarrow \top \;:\; p^{\pm} \in V_1 \text{ and } p \in V_1\} \cup \{p \leftarrow \bot \;:\; p^{\pm} \in V_1 \text{ and } p \notin V_1\}$$

Furthermore, let V_2 extend V_1 with variables p^{+} and p^{-} which encode the no-effect actions between V_0 and $V_0 \diamond U$:

$$V_2 = V_1 \cup \{p^+ \mid p \leftarrow \top \in \mathsf{ne}(V_0, V_0 \diamond U)\} \cup \{p^- \mid p \leftarrow \bot \in \mathsf{ne}(V_0, V_0 \diamond U)\}$$

We have to prove that $\langle V_0, V_2\rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta)); \mathsf{Justified}(\eta)?||$ iff U is a justified weak repair of V_0 by η. According to Definition 2.8, the latter is equivalent to the fact that (a) $U \cap \mathsf{ne}(V_0, V_0 \diamond U) = \emptyset$, (b) $U \cup \mathsf{ne}(V_0, V_0 \diamond U)$ is closed under η and (c) $U \cup \mathsf{ne}(V_0, V_0 \diamond U)$ is a minimal set containing $\mathsf{ne}(V_0, V_0 \diamond U)$ which is closed under η. Note that by construction, V_1 is the repaired database $V_0 \diamond U$ together with the variables $p^{\pm}$.

For the left-to-right direction, the construction of V_2 through the use of the programs $\mathsf{weakRepair}(\mathsf{C}(\eta))$, $\mathsf{ne}(\eta)$, $\mathsf{Closed}(\eta)?$ and $\mathsf{MinimallyClosed}(\eta)?$ gives:

(1) $\langle V_2, V_2\rangle \in \left|\left|\mathsf{Closed}(\eta)?\right|\right|$ and thus $V_2 \in \left|\left|\mathsf{Closed}(\eta)\right|\right|$

(2) $\langle V_2, V_2\rangle \in \left|\left|\mathsf{MinimallyClosed}(\eta)?\right|\right|$ and thus $V_2 \in \left|\left|\mathsf{MinimallyClosed}(\eta)\right|\right|$

Property (a) is derived from the fact that U is relevant w.r.t. V_0. Property (b) is immediate by (1) and Lemma 2.2. So the set $U \cup \mathsf{ne}(V_0, V_0 \diamond U)$ obviously contains the set $\mathsf{ne}(V_0, V_0 \diamond U)$ and is also closed under η. For property (c), suppose there exists a repair U' such that $U' \subset U \cup \mathsf{ne}(V_0, V_0 \diamond U)$ and which also contains the set $\mathsf{ne}(V_0, V_0 \diamond U)$ and is closed under η, and let $U'' = U' \setminus \mathsf{ne}(V_0, V_0 \diamond U)$. Define also V' to be $V_0 \diamond U''$, together with the variables $p^{\pm}$ that keep track of the changes between V_0 and $V_0 \diamond U''$ plus the variables p^+ and p^- that encode the no-effect actions in $\mathsf{ne}(V_0, V_0 \diamond U)$. Then $\langle V', V'\rangle \in ||\mathsf{Closed}(\eta)?||$ and since $U'' \subset U$ we have that $V_2 \notin \left|\left|\mathsf{MinimallyClosed}(\eta)\right|\right|$, which is contrary to property (2).

For the right-to-left direction, since U is also a weak repair of V_0 by η, Theorem 2.2 ensures that $\langle V_0, V_1\rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta))||$ (remember that we chose the repaired database V_1 to contain variables $p^{\pm}$). Then $\langle V_1, V_2\rangle \in ||\mathsf{ne}(\eta)||$ by Lemma 2.1. It follows from Lemma 2.2 and property (b) that $V_2 \in ||\mathsf{Closed}(\eta)||$. Since by property (c) $U \cup \mathsf{ne}(V_0, V_0 \diamond U)$ is a minimal set containing $\mathsf{ne}(V_0, V_0 \diamond U)$ which is closed under η, then $V_2 \in ||\mathsf{MinimallyClosed}(\eta)||$ as well. So $\langle V_2, V_2\rangle \in ||\mathsf{Closed}(\eta)?||$ and $\langle V_2, V_2\rangle \in ||\mathsf{MinimallyClosed}(\eta)?||$ and thus $\langle V_0, V_2\rangle \in ||\mathsf{weakRepair}(\mathsf{C}(\eta)); \mathsf{Justified}(\eta)?||$.

Once again, the rest of the proof (regarding minimality and non-existence of any $p^{\pm}, p^+$ and p^- in U) is similar to that of Theorem 2.2. □

2.5 A New Definition of Repair in DL-PA

We now propose some new definitions that take advantage of the resources of DL-PA. More precisely, we make use of **while** loops in order to iterate the application of active constraints. We start by discussing how databases can be repaired by applying active constraints in sequence. This will lead us to the definition of dynamic repair. We show that it is incomparable with both founded and justified repairs and discuss its properties and some variants.

2.5.1 Repairing a Database: a Dynamic View

Suppose there is only one active constraint r that is standard. Then it is clear how to proceed: either $V \models \mathsf{C}(r)$ and there is nothing to do, or $V \not\models \mathsf{C}(r)$ and we have to apply r. In the second case, each $\alpha_i \in \mathsf{R}(r)$ provides a PMA repair of V achieving $\mathsf{C}(r)$.[6] What about the case where $\mathsf{R}(r)$ is empty? Well, then V cannot be repaired and we are stuck.

So far so good. The situation may get way more intricate when the set of active constraints η contains two or more elements that can interact. Firstly, the example of the introductory Section 2.1 and Example 2.4 illustrated an instance of active constraints which intuitively should have a repair (and it does, in the case of PMA repairs) but for which there is no founded or justified weak repair. We would like to find a definition of a repair which depends only on the preferred update actions and always provides a repaired database, as long as there are update actions from each $\mathsf{C}(r)$ to choose from. Moreover, even for standard active constraints it might not be enough to apply an update action $\alpha_i \in \bigcup_{r\in\eta} \mathsf{R}(r)$ only once: some of the active constraints might have to be applied several times in order to obtain integrity. The following active constraints that are inspired by an $(n{+}1)$-bit counter highlight this.

Suppose for $n \geq 0$ we represent binary numbers up to $2^{n+1}{-}1$ by means of $n{+}1$ propositional variables: $\neg p_n \wedge \cdots \wedge \neg p_0$ represents the integer zero and $p_n \wedge \cdots \wedge p_0$ repre-

[6] For our more general active constraints where there is no syntactical link between $\mathsf{C}(r)$ and $\mathsf{R}(r)$ we have to compute all possible minimal subsets $U \subseteq \mathsf{R}(r)$ such that $V \diamond U \models \mathsf{C}(r)$. All of them will be PMA repairs.

sents $2^{n+1}-1$. For each bit we also need an auxiliary variable p_i. Let:

$$
\begin{aligned}
r_1 &= \langle p_0 \vee x_1 \vee \cdots \vee x_n, \{p_0 \leftarrow \top\}\rangle \\
r_{2_k} &= \langle p_k \vee \neg p_{k-1} \vee \cdots \vee \neg p_0 \vee x_k, \{x_k \leftarrow \top\}\rangle, \quad \text{for } 1 \leq k \leq n \\
r_{3_{k_i}} &= \langle \neg p_i \vee \neg x_k, \{p_i \leftarrow \bot\}\rangle, \quad \text{for } 1 \leq k \leq n \text{ and } 0 \leq i \leq k-1 \\
r_{3_{k_k}} &= \langle p_k \vee \neg x_k, \{p_k \leftarrow \top\}\rangle, \quad \text{for } 1 \leq k \leq n \\
r_{4_k} &= \langle \neg p_k \vee p_{k-1} \vee \cdots \vee p_0 \vee \neg x_k, \{x_k \leftarrow \bot\}\rangle, \quad \text{for } 1 \leq k \leq n
\end{aligned}
$$

The idea is that when $\neg p_k \wedge p_{k-1} \wedge \cdots \wedge p_0$ is true, i.e., when the number $011\ldots1$ has to be incremented to $100\ldots0$, then x_k is made true by r_{2_k} and remains so until $100\ldots0$ has been attained. This involves flipping the k digits in the conjunction $\neg p_k \wedge p_{k-1} \wedge \cdots \wedge p_0$: with active constraints this is done one-by-one by $r_{3_{k_i}}$ and $r_{3_{k_k}}$. Then x_k is set to false again by r_{4_k}. Let η_n be the set of all the above rules, for a given n, i.e., η_n is the set:

$$\Big\{r_1\} \cup \{r_{2_1}, \ldots, r_{2_n}\} \cup \{r_{3_{1_0}}, r_{3_{1_1}}\} \cup \{r_{3_{2_0}}, r_{3_{2_1}}, r_{3_{2_2}}\} \cup \ldots \cup \{r_{3_{n_0}}, \ldots, r_{3_{n_n}}\} \cup \{r_{4_1}, \ldots, r_{4_n}\}$$

Successive repairing steps then implement an $(n+1)$-bit counter counting from the initial database $\emptyset$ to the database $\{p_n, \ldots, p_0\}$.

The computation takes a number of steps that is exponential in n, while the number of update actions is $\frac{1}{2}(n^2+7n)+1$, demonstrating that sometimes atomic repairs must be performed an exponential number of times: for example r_1 needs to be applied 2^n times in order to repair $V_0 = \emptyset$ by η_n. Let us illustrate by the 3-bit counter how the repairs are done.

Example 2.7. *Let's take $n = 2$ and try to obtain the integer 111 starting from 000. In Figure 2.1 we can see the steps needed through which we will reach the set of update actions $U = \{p_0 \leftarrow \top, p_1 \leftarrow \top, p_2 \leftarrow \top\}$ that will update the database $\emptyset$ in order for it to satisfy the active integrity constraints in η_2. The first column represents the current database (starting from $\emptyset$), the second column shows the constraint that was applied in order to reach it and in the third we see the current integer in the counter. Last but not least, the last column shows when an x_k is kept* true *in order for the procedure to reach the integer 10 from 01 and the integer 100 from 011 when needed.*

$\neg p_2 \wedge \neg p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge \neg x_2$		000	–
$\neg p_2 \wedge \neg p_1 \wedge p_0 \wedge \neg x_1 \wedge \neg x_2$	r_1	001	–
$\neg p_2 \wedge \neg p_1 \wedge p_0 \wedge x_1 \wedge \neg x_2$	r_{2_1}	001	✓
$\neg p_2 \wedge \neg p_1 \wedge \neg p_0 \wedge x_1 \wedge \neg x_2$	$r_{3_{1_0}}$	000	✓
$\neg p_2 \wedge p_1 \wedge \neg p_0 \wedge x_1 \wedge \neg x_2$	$r_{3_{1_1}}$	010	✓
$\neg p_2 \wedge p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge \neg x_2$	r_{4_1}	010	–
$\neg p_2 \wedge p_1 \wedge p_0 \wedge \neg x_1 \wedge \neg x_2$	r_1	011	–
$\neg p_2 \wedge p_1 \wedge p_0 \wedge \neg x_1 \wedge x_2$	r_{2_2}	011	✓
$\neg p_2 \wedge p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge x_2$	$r_{3_{2_0}}$	010	✓
$\neg p_2 \wedge \neg p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge x_2$	$r_{3_{2_1}}$	000	✓
$p_2 \wedge \neg p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge x_2$	$r_{3_{2_2}}$	100	✓
$p_2 \wedge \neg p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge \neg x_2$	r_{4_2}	100	–
$p_2 \wedge \neg p_1 \wedge p_0 \wedge \neg x_1 \wedge \neg x_2$	r_1	101	–
$p_2 \wedge \neg p_1 \wedge p_0 \wedge x_1 \wedge \neg x_2$	r_{2_1}	101	✓
$p_2 \wedge \neg p_1 \wedge \neg p_0 \wedge x_1 \wedge \neg x_2$	$r_{3_{1_0}}$	100	✓
$p_2 \wedge p_1 \wedge \neg p_0 \wedge x_1 \wedge \neg x_2$	$r_{3_{1_1}}$	110	✓
$p_2 \wedge p_1 \wedge \neg p_0 \wedge \neg x_1 \wedge \neg x_2$	r_{4_1}	110	–
$p_2 \wedge p_1 \wedge p_0 \wedge \neg x_1 \wedge \neg x_2$	r_1	111	–

Figure 2.1: Incrementing 000 to 111 through η_2

We can see how some constraints need to be applied many times in order to succeed in repairing the original database. This calls for a dynamic way through which a database is updated in order for it to be repaired: a procedure that modifies the database according to the integrity constraints step by step, until it reaches a satisfactory form (i.e., satisfies the integrity constraints). Founded and justified repairs cannot do the job in this and other scenarios of that kind, as an active constraint can only be used once: indeed, in the example of the $(n+1)$-bit counter, no repair can be obtained by means of founded and justified repairs. That's why we introduce dynamic repairs.

2.5.2 Dynamic Weak Repairs and Dynamic Repairs

We associate with every active constraint r the DL-PA programs:

$$\pi_r = \neg \mathsf{C}(r)\,?\,; \bigcup_{\alpha \in \mathsf{R}(r)} \alpha$$
$$\pi_r^{\pm} = \neg \mathsf{C}(r)\,?\,; \bigcup_{p \leftarrow \mathsf{X} \in \mathsf{R}(r)} \left(p \leftarrow \mathsf{X}; p^{\pm} \leftarrow \top \right)$$

Remember that $\bigcup_{\alpha \in \mathsf{R}(r)} \alpha$ equals **fail** when $\mathsf{R}(r)$ is empty. This matches the intuitive reading that we have given to active constraints in Section 2.2.2: the repair program π_r checks whether the static integrity constraint associated with r is violated, and if so nondeterministically applies one of the update actions from $\mathsf{R}(r)$. The program $\pi_r^{\pm}$ moreover stores that p has been changed. These intuitions are also supported by the following proposition, which tells us that applicability of an active constraint r (the fact that $\mathsf{C}(r)$ is violated) is matched by the DL-PA notion of executability of the program π_r.

Proposition 2.3. *Let r be an active constraint and let V be a database. Then applicability of r at V is equivalent to both $V \models \langle \pi_r \rangle \top$ and $V \models \langle \pi_r^{\pm} \rangle \top$.*

Proof. It suffices to observe that when π is a nondeterministic composition of update actions then the equivalence $\neg \mathsf{C}(r) \leftrightarrow \langle \neg \mathsf{C}(r)?; \pi \rangle \top$ is DL-PA valid for every $\mathsf{C}(r)$. □

Based on these, the definitions of a *dynamic weak repair* and a *dynamic repair* are the following.

Definition 2.9. *Let V be a database and let η be a set of active constraints. A dynamic weak repair of V by η is a consistent set of update actions U such that U is relevant w.r.t. V and:*

$$\langle V, V \diamond U \rangle \in \left\| \textbf{while } \neg \left(\bigwedge \mathsf{C}(\eta) \right) \textbf{ do } \left(\bigcup_{r \in \eta} \pi_r \right) \right\|$$

Moreover, if U is also a PMA repair of V achieving $\mathsf{C}(\eta)$, then U is a dynamic repair of V by η.

In the following example we see that dynamic repairs can sometimes coincide with founded repairs.

Example 2.8 (Example 2.5, ctd.)**.** *Consider again $V = \emptyset$ and $\eta = \big\{ \langle p \vee q, \{p \leftarrow \top\} \rangle, \langle \neg p \vee q, \{q \leftarrow \top\} \rangle, \langle p \vee \neg q, \{p \leftarrow \top\} \rangle \big\}$. There is a single dynamic weak repair (and also dynamic repair) of V by η, viz. $\{p \leftarrow \top, q \leftarrow \top\}$. Remember by Example 2.6 that there is no justified repair.*

As we have already witnessed with the $(n{+}1)$-bit counter though, dynamic weak repairs are not necessarily founded. The next example is simpler.

Example 2.9 (Example 2.3, ctd.)**.** *Consider again* $V = \emptyset$ *and* $\eta = \big\{\langle p, \{p \leftarrow \top\}\rangle, \langle p \vee q, \{q \leftarrow \top\}\rangle\big\}$*, whose only founded weak repair was* $\{p \leftarrow \top\}$*. There are two dynamic weak repairs of* V *by* η*, namely* $\{p \leftarrow \top\}$ *and* $\{p \leftarrow \top, q \leftarrow \top\}$*. Only the former is a dynamic repair.*

Remember also that at the beginning of Section 2.5.1 we argued against founded and justified repairs using Example 2.4 (itself an adaptation of the example discussed in the introductory Section 2.1), for which we would like to have a way to repair V by η. The next examples shows that dynamic weak repairs solve this problem. Let us also note that just like founded and justified repairs, dynamic weak repairs do not necessarily coincide with dynamic repairs.

Example 2.10 (Example 2.4, ctd.)**.** *Consider again* $V = \emptyset$ *and* $\eta = \big\{\langle p \vee q, \{p \leftarrow \top\}\rangle, \langle \neg p \vee q, \{q \leftarrow \top\}\rangle\big\}$*. The only dynamic weak repair of* V *by* η *is the set of update actions* $\{p \leftarrow \top, q \leftarrow \top\}$*. But* $\{q \leftarrow \top\}$ *is the PMA repair of* V *achieving* $\mathsf{C}(\eta)$*, so there is no dynamic repair.*

Finally, in a similar manner as in the previous sections, the next theorem characterises dynamic repairs in terms of DL-PA programs.

Theorem 2.5. *Let* η *be a set of active integrity constraints in the language of* $\mathbb{P}$ *and let* $V \subseteq \mathbb{P}$ *be a database (i.e., no* $p^{\pm}$ *occurs in either of them). Let* U *be a consistent set of update actions that is relevant w.r.t.* V*.* U *is a dynamic repair of* V *by* η *iff:*

$$\langle V, V \diamond U\rangle \in \left\| \mathbf{while}\ \neg \left(\bigwedge \mathsf{C}(\eta)\right)\ \mathbf{do}\ \left(\bigcup_{r\in\eta} \pi_r^{\pm}\right); \mathsf{Minimal}\Big(\mathsf{C}(\eta)\Big)?; {\mathbb{P}_C}^{\pm} \leftarrow \bot \right\|$$

Proof. The proof is quite trivial and based on the definitions. Given a database $V \subseteq \mathbb{P}$ and a set of active integrity constraints η, a dynamic repair of V by η is both a dynamic weak repair of V by η and a PMA repair of V achieving $\mathsf{C}(\eta)$. In DL-PA terms then, this is given by the sequential composition of the programs $\mathbf{while}\ \neg \left(\bigwedge \mathsf{C}(\eta)\right)\ \mathbf{do}\ \left(\bigcup_{r\in\eta} \pi_r^{\pm}\right)$ and $\mathsf{Minimal}(\mathsf{C}(\eta))?$, with the DL-PA program $\pi_r^{\pm}$ keeping track of which propositions have

been modified along the update so that they can be checked again in the latter. Finally, as before the program $\mathbb{P}_C{}^{\pm} \leftarrow \bot$ ensures that no $p^{\pm}$ exists in U. □

2.5.3 Some Interesting Properties

In this subsection we present some interesting properties of this dynamic procedure that distinguishes it from the other main repairs which have been studied and prevailed in the literature, viz. weak repairs, PMA repairs, founded and justified repairs. Our goal is to provide a concrete argument that repairs produced in this way are an interesting kind of repairs, possessing the advantages of the others while not comprising some of their disadvantages.

The main problem with founded repairs is the so called *circularity of support* which has been already mentioned at the end of Section 2.2.2. This undesirable property is what ultimately led to the definition of justified repairs, which are way more complex and difficult to understand, at least at first sight. Dynamic repairs on the other hand provide a solution to this problem without straying too far from the initial definition, making it far less intricate. In contrast with founded repairs which need the *foundedness* property to give priority to the "preferred" repairs, dynamic repairs simply select an update action from the set that they have access to (the set of preferred ones) without checking for any other condition. This leads to no *circular support* between any set of preferred actions and can also be seen in Example 2.8 where $\{p \leftarrow \top, q \leftarrow \top\}$ remains a dynamic repair of V by η even if the constraint '$\langle p \vee \neg q, \{p \leftarrow \top\}\rangle$' is absent (something that cannot be said for founded repairs, as this constraint is required for the foundedness of 'p'). Furthermore, although justified repairs solve this problem, they often do not exist, as can be seen by Example 2.6. Through dynamic repairs we can provide a solution to cases like this, avoiding the *circularity of support* found in founded repairs, while still being able to provide a repaired database. So not only are dynamic repairs more intuitive, but they also comprise the best of both situations.

Despite this however, one could still argue that they are too "strict", sometimes requiring that every integrity constraint in $\mathsf{C}(\eta)$ have a way to be repaired (an update action in $\mathsf{R}(r)$ should exist for all $r \in \eta$ in order to make $\mathsf{C}(r)$ true). If $\mathsf{R}(r) = \emptyset$ for some $r \in \eta$,

then the whole dynamic repairing procedure could collapse and a dynamic weak repair never occur. The next example illustrates exactly this.

Example 2.11. *Let* $V = \{q\}$ *and* $\eta = \Big\{\langle p \vee q, \{p \leftarrow \top\}\rangle, \langle p, \emptyset\rangle\Big\}$. *The set* $\{p \leftarrow \top\}$ *is a PMA repair of* V *achieving* $\mathsf{C}(\eta)$. *However, there are no dynamic weak repairs (and thus no dynamic repairs either).*

As we can see, the problem arises when an integrity constraint has no preferred update actions and cannot be satisfied by the application of some other constraint when repairing the database. One could differentiate this behavior into three classes of repairs, based on the level of "strictness" of the preference that is involved in the active constraints. The more strict repairs are those conforming to the idea that every integrity constraint $\mathsf{C}(r)$ should be repaired only through an update action in $\mathsf{R}(r)$, the less strict allow any update actions in $\bigcup_{r \in \eta} \mathsf{R}(r)$ to be used for any integrity constraint $\mathsf{C}(r)$, while the middle ground is to keep a balance between the two. As we can see, the passing from the more strict class to the others changes the meaning of the update actions in $\mathsf{R}(r)$ from *permitted* repair actions to *preferred* ones, a distinction that is not always made clear in the literature. As Example 2.11 shows, dynamic repairs and dynamic weak repairs possess some of this "strict" nature: an update action will only arise while updating a database if it helps to repair some constraint. This forbids repairs in cases where all clauses apart from those having no preferred actions are already satisfied. In contrast, in such cases solutions with founded weak repairs can occur, as shown in the next example.

Example 2.12 ([Caroprese and Truszczynski, 2011], Example 2)**.** *Let* $V = \emptyset$ *and* $\eta = \Big\{\langle p \vee \neg q \vee \neg r, \{p \leftarrow \top\}\rangle, \langle \neg p \vee q \vee \neg r, \{q \leftarrow \top\}\rangle, \langle \neg p \vee \neg q \vee r, \{r \leftarrow \top\}\rangle, \langle p, \emptyset\rangle\Big\}$. *The set* $\{p \leftarrow \top,\ q \leftarrow \top,\ r \leftarrow \top\}$ *is the only founded weak repair of* V *by* η. *Furthermore, the set* $\{p \leftarrow \top\}$ *is a PMA repair of* V *achieving* $\mathsf{C}(\eta)$. *There are no dynamic weak repairs (and thus no dynamic repairs either).*

So this leads to the following question: should we require a repair to exist in such cases or not? Are we willing to agree with the fact that such databases do not and should not have a repair, or is repairing the database in order to satisfy the integrity constraints in $\mathsf{C}(\eta)$ of the highest priority?

If the answer to the last question is positive then dynamic repairs could be less interesting. We can however tweak the definition slightly and define *global-dynamic* weak repairs to be a kind of dynamic repairs with the same intuitive behavior as before but belonging to the least strict of the aforementioned classes of repairs. More precisely, the reason that dynamic repairs cannot repair a constraint using update actions found in other clauses is the *local* nature of the **do** part in the **while** loop. Before trying to repair the whole set of active integrity constraints in η, a dynamic repair *locally* checks if every clause (integrity constraint) is satisfied. If we drop this requirement and allow the dynamic procedure to *globally* choose update actions found in other clauses, then we will have a solution in Examples 2.11, 2.12 and more generally the cases we have discussed.

Definition 2.10. *Let V be a database and let η be a set of active constraints. A global-dynamic weak repair of V by η is a consistent set of update actions U such that U is relevant w.r.t. V and:*

$$\langle V, V \diamond U \rangle \in \left\| \mathbf{while}\ \neg \left(\bigwedge \mathsf{C}(\eta) \right)\ \mathbf{do}\ \left(\bigcup_{\substack{r \in \eta \\ \alpha \in \mathsf{R}(r)}} \alpha \right) \right\|$$

In the same vein as before, if U is also a PMA repair of V achieving $\mathsf{C}(\eta)$, then U is a global-dynamic repair of V by η.

It is easy to see now that this tweaked definition provides us with the desired repaired database in Examples 2.11 and 2.12. Specifically, the set $\{p \leftarrow \top\}$ is a global-dynamic repair of V by η in both examples. Furthermore, a dynamic weak repair is always a global-dynamic weak repair, as it can be created by the same procedure using one step less, namely by not checking the condition "$\neg\mathsf{C}(r)?$" in the **do** part of the program. This makes dynamic weak repairs a subset of global-dynamic weak repairs (and also dynamic repairs a subset of global-dynamic repairs).

It is our intention to use the global-dynamic repairs mainly as a tool of comparison between the different classes of repairs and less as a practical repairing technique that would replace the others. As we will see, the most important attribute of global-dynamic repairs is that they are exactly the global-dynamic weak repairs that are minimal w.r.t. set inclusion (i.e., if U is a global-dynamic repair of V then there is no global-dynamic weak repair U' of V such that $U' \subset U$). The recipe when defining repairs till now is to first

give the definition of their *weak* versions and then state that they also have to be PMA repairs. This is of course different from saying that these repairs are the weak repairs that are minimal w.r.t. set inclusion, as in this case they would always exist if at least one of their weak counterparts existed. But it is not always the case that they may coincide with PMA repairs and usually may not exist altogether. This can be witnessed in Example 2.12 for founded repairs and in Example 2.10 for dynamic repairs.

However, when minimality w.r.t. set inclusion and coincidence with PMA repairs is the same, we have a much more powerful and reliable tool in our hands that avoids the main disadvantage of other repairs, namely that they may not exist (even if their weak versions do). This is shown in Theorem 2.6. Before this, a small lemma characterises this global-dynamic nature.

Lemma 2.3. *Let η be a set of active integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database. Let U_1 be a global-dynamic weak repair of V by η and U_2 be a weak repair of V achieving $\mathsf{C}(\eta)$ such that $U_2 \subset U_1$. Then U_2 is also a global-dynamic weak repair of V by η.*

Proof. By hypothesis, U_1 and U_2 are both consistent sets of update actions that are relevant w.r.t. V such that $V \diamond U_1 \models \bigwedge \mathsf{C}(\eta)$ and $V \diamond U_2 \models \bigwedge \mathsf{C}(\eta)$ with $U_2 \subset U_1$. This means that V can be updated with less update actions than U_1 in order to satisfy the integrity constraints in η. But U_1 was constructed by iteration on checking the satisfaction of $\bigwedge \mathsf{C}(\eta)$ and applying update actions from $\bigcup_{r \in \eta} \mathsf{R}(r)$. Then U_2 can be constructed in exactly the same way, since it doesn't comprise any update actions outside of U_1, with the difference of taking less update actions into account: namely, by restricting the non-deterministic choice to updates from U_2 and leaving the update actions in $U_1 \setminus U_2$ out of consideration. This will also lead to a repaired database. So U_2 is a global-dynamic weak repair of V by η as well. □

Theorem 2.6. *Let η be a set of active integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database. Let U be a consistent set of update actions that is relevant w.r.t. V. U is a global-dynamic repair of V by η iff U is a global-dynamic weak repair of V by η that is minimal w.r.t. set inclusion.*

Proof. Let the set GDR consist of all the global-dynamic repairs of V by η and let $MGDWR$ consist of all the global-dynamic weak repairs of V by η that are minimal w.r.t. set inclusion. For convenience we also define in the same way R as the set of all PMA repairs and WR as the set of all weak repairs of V achieving $\mathsf{C}(\eta)$. Finally, let $GDWR$ be the set of all global-dynamic weak repairs of V by η. First of all, observe that $GDR = GDWR \cap R$ (1) and $GDWR \subseteq WR$ (2). Let us show that $GDR = MGDWR$.

- $GDR \subseteq MGDWR$: let $U_1 \in GDR$. By (1) then, $U_1 \in GDWR$ and $U_1 \in R$. Let $U_2 \in GDWR$ such that $U_2 \subset U_1$. By (2) we also have that $U_2 \in WR$. This means that $U_1 \in R$ and $U_2 \in WR$ with $U_2 \subset U_1$. But this cannot be the case, as a PMA repair is a minimal weak repair w.r.t. set inclusion. So there is no $U_2 \in GDWR$ such that $U_2 \subset U_1$, where $U_1 \in GDWR$. Thus $U_1 \in MGDWR$.
  Note that this also applies to founded, justified and dynamic repairs. The difference is in the other direction.

- $MGDWR \subseteq GDR$: let $U_1 \in MGDWR$. By definition then, $U_1 \in GDWR$ and there is no $U' \in GDWR$ such that $U' \subset U_1$ (3). Let $U_2 \in WR$ such that $U_2 \subset U_1$. By Lemma 2.3 it is also the case then that $U_2 \in GDWR$. But this cannot be the case by (3). So there is no $U_2 \in WR$ such that $U_2 \subset U_1$. Since by (2) $U_1 \in WR$ as well, this means that $U_1 \in R$. Thus $U_1 \in GDWR \cap R$ and using (1) we get $U_1 \in GDR$.

□

So there is enough evidence to support the idea of using global-dynamic repairs as our repairs of choice when we want to update a database taking into account active integrity constraints in the cases where other repairs don't work. They are the closest thing to a PMA repair as shown by the next Proposition, with the only limitation of being non-existent if the set of update actions $\bigcup_{r \in \eta} \mathsf{R}(r)$ is empty or if an integrity constraint can't be repaired even through update actions existing in other clauses (in both cases a solution to this problem shouldn't exist intuitively).

Proposition 2.4. *Let η be a set of active integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database. Let U be a consistent set of update actions that is relevant w.r.t. V.*

U is a global-dynamic repair of V by η iff U is a PMA repair of V achieving $\mathsf{C}(\eta)$ and $U \subseteq \bigcup_{r\in\eta} \mathsf{R}(r)$.

Proof. The left-to-right direction is trivial by the definition of global-dynamic repairs, whereas the right-to-left direction follows immediately by Theorem 2.6. □

But ultimately the choice between dynamic and global-dynamic repairs is traced back to what the answer should be regarding the repairing or not of a database in all of these cases. The former use a more restrictive procedure that makes it more *local*, while the latter do not.

Remark 2. *In dynamic weak repairs and dynamic repairs we can use the* $\bigcup_{p\in\mathbb{P}_{\mathsf{R}(r)}} \mathsf{toggle}(p)$ *program in the place of* $\bigcup_{\alpha\in\mathsf{R}(r)} \alpha$ *in* π_r *without any change in the dynamic behavior of the repairs. In the case of global-dynamic weak repairs, at first sight the programs:*

$$\textbf{while } \neg\left(\bigwedge \mathsf{C}(\eta)\right) \textbf{ do } \left(\bigcup_{\substack{r\in\eta\\ \alpha\in\mathsf{R}(r)}} \alpha\right) \quad \textit{and} \quad \textbf{while } \neg\left(\bigwedge \mathsf{C}(\eta)\right) \textbf{ do } \left(\bigcup_{\substack{r\in\eta\\ p\in\mathbb{P}_{\mathsf{R}(r)}}} \mathsf{toggle}(p)\right)$$

seem to once again bring the same results. So why not use the second definition which abbreviates:

$$\left(\neg\left(\bigwedge \mathsf{C}(\eta)\right)?\ ;\bigcup_{\substack{r\in\eta\\ p\in\mathbb{P}_{\mathsf{R}(r)}}} \mathsf{toggle}(p)\right)^*;\left(\bigwedge \mathsf{C}(\eta)\right)?$$

and which is highly reminiscent of the program $\mathsf{weakRepair}(\mathsf{C}(\eta))$*, showing its close relationship with weak and PMA repairs. It should be clarified why* $\mathsf{toggle}(p)$ *doesn't work anymore. The reason is that* toggling *a propositional variable in this case is not the same as choosing the respective update action, as* $\mathsf{toggle}(p)$ *can bring the opposite results. We can see this when* $V = \emptyset$ *and* $\eta = \left\{\langle \neg p \vee q, \{p \leftarrow \bot\}\rangle,\ \langle \neg q \vee p, \{q \leftarrow \bot\}\rangle,\ \langle r, \{r \leftarrow \top\}\rangle\right\}$*. A global-dynamic weak repair of V by η using the alternative definition with* $\mathsf{toggle}(p)$ *is the set of update actions* $\{p \leftarrow \top, q \leftarrow \top, r \leftarrow \top\}$*, which is obviously absurd.*

As already mentioned, this does not happen with dynamic weak repairs and dynamic repairs. It is another aspect of their "strict" and *local* nature, as they check if a clause needs repairing before toggling any propositional variable, thus making any update action chosen to be exactly the intended one from the set of preferred ones.

2.6 Complexity of Dynamic Repairs

In this section we provide tight complexity bounds for the problems of existence of a dynamic weak repair and a dynamic repair. It is known that deciding the existence of a repair is NP-complete for weak repairs, PMA repairs and founded weak repairs, while it is Σ_P^2-complete for founded repairs, justified weak repairs and justified repairs [Caroprese and Truszczynski, 2011]. As we will see, the same problem proves to be more difficult for dynamic weak repairs and dynamic repairs: deciding their existence is PSPACE-complete. For the lower bound (hardness) we provide a reduction from the problem of checking whether a fully quantified boolean formula is true, whereas for the upper bound (membership) a reduction to the model checking problem of DL-PA will suffice. The result follows from the fact that checking whether a fully quantified boolean formula is true and DL-PA model checking are both PSPACE-complete problems [Stockmeyer and Meyer, 1973, Balbiani et al., 2014].

2.6.1 Lower Complexity Bound

In order to show that the existence of dynamic weak repairs and dynamic repairs is PSPACE-hard we will provide a reduction from the following problem: given a fully quantified boolean formula G, decide whether G is true. We suppose w.l.o.g. that G is in prenex normal form, with the variables in the prefix being all different and the matrix containing only the boolean connectives $\neg$ and $\wedge$. Let $\mathrm{subf}(G)$ be the set comprising all the subformulas of G and let $\mathrm{subf}^v(G)$ be the set comprising all the variables in G. Furthermore, let $\mathrm{subf}^\neg(G)$, $\mathrm{subf}^\wedge(G)$, $\mathrm{subf}^\exists(G)$ and $\mathrm{subf}^\forall(G)$ be the sets comprising all subformulas of G that have the form $\neg A$, $A \wedge B$, $\exists x.A$ and $\forall x.A$, respectively. It is easy to see that $\mathrm{subf}(G) = \mathrm{subf}^v(G) \cup \mathrm{subf}^\neg(G) \cup \mathrm{subf}^\wedge(G) \cup \mathrm{subf}^\exists(G) \cup \mathrm{subf}^\forall(G)$. Next, we define the set $\mathbb{P}_G$ of propositional variables to be composed of:

- all x, $x?$ and $x!$ such that $x \in \mathrm{subf}^v(G)$
- $A+?, A-?, A+!, A-!$ and $A\pm$ for each $A \in \mathrm{subf}(G)$

Intuitively, the elements of $\mathbb{P}_G$ play the following roles: x stores the truth value of x in

G, $x?$ indicates that a value for x must be chosen and $x!$ indicates that the value for x has been chosen. Similarly, $A{+}?$ indicates that we check if A is true and $A{+}!$ indicates that A has been proved to be true. The same goes for $A{-}?$ and $A{-}!$, for checking and proving that A is false. Last but not least, $A\pm$ is used to record the intermediate state whenever we need to verify that A is true for both x and $\neg x$ (viz. when we have already checked the case where x is true and still have to check the case where x is false).

The idea now is to start from the initial formula G and to compute whether it is true by asking whether there is a dynamic procedure that repairs the database $\{G{+}?\}$ under a set of active constraints. We will define this set by first assigning a set of active constraints to each $A \in \mathrm{subf}(G)$, indicating the required steps for checking whether or not A can be proved true or false, and taking their union. For each A, the goal is to reach a state satisfying $\neg A{+}? \wedge A{+}!$ if we want to prove that A is true and $\neg A{-}? \wedge A{-}!$ if we want to prove that A is false. Indeed, each $A \in \mathrm{subf}(G)$ is associated with a set of active constraints which repair any database satisfying $A{+}? \wedge \neg A{+}!$ (respectively $A{-}? \wedge \neg A{-}!$) to one satisfying $\neg A{+}? \wedge A{+}!$ (respectively $\neg A{-}? \wedge A{-}!$). So, starting from the database $\{G{+}?\}$ which satisfies $G{+}? \wedge \neg G{+}!$, if there is a successful dynamic repair procedure using these sets of active constraints then $\neg G{+}? \wedge G{+}!$ will be reached and the initial formula G will be proved to be true. On the other hand, if there is no dynamic weak repair of $\{G{+}?\}$ by these active constraints then G will be false.

In the following we define the sets of active integrity constraints for each $A \in \mathrm{subf}(G)$ that encode the truth conditions that are used in the evaluation of G. The literals underlined highlight the differences between the static constraints while the rightmost column explains the action taken. A small lemma after each definition proves why the respective case works, where (1) $\mathsf{IndHyp}(A, V)^+$ and (2) $\mathsf{IndHyp}(A, V)^-$ denote the induction hypotheses for $A \in \mathrm{subf}(G)$ and $V \subseteq \mathrm{subf}^v(G)$, i.e.:

$(1) \coloneqq$ the set $\{A{+}? \leftarrow \bot, A{+}! \leftarrow \top\}$ is a dynamic weak repair of $\{A{+}?\}$ iff $V \models A$

$(2) \coloneqq$ the set $\{A{-}? \leftarrow \bot, A{-}! \leftarrow \top\}$ is a dynamic weak repair of $\{A{-}?\}$ iff $V \not\models A$

• When $A = \exists x.B$ we set:

a_1:	$\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee B{+}! \vee x? \vee x! \vee x, \{x? \leftarrow \top\}\rangle$	ask for a truth value for x
a_2:	$\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee B{+}! \vee \underline{\neg x?} \vee x! \vee x, \{x \leftarrow \top, x! \leftarrow \top\}\rangle$	toggle x or set it to false
a_3:	$\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee B{+}! \vee \neg x? \vee x! \vee \underline{\neg x}, \{x! \leftarrow \top\}\rangle$	set x to true
a_4:	$\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee B{+}! \vee \neg x? \vee \underline{\neg x!}, \{x? \leftarrow \bot\}\rangle$	end the choice of x
a_5:	$\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee B{+}! \vee \underline{x?} \vee \neg x!, \{B{+}? \leftarrow \top\}\rangle$	ask for B to be true
a_6:	$\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee \underline{\neg B{+}!} \vee x? \vee \neg x!, \{A{+}! \leftarrow \top\}\rangle$	A is now proved true
a_7:	$\langle \neg A{+}? \vee \underline{\neg A{+}!} \vee B{+}? \vee \neg B{+}! \vee x? \vee \neg x!, \{x! \leftarrow \bot\}\rangle$	free x
a_8:	$\langle \neg A{+}? \vee \neg A{+}! \vee B{+}? \vee \neg B{+}! \vee x? \vee \underline{x!}, \{B{+}! \leftarrow \bot\}\rangle$	remove the result for B
a_9:	$\langle \neg A{+}? \vee \neg A{+}! \vee B{+}? \vee \underline{B{+}!} \vee x? \vee x! \vee \neg x, \{x \leftarrow \bot\}\rangle$	remove x if it is set to true
a_{10}:	$\langle \neg A{+}? \vee \neg A{+}! \vee B{+}? \vee B{+}! \vee x? \vee x! \vee \underline{x}, \{A{+}? \leftarrow \bot\}\rangle$	end the request for A

Lemma 2.4. *Let $A \in \mathrm{subf}^{\exists}(G)$ and $V \subseteq \mathrm{subf}^{v}(G)$. Furthermore, consider that the induction hypothesis holds for B, where $A = \exists x.B$, i.e., at least one of $\mathsf{IndHyp}(B, V \setminus \{x\})^{+}$ or $\mathsf{IndHyp}(B, V \cup \{x\})^{+}$ holds. Then:*

$$V \models A \textit{ iff there is a dynamic weak repair of } V \cup \{A{+}?\} \textit{ by } \{a_1, \ldots, a_{10}\}$$

Proof. Consider $A = \exists x.B$, the database V and the set of active constraints $\eta = \{a_1, \ldots, a_{10}\}$. If $V \models A$ then either $V \setminus \{x\} \models B$ or $V \cup \{x\} \models B$ (1). Starting from $V \cup \{A{+}?\}$ then, the active constraints a_1 to a_4 will make sure that the truth value of x is set, i.e., x is in the database or $\neg x$ is in the database, via the auxiliary variables $x?$ and $x!$. The active constraint a_5 then will ask if B is satisfied given the truth value of x that is already set in the database. This will be checked by the active constraints that are assigned to B. By (1) then, since B is indeed satisfied for at least one of the truth values of x, choosing the correct one will result in the induction hypothesis removing the variable $B{+}?$ from the database and adding the variable $B{+}!$ to the database. Thus, the constraint a_6 will be violated and it is recorded that A is proved to be satisfied via the variable $A{+}!$. The active constraints a_7 to a_{10} then remove all the auxiliary variables in case A needs to be checked again.[7] Given these we have that for at least one value of x the procedure is ending and a dynamic weak repair of $V \cup \{A{+}?\}$ by η exists. On the other hand, if a dynamic weak repair of $V \cup \{A{+}?\}$ by η exists then B has been proved to be satisfied for at least one truth value of x. This is done in the transition from a_5 to a_6 where, by adding $B{+}?$ to the database, the active constraints assigned to B repair the database into one satisfying $\neg B{+}? \wedge B{+}!$. By induction hypothesis then, this means that

[7] This may happen if, for instance, $A' = \forall x.A$ and the truth value for A needs to be checked twice while checking if A' is true.

either $V \setminus \{x\} \models B$ or $V \cup \{x\} \models B$ (according to which of a_2 or a_3 took place last) which means that $V \models A$. □

• When $A = \forall x.B$ we set:

b_1:	$\langle \neg A+? \vee A+! \vee A\pm \vee B+? \vee B+! \vee x, \{x \leftarrow \top\}\rangle$	make sure x is true
b_2:	$\langle \neg A+? \vee A+! \vee A\pm \vee B+? \vee B+! \vee \underline{\neg x}, \{B+? \leftarrow \top\}\rangle$	ask for B to be true, x being true
b_3:	$\langle \neg A+? \vee A+! \vee A\pm \vee B+? \vee \underline{\neg B+!} \vee \neg x, \{A\pm \leftarrow \top\}\rangle$	record the intermediate state
b_4:	$\langle \neg A+? \vee A+! \vee \underline{\neg A\pm} \vee B+? \vee \neg B+! \vee \neg x, \{B+! \leftarrow \bot\}\rangle$	x being true, erase the result for B
b_5:	$\langle \neg A+? \vee A+! \vee \neg A\pm \vee B+? \vee \underline{B+!} \vee \neg x, \{x \leftarrow \bot\}\rangle$	now make x false
b_6:	$\langle \neg A+? \vee A+! \vee \neg A\pm \vee B+? \vee B+! \vee \underline{x}, \{B+? \leftarrow \top\}\rangle$	ask for B to be true, x being false
b_7:	$\langle \neg A+? \vee A+! \vee \neg A\pm \vee B+? \vee \underline{\neg B+!} \vee x, \{A+! \leftarrow \top\}\rangle$	A is now proved true
b_8:	$\langle \neg A+? \vee \underline{\neg A+!} \vee \neg A\pm \vee B+? \vee \neg B+! \vee x, \{B+! \leftarrow \bot\}\rangle$	remove the result for B
b_9:	$\langle \neg A+? \vee \neg A+! \vee \neg A\pm \vee B+? \vee \underline{B+!} \vee x, \{A\pm \leftarrow \bot\}\rangle$	remove the intermediate state
b_{10}:	$\langle \neg A+? \vee \neg A+! \vee \underline{A\pm} \vee B+? \vee B+! \vee x, \{A+? \leftarrow \bot\}\rangle$	end the request for A

Lemma 2.5. *Let $A \in \mathrm{subf}^{\forall}(G)$ and $V \subseteq \mathrm{subf}^{v}(G)$. Furthermore, consider that the induction hypothesis holds for B, where $A = \forall x.B$, i.e., both of $\mathsf{IndHyp}(B, V \setminus \{x\})^+$ and $\mathsf{IndHyp}(B, V \cup \{x\})^+$ hold. Then:*

$$V \models A \textit{ iff there is a dynamic weak repair of } V \cup \{A+?\} \textit{ by } \{b_1, \ldots, b_{10}\}$$

Proof. Consider $A = \forall x.B$, the database V and the set of active constraints $\eta = \{b_1, \ldots, b_{10}\}$. If $V \models A$ then $V \setminus \{x\} \models B$ and $V \cup \{x\} \models B$ (1). Starting from $V \cup \{A+?\}$ then, as a first step we make sure that x is true through b_1. We then check, in a similar manner as before, that B is indeed satisfied for both truth values of x. This is done through the active constraints b_2 to b_6. The only difference here is that we have to record an intermediate state via the variable $A\pm$ when switching from x being true to x being false. The reason is that, if this intermediate state is not recorded through b_3, a database which doesn't include the variable x will violate both b_1 and b_6 and may jump to the latter constraint, thus not checking if B is satisfied when x is true. The rest is the same: the auxiliary variables are removed from the database before ending the check in b_{10}. By (1) then, since B is indeed satisfied for both truth values of x, and b_3 as well as b_7 are both violated, using the induction hypothesis we have that the procedure is ending and a dynamic weak repair of $V \cup \{A+?\}$ by η exists. On the other hand, if a dynamic weak repair of $V \cup \{A+?\}$ by η exists then, similarly with before, B has been proved to be satisfied for both truth values of x: for x in the transition from b_2 to b_3 and for $\neg x$ and in the transition from b_6 to b_7.

By induction hypothesis then, this means that $V \setminus \{x\} \models B$ and $V \cup \{x\} \models B$ and thus $V \models A$. □

• When $A = \neg B$ we set:

c_1: $\langle \neg A{+}? \vee \quad A{+}! \vee B{-}? \vee \quad B{-}!, \{B{-}? \leftarrow \top\}\rangle$ ask for B to be false
c_2: $\langle \neg A{+}? \vee \quad A{+}! \vee B{-}? \vee \underline{\neg B{-}!}, \{A{+}! \leftarrow \top\}\rangle$ A is now proved true
c_3: $\langle \neg A{+}? \vee \underline{\neg A{+}!} \vee B{-}? \vee \neg B{-}!, \{B{-}! \leftarrow \bot\}\rangle$ remove the result for B
c_4: $\langle \neg A{+}? \vee \neg A{+}! \vee B{-}? \vee \quad \underline{B{-}!}, \{A{+}? \leftarrow \bot\}\rangle$ end the request for A

c_5: $\langle \neg A{-}? \vee \quad A{-}! \vee B{+}? \vee \quad B{+}!, \{B{+}? \leftarrow \top\}\rangle$ ask for B to be true
c_6: $\langle \neg A{-}? \vee \quad A{-}! \vee B{+}? \vee \underline{\neg B{+}!}, \{A{-}! \leftarrow \top\}\rangle$ A is now proved false
c_7: $\langle \neg A{-}? \vee \underline{\neg A{-}!} \vee B{+}? \vee \neg B{+}!, \{B{+}! \leftarrow \bot\}\rangle$ remove the result for B
c_8: $\langle \neg A{-}? \vee \neg A{-}! \vee B{+}? \vee \quad \underline{B{+}!}, \{A{-}? \leftarrow \bot\}\rangle$ end the request for A

Lemma 2.6. *Let $A \in \mathrm{subf}^{\neg}(G)$ and $V \subseteq \mathrm{subf}^{v}(G)$. Furthermore, consider that the induction hypothesis holds for B, where $A = \neg B$, i.e., both of $\mathsf{IndHyp}(B, V)^{+}$ and $\mathsf{IndHyp}(B, V)^{-}$ hold. Then:*

- *$V \models A$ iff there is a dynamic weak repair of $V \cup \{A{+}?\}$ by $\{c_1, c_2, c_3, c_4\}$*
- *$V \not\models A$ iff there is a dynamic weak repair of $V \cup \{A{-}?\}$ by $\{c_5, c_6, c_7, c_8\}$*

Proof. For the first case, consider $A = \neg B$, the database V and the set of active constraints $\eta = \{c_1, c_2, c_3, c_4\}$. Clearly, if $V \models A$ then $V \not\models B$ (1). So starting from $V \cup \{A{+}?\}$, the active constraint c_1 will ask for B to be false. This will be checked by the active constraints that are assigned to B. By (1) then, since B is indeed false in V, the (second) induction hypothesis will remove the variable $B{-}?$ from the database and will add the variable $B{-}!$ in the database. Thus, the constraint c_2 will be violated and it is recorded that A is proved to be satisfied via the variable $A{+}!$. The active constraint c_3 then removes the variable $B{-}!$ in case A needs to be checked again. Finally, c_4 removes the variable $A{+}?$, thus ending the check for A. Given these observations, we have that the procedure is ending and a dynamic weak repair of $V \cup \{A{+}?\}$ by η exists. On the other hand, if a dynamic weak repair of $V \cup \{A{+}?\}$ by η exists then B has been proved to be false in V, i.e., $V \not\models B$, in the transition from c_1 to c_2 and the (second) induction hypothesis. Thus, $V \models A$.

For the second case, the argument is similar (using the first induction hypothesis). □

• When $A = B \wedge C$ we set:

d_1: $\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee C{+}? \vee B{+}! \vee C{+}!, \{B{+}? \leftarrow \top\}\rangle$ ask for B to be true
d_2: $\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee C{+}? \vee \underline{\neg B{+}!} \vee C{+}!, \{C{+}? \leftarrow \top\}\rangle$ ask for C to be true
d_3: $\langle \neg A{+}? \vee A{+}! \vee B{+}? \vee C{+}? \vee \neg B{+}! \vee \underline{\neg C{+}!}, \{A{+}! \leftarrow \top\}\rangle$ A is now proved true
d_4: $\langle \neg A{+}? \vee \underline{\neg A{+}!} \vee B{+}? \vee C{+}? \vee \neg B{+}! \vee \neg C{+}!, \{B{+}! \leftarrow \bot\}\rangle$ remove the result for B
d_5: $\langle \neg A{+}? \vee \neg A{+}! \vee B{+}? \vee C{+}? \vee \underline{B{+}!} \vee \neg C{+}!, \{C{+}! \leftarrow \bot\}\rangle$ remove the result for C
d_6: $\langle \neg A{+}? \vee \neg A{+}! \vee B{+}? \vee C{+}? \vee B{+}! \vee \underline{C{+}!}, \{A{+}? \leftarrow \bot\}\rangle$ end the request for A

d_7: $\langle \neg A{-}? \vee A{-}! \vee B{-}? \vee C{-}? \vee B{-}! \vee C{-}!, \{B{-}? \leftarrow \top, C{-}? \leftarrow \top\}\rangle$ ask for B or C to be false
d_8: $\langle \neg A{-}? \vee A{-}! \vee B{-}? \vee C{-}? \vee \underline{\neg B{-}!} \vee C{-}!, \{A{-}? \leftarrow \top\}\rangle$ B being false, A is proved false
d_9: $\langle \neg A{-}? \vee A{-}! \vee B{-}? \vee C{-}? \vee B{-}! \vee \underline{\neg C{-}!}, \{A{-}? \leftarrow \top\}\rangle$ C being false, A is proved false
d_{10}: $\langle \neg A{-}? \vee \underline{\neg A{-}!} \vee B{-}? \vee C{-}? \vee \neg B{-}! \vee C{-}!, \{B{-}! \leftarrow \bot\}\rangle$ remove the result for B
d_{11}: $\langle \neg A{-}? \vee \neg A{-}! \vee B{-}? \vee C{-}? \vee \underline{B{-}!} \vee \neg C{-}!, \{C{-}! \leftarrow \bot\}\rangle$ remove the result for C
d_{12}: $\langle \neg A{-}? \vee \neg A{-}! \vee B{-}? \vee C{-}? \vee B{-}! \vee \underline{C{-}!}, \{A{-}? \leftarrow \bot\}\rangle$ end the request for A

Lemma 2.7. *Let $A \in \mathrm{subf}^{\wedge}(G)$ and $V \subseteq \mathrm{subf}^{v}(G)$. Furthermore, consider that the induction hypothesis holds for B and C, where $A = B \wedge C$, i.e., both of $\mathsf{IndHyp}(B, V)^{+}$ and $\mathsf{IndHyp}(C, V)^{+}$ hold whereas at least one of $\mathsf{IndHyp}(B, V)^{-}$ or $\mathsf{IndHyp}(C, V)^{-}$ holds. Then:*

- *$V \models A$ iff there is a dynamic weak repair of $V \cup \{A{+}?\}$ by $\{d_1, \ldots, d_6\}$*
- *$V \not\models A$ iff there is a dynamic weak repair of $V \cup \{A{-}?\}$ by $\{d_7, \ldots, d_{12}\}$*

Proof. It can be checked that the procedure is similar to the one of Lemma 2.6, with the only points of interest being that in the first case we have to check through d_1 and d_2 that both B and C are satisfied by V before recording that A has been proved to be satisfied by V via the variable $A{+}!$. In the second case, since only one of B or C is needed to be proved false in V for A to be false in V, we distinguish these cases through the constraints d_8 and d_9. □

• When $A = x$ we set:

e_1: $\langle \neg A{+}? \vee A{+}! \vee \neg A, \{A{+}! \leftarrow \top\}\rangle$ A is proved true
e_2: $\langle \neg A{+}? \vee \underline{\neg A{+}!} \vee \neg A, \{A{+}? \leftarrow \bot\}\rangle$ end the request for A

e_3: $\langle \neg A{-}? \vee A{-}! \vee A, \{A{-}! \leftarrow \top\}\rangle$ A is proved false
e_4: $\langle \neg A{-}? \vee \underline{\neg A{-}!} \vee A, \{A{-}? \leftarrow \bot\}\rangle$ end the request for A

Lemma 2.8. *Let $A \in \mathrm{subf}^{v}(G)$ and $V \subseteq \mathrm{subf}^{v}(G)$. Then:*

- *$V \models A$ iff there is a dynamic weak repair of $V \cup \{A{+}?\}$ by $\{e_1, e_2\}$*

- *$V \not\models A$ iff there is a dynamic weak repair of $V \cup \{A-?\}$ by $\{e_3, e_4\}$*

Proof. For the first case, consider the database V and the set of active constraints $\eta = \{e_1, e_2\}$. Since $A \in \mathrm{subf}^v(G)$ and $V \subseteq \mathrm{subf}^v(G)$, $V \models A$ means that $A \in V$. Starting from $V \cup \{A+?\}$ then, the active constraint e_1 will confirm that A is in the database, whereas through e_2 the check for A will stop. Given these, we have that the procedure is ending and a dynamic weak repair of $V \cup \{A+?\}$ by η exists. On the other hand, if a dynamic weak repair of $V \cup \{A+?\}$ by η exists then e_1 was violated and consequently $A \in V$, which means that $V \models A$. Similarly for the second case, where $V \not\models A$ means that $A \notin V$ and e_3 is violated by $V \cup \{A-?\}$. □

Finally, consider $g = \langle G+? \vee \neg G+!, \{G+! \leftarrow \bot\}\rangle$ which is used in the final step to ensure minimality of the repair procedure, i.e., in order to ensure that only the update action $G+? \leftarrow \bot$ survives at the end. We then define the set of active constraints η_G to be the following:

$$\bigcup_{A \in \mathrm{subf}^{\exists}(G)} \{a_1, \ldots, a_{10}\} \cup \bigcup_{A \in \mathrm{subf}^{\forall}(G)} \{b_1, \ldots, b_{10}\} \cup \bigcup_{A \in \mathrm{subf}^{\neg}(G)} \{c_1, \ldots, c_8\} \cup \bigcup_{A \in \mathrm{subf}^{\wedge}(G)} \{d_1, \ldots, d_{12}\} \cup \bigcup_{A \in \mathrm{subf}^{v}(G)} \{e_1, \ldots, e_4\} \cup g$$

Then we have the following lemma, proposition and theorem.

Lemma 2.9. *If there is a dynamic weak repair of $\{G+?\}$ by η_G then there is also a dynamic repair of $\{G+?\}$ by η_G.*

Proof. By the construction of η_G and the active constraint g, the set $\{G+? \leftarrow \bot\}$ will always be the only dynamic weak repair of $\{G+?\}$ by η_G and, since $\emptyset$ is not a weak repair of $\{G+?\}$ by η_G, it is also a PMA repair. So if a dynamic weak repair of $\{G+?\}$ by η_G exists, it will be the set $\{G+? \leftarrow \bot\}$ which is also a dynamic repair of $\{G+?\}$ by η_G. □

Proposition 2.5. *Let G be a fully quantified boolean formula in prenex normal form, with the variables in the prefix being all different and the matrix containing only the boolean connectives $\neg$ and $\wedge$. Then:*

- *G is true iff there is a dynamic weak repair of $\{G+?\}$ by η_G*
- *G is true iff there is a dynamic repair of $\{G+?\}$ by η_G*

Proof. The first result is immediate by the construction of η_G and Lemmas 2.4, 2.5, 2.6, 2.7 and 2.8. The second result follows from the first one and Lemma 2.9. □

Theorem 2.7. *The problems of existence of a dynamic weak repair and a dynamic repair are both PSPACE-hard.*

Proof. It is known that checking whether a fully quantified boolean formula is true is a PSPACE-complete problem. The result follows from Proposition 2.5 and the fact that, given a formula G, the cardinality of the set η_G is linear in the length of G. □

2.6.2 Upper Complexity Bound

Next, in order to show that deciding the existence of dynamic weak repairs and dynamic repairs is in PSPACE we just need to reduce the problem into the model checking problem of DL-PA, since the latter is known to be PSPACE-complete [Balbiani et al., 2014]. The reduction is easy and is based on the following proposition.

Proposition 2.6. *Let η be a set of active integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database.*

- *A dynamic weak repair of V by η exists iff $V \models \langle \pi_\eta \rangle \top$, where π_η is the program:*

$$\textbf{while } \neg \Big(\bigwedge \mathsf{C}(\eta) \Big) \textbf{ do } \Big(\bigcup_{r \in \eta} \pi_r \Big)$$

- *A dynamic repair of V by η exists iff $V \models \langle \pi_\eta \rangle \top$, where π_η is the program:*

$$\textbf{while } \neg \Big(\bigwedge \mathsf{C}(\eta) \Big) \textbf{ do } \Big(\bigcup_{r \in \eta} \pi_r^{\pm} \Big) ; \mathsf{Minimal}(\mathsf{C}(\eta))? ; {\mathbb{P}_C}^{\pm} \leftarrow \bot$$

Proof. In both cases, for the left to right direction consider that a dynamic weak repair (respectively, dynamic repair) of V by η exists. By Definition 2.9 and Theorem 2.5, this means that there exists a set of update actions U such that U is relevant w.r.t. V and $\langle V, V \diamond U \rangle \in ||\pi_\eta||$. Since $V \diamond U \models \top$, the result follows.

Again in both cases, for the right to left direction let $V \models \langle \pi_\eta \rangle \top$. This means that there exists a V' such that $\langle V, V' \rangle \in ||\pi_\eta||$. It is easy to see that, if we set $U = \{p \leftarrow \top \ :$

$p \in V'$ and $p \notin V\} \cup \{p \leftarrow \bot \ : \ p \in V$ and $p \notin V'\}$ then $V' = V \diamond U$ and U is relevant w.r.t. V. So, in other words, there exists a U such that U is relevant w.r.t. V and $\langle V, V \diamond U \rangle \in ||\pi_\eta||$. By Definition 2.9 and Theorem 2.5 then, this means that there exists a dynamic weak repair (respectively, dynamic repair) of V by η. □

The theorem for membership then follows immediately.

Theorem 2.8. *The problems of existence of a dynamic weak repair and a dynamic repair are both in PSPACE.*

Proof. It is known that model checking in DL-PA is a PSPACE-complete problem. The result follows from Proposition 2.6 and the fact that, given a set of active integrity constraints η, the length of the program π_η is linear in the size of the set $\mathsf{C}(\eta)$. □

Using Theorems 2.7 and 2.8 then, we have the following corollary.

Corollary 2.1. *The problems of existence of a dynamic weak repair and a dynamic repair are both PSPACE-complete.*

2.7 History-Based Repairs

In this section we would like to discuss *history-based repairs*, an extension of the repairs seen so far taking into account databases with *history*. What we mean by *history* is the consistent set of transactions that took place from the last time the database satisfied the integrity constraints up until its current form. So let's say that apart from the initial database, we are also provided with a consistent set of update actions: these are what we refer to as *history*, the extra information of the route taken from an earlier point in time (more specifically, the last time the integrity constraints were satisfied) until the current state of affairs.

So given this extra information, how should we make use of it? A starting point would be to make sure that the update actions which took place in order to reach the current database will not appear again in the future. Imagine, for instance, that we are

provided with the database V, a set of active integrity constraints η and an update action that took place in order to reach it, $p \leftarrow \bot$. If there exist two repairs of V by η, namely $U_1 = \{p \leftarrow \top\}$ and $U_2 = \{q \leftarrow \top\}$, then we would like to disregard U_1 as it would repair V by adding p and V would return to an "earlier state" (from which it was updated by removing p) thus violating the 'priority of the new information' principle that was widely considered in the update literature [Katsuno and Mendelzon, 1992]. Furthermore, we should consider what happens in the case where, although repairs exist, there is no repair that updates the database without returning it to an "earlier state". Should we make use of them and disregard the given history or not? Intuitively, the repair actions of active integrity constraints are of the highest priority when repairing a database, whereas the aforementioned *history* is based on a set of update actions which was used to repair a previous database into the current one, but can be undone if needed. We can see this in the previous example as well, where if U_1 was the only repair of V by η then it should be applied regardless of $p \leftarrow \bot$ being used to reach V.

But perhaps a more concrete example is the following, based on the "an employee cannot be in 2 departments" constraint: let $\mathbb{P} = \{d_1, d_2\}$, where d_1 and d_2 denote departments 1 and 2 respectively, and the integrity constraint $r = \langle \neg d_1 \vee \neg d_2, \{d_1 \leftarrow \bot, d_2 \leftarrow \bot\}\rangle$ which says that no employee should work in both departments at the same time; if this is the case, then they should be removed from either one, without any specific preference. Assume now that $H = \langle \{d_1, d_2\}, \{d_1 \leftarrow \top\}\rangle$ is our history-based database, where $\{d_1, d_2\}$ is our actual database V (saying that there is someone working on both departments) and the set $\{d_1 \leftarrow \top\}$ represents the history, i.e., that the last department which they joined was d_1. In this case, we would prefer the repairing of V by $\{d_2 \leftarrow \bot\}$, instead of $\{d_1 \leftarrow \bot\}$, considering the latest action of putting the specific employee recently in department 1 to be of higher priority. This is done by repairing V by $r_H = \langle \neg d_1 \vee \neg d_2, \{d_2 \leftarrow \bot\}\rangle$ instead of r and actually disregarding the "preferred" update action $d_1 \leftarrow \bot$ in $\mathsf{R}(r)$ which conflicts with the provided history.

With that in mind, we call $H = \langle V, U\rangle$ a history-based database when $V \subseteq \mathbb{P}$ is a database and $U \subseteq \mathbb{U}$ a consistent set of update actions such that there is $V' \subseteq \mathbb{P}$ with $V' \diamond U = V$. We also define as U^{-1} the set comprising the opposite update actions in U: $U^{-1} = \{p \leftarrow \bot : p \leftarrow \top \in U\} \cup \{p \leftarrow \top : p \leftarrow \bot \in U\}$.

According to what has been said so far, we would like U' to be a repair of a history-based database $H = \langle V, U\rangle$ by a set of active integrity constraints η, only if $U' \cap U^{-1} = \emptyset$. In order for this to happen we have to repair V by η_H instead of η, where η_H has:

$$r_H = \left\langle \mathsf{C}(r), \mathsf{R}(r) \setminus U^{-1} \right\rangle \quad \text{, for } r \in \eta$$

In this way we disregard update actions which have the risk of returning our current database to a previous state and give priority to the new ones. As already mentioned, by choosing to ignore update actions based on the history U that we have, we may risk reducing a set $\mathsf{R}(r)$ to be empty for some $r \in \eta$, thus leading to no repairs occurring. In this case, choosing to not take U into consideration is the only option and we return to the repairing of V by η, instead of H by η. The following example highlights everything that's been said so far.

Example 2.13. *Let $H = \langle\{d_1, d_2\}, \{d_1 \leftarrow \top\}\rangle$ and $\eta_1 = \langle \neg d_1 \vee \neg d_2, \{d_1 \leftarrow \bot, d_2 \leftarrow \bot\}\rangle$. Both $U_1 = \{d_1 \leftarrow \bot\}$ and $U_2 = \{d_2 \leftarrow \bot\}$ are founded, justified, dynamic and global-dynamic repairs of $V = \{d_1, d_2\}$ by η_1. Only the second is a founded, justified, dynamic and global-dynamic repair of H by η_1. Similarly, consider $\eta_2 = \langle \neg d_1 \vee \neg d_2, \{d_1 \leftarrow \bot\}\rangle$. Then $U_1 = \{d_1 \leftarrow \bot\}$ is a founded, justified, dynamic and global-dynamic repair of $V = \{d_1, d_2\}$ by η_2. There are no repairs of H by η_2, however, making us reduce H to just V and use U_1 once again.*

Finally, let us define:

$$\mathsf{weakRepairH}(\mathsf{C}(\eta), U) = \left(\bigcup_{p \in A} \mathsf{toggle}(p)\right)^* ; \left(\bigwedge \mathsf{C}(\eta)\right)?$$

$$\left(\pi^{\pm}_{r,U}\right)' = \neg \mathsf{C}(r)\,? \,; \bigcup_{p \leftarrow \mathsf{X} \in B} \left(p \leftarrow \mathsf{X}; p^{\pm} \leftarrow \top\right) \qquad \text{and} \qquad \left(\pi^{\pm}_{r,U}\right)'' = \bigcup_{p \leftarrow \mathsf{X} \in B} \left(p \leftarrow \mathsf{X}; p^{\pm} \leftarrow \top\right)$$

where $A = \mathbb{P}_{\mathsf{C}(\eta) \setminus U^{-1}}$ (the set of the variables that occur in $\mathsf{C}(\eta)$ and not in U^{-1}) and $B = \mathsf{R}(r) \setminus U^{-1}$.

The next theorem characterises founded, justified, dynamic and global-dynamic history-based repairs in terms of DL-PA programs.

Theorem 2.9. *Let η be a set of active integrity constraints in the language of $\mathbb{P}$ and let $V \subseteq \mathbb{P}$ be a database (so no $p^{\pm}, p^{+}$ and p^{-} occurs in either of them). Let U and U' be*

consistent sets of update actions such that U is relevant w.r.t. V.

- *U is a founded repair of $H = \langle V, U' \rangle$ by η iff:*

$$\langle V,\, V \diamond U \rangle \in \Big\| \mathsf{weakRepairH}(\mathsf{C}(\eta), U'); \mathsf{Founded}(\eta)? \,;\, \mathsf{Minimal}(\mathsf{C}(\eta))? \,;\, {\mathbb{P}_C}^{\pm} \leftarrow \bot \Big\|$$

- *U is a justified repair of $H = \langle V, U' \rangle$ by η iff η is a set of standard active constraints and:*

$$\langle V,\, V \diamond U \rangle \in \Big\| \mathsf{weakRepairH}(\mathsf{C}(\eta), U'); \mathsf{Justified}(\eta)?; \mathsf{Minimal}(\mathsf{C}(\eta))?; \mathsf{ClearAll} \Big\|$$

- *U is a dynamic repair of $H = \langle V, U' \rangle$ by η iff:*

$$\langle V,\, V \diamond U \rangle \in \Big\| \mathbf{while}\ \neg \Big(\bigwedge \mathsf{C}(\eta) \Big)\ \mathbf{do}\ \Big(\bigcup_{r \in \eta} \Big(\pi^{\pm}_{r,U'} \Big)' \Big) \,;\, \mathsf{Minimal}\Big(\mathsf{C}(\eta) \Big)? \,;\, {\mathbb{P}_C}^{\pm} \leftarrow \bot \Big\|$$

- *U is a global-dynamic repair of $H = \langle V, U' \rangle$ by η iff:*

$$\langle V,\, V \diamond U \rangle \in \Big\| \mathbf{while}\ \neg \Big(\bigwedge \mathsf{C}(\eta) \Big)\ \mathbf{do}\ \Big(\bigcup_{r \in \eta} \Big(\pi^{\pm}_{r,U'} \Big)'' \Big) \,;\, \mathsf{Minimal}\Big(\mathsf{C}(\eta) \Big)? \,;\, {\mathbb{P}_C}^{\pm} \leftarrow \bot \Big\|$$

Proof. Note that the construction of all history-based repairs in DL-PA terms are identical to their counterparts in the previous sections, with the difference being in the choice of p's and update actions $p \leftarrow \mathsf{X}$ in the nondeterministic composition place of the programs. By removing the set U^{-1} from each nondeterministic choice imposed by the programs $\mathsf{weakRepairH}(\mathsf{C}(\eta), U)$, $\big(\pi^{\pm}_{r,U}\big)'$ and $\big(\pi^{\pm}_{r,U}\big)''$ we get exactly the active integrity constraints r_H (instead of r) needed in order to repair a history-based database by the corresponding set of active integrity constraints η_H. □

2.8 Conclusion

We have shown how several definitions of database repair via active integrity constraints can be expressed in DL-PA, including new proposals in terms of their iterated application. More specifically, we have introduced a new, dynamic way of handling database repair under a set of active integrity constraints and have shown some interesting properties (including advantages over other repairs, complexity) and alternatives (history-based

repairs), all through the use of the quite simple but expressive Dynamic Logic of Propositional Assignments DL-PA. This allows us to claim that DL-PA is a nice integrated framework for database updates: it not only provides operators $p \leftarrow \top$ of insertion and $p \leftarrow \bot$ of deletion and more generally sets U of such assignments that can be applied to a database V; it also provides a means to reason about the repair of the resulting $V \diamond U$ when some element of the set of integrity constraints is violated.

In the following, the program repair denotes one of the repair programs of Theorems 2.2, 2.3, 2.4, 2.5, 2.9, as well as Definitions 2.9 and 2.10. We can witness the aforementioned treatment of DL-PA as a means of reasoning between repairs in the following two instances:

- V' is a possible repair of the update of the database V by the insertion or deletion of p if and only if the couple $\langle V, V' \rangle$ belongs to the interpretation of the DL-PA programs $p \leftarrow \top$; repair or $p \leftarrow \bot$; repair respectively.
- The set of candidate repaired databases is the interpretation of the DL-PA formula $\langle \mathsf{repair}^{\mathsf{c}} \rangle \, \varphi_V$, where φ_V is a conjunction of literals describing V syntactically.

But beyond identifying possible repaired databases, what is even more interesting is that our programs repair also allow to solve decision problems. Some notable examples follow:

- We may check whether it is possible at all to repair V by model checking in DL-PA whether $V \models \langle \mathsf{repair} \rangle \top$.
- We can check whether there is a unique repair of V by model checking whether the set of databases V' such that $\langle V, V' \rangle \in ||\mathsf{repair}||$ is a singleton. This amounts to model check for each of the variables p occurring in the constraints whether $V \models [\mathsf{repair}]p \vee [\mathsf{repair}]\neg p$.
- We might as well wish to check possibility or unicity of the repairs independently of a specific database V. For instance, we can check whether η can repair any database by checking whether the formula $\langle \mathsf{repair} \rangle \top$ is DL-PA valid.
- A further interesting reasoning task is to check whether two sets of active constraints η_1 and η_2 are equivalent under a given semantics by checking whether

$||\mathsf{repair}_{\eta_1}|| = ||\mathsf{repair}_{\eta_2}||$.

All of the above demonstrate the usefulness of DL-PA as a logic dealing with database repair. The related decision problems also provide a hint of the variety of applications it provides. Furthermore, our way of handling active integrity constraints of the form $r = \langle \mathsf{C}(r), \mathsf{R}(r) \rangle$ allowed us to generalise the condition $\mathsf{C}(r)$ from clauses to arbitrary formulas (that could actually even be DL-PA formulas). This opens up two perspectives. First, our definition also covers revision programs [Caroprese and Truszczynski, 2011]; we leave it to future work to establish the exact relationship. Second, we could further generalise the action $\mathsf{R}(r)$ from a set of update actions to arbitrary DL-PA programs. Dynamic repairs would then still make sense, while it is not clear how founded and justified repairs would have to be defined.

Last but not least, although we have argued that there are real world problems, such as the one in the introductory Section 2.1, where dynamic repairs are preferable over founded or justified repairs, we did see that this comes at a cost: the computational complexity of dynamic repairs is higher. We leave it to future work to explore possible avenues of improving this dynamic behavior, especially in terms of computational resources. A possible extension to a first-order setting is also a good avenue for future research, seeing that the nature of the procedure is independent of the propositional setting that we worked on and that it could easily be adapted on higher level formalisms. We immediately take the first steps and venture into the setting of *Description Logics* which lie between propositional and first-order logic.

CHAPTER 3

Repairing ABoxes via Active TBoxes: a Syntactic Approach

Contents

3.1 Introduction

As we have seen so far, the integrity constraints of the database literature 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 [Lembo et al., 2010, Bienvenu et al., 2014, Bienvenu et al., 2016].

In this chapter, we integrate the idea behind active integrity constraints to the TBoxes of Description Logic and extend the TBox axioms with *preferred* update actions. The

resulting extended TBoxes are called *active* TBoxes in accordance with the nomenclature of *active* constraints. This extension of TBox axioms is achieved by means of the operations $\mathsf{add}(A)$ and $\mathsf{remove}(A)$, where A is an atomic concept. The intuition behind $\mathsf{add}(A)$ and $\mathsf{remove}(A)$ is similar to the one we have seen for $\mathcal{AIC}$s, i.e., whenever an inclusion axiom of the form $C \sqcap D \sqsubseteq \bot$ appears inside the TBox, for literals[1] C and D, then the TBox should also indicate the preferred way that this should be repaired when the axiom is violated in an ABox. While the problem is easier when an assertion of the form $a : A \sqcap B$ has to be updated to either the assertion $a : A \sqcap \neg B$ or the assertion $a : A$, difficulties start to arise when such a conjunction appears inside the scope of a quantifier, e.g. when having to update the assertion $a : \exists R.(A \sqcap B)$ to the either $a : \exists R.(A \sqcap \neg B)$ or $a : \exists R.A$. It is mainly for this reason that we start our investigations with a syntactic approach, i.e., making *syntactic* modifications to ABox assertions until consistency with the TBox is achieved.

As we mentioned in Section 1.2.3, in this chapter we will make use of the basic description logic $\mathcal{ALC}$. Furthermore, throughout the chapter we impose the following conventions:

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

The chapter will be presented as follows. In Section 3.2 we discuss the main issues of extending TBoxes with update actions and the main differences with $\mathcal{AIC}$s on databases. Section 3.2.1 is where the ideas discussed so far start to materialize with the first formal definition of '*active*' TBoxes as extensions of the usual TBoxes. We then continue by taking a syntactic approach in Section 3.3, exploring ways in which we can reach a desired ABox that is repaired according to the preferences of these active TBoxes. We conclude in Section 3.4 with a discussion on the limitations of this syntactic way of repairing ABoxes, thus motivating further the semantic approaches that will follow in subsequent chapters.

[1] A *literal* in the DL setting is defined similarly to the propositional, i.e., either an atomic concept or the negation thereof.

3.2 Integrating Active Constraints to the TBox

We have already witnessed and discussed the importance and effectiveness of $\mathcal{AIC}$s on databases and how giving a designer tools to express preferences can lead to more informed ways of maintaining databases. We have also argued that such tools should be very useful for ontology engineers as well. Let's start by giving a simple example of a TBox in Figure 3.1, 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 $\mathsf{Bachelor} \sqcap \mathsf{Married}$ among its assertions is clearly inconsistent with respect to this TBox and has to be repaired. Dropping any 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, dropping 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 inclusion of Figure 3.1 everybody has to have a marital status, the preferred update action would be to add the concept $\mathsf{Bachelor}$ to an individual violating this axiom, as in any other case the person would have to declare that s/he got married, divorced or widowed. Finally, as we can see it is clearly stated that someone cannot be identified as *divorced* and *widowed* at the same time. However, as a distinction between divorced and widowed cannot be made without further information, the axiom $\mathsf{Divorced} \sqcap \mathsf{Widowed} \sqsubseteq \bot$ should not give any preference between the concepts $\mathsf{Divorced}$ and $\mathsf{Widowed}$. So, as witnessed by this example, there exist logical reasons for extending TBoxes and equipping them with extra information on preferences as well as investigating ways to make use of them.

A first difficulty comes in the form of the concept constructors that are employed by most DL languages and, more specifically, value restriction and full existential quantification. While $\mathcal{AIC}$s on propositional databases have a specific structure that makes the addition of update actions easy and straightforward (we have already investigated this structure in Chapter 2), it is not as straightforward to find the logical form of $\mathcal{AIC}$s in the case of DLs. Having complex concept descriptions inside TBox axioms means that

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

Figure 3.1: Example of a TBox

'simple' update actions like additions or deletions of atomic concepts is not enough, since the inconsistencies that may arise from complex concepts inside the axioms will not have a way to be repaired. It is for this reason that the active axioms that we will define in this and the next chapter will ultimately prove to be quite limited on the repairing routes that they will be able to provide.

Another differentiation between the TBox axioms and the constraints on databases is the *open world* semantics that the ABoxes of DL Knowledge Bases employ in contrast to the *closed world* semantics of classical databases. Whereas in the latter one assumes to have *complete* knowledge of the facts expressed in a database, the description of a situation through an ABox is always incomplete, i.e., the *absence* of a statement does not imply its *negation*. We can also witness this distinction from the models of the DL Knowledge Bases and the fact that consistent ABoxes have multiple models which may describe differently the various parts of the domain not expressed by the ABox. This means that a repaired ABox with respect to a TBox is one that is *consistent* with the TBox: there exists an interpretation which is a model of both. Whereas for databases a repaired database is one which entails all the constraints, in the case of DL Knowledge Bases there may be other interpretations which do not satisfy the KB. So instead of *model checking* if a repaired database is the model of (i.e., satisfies) the integrity constraints, we now move on to *consistency checking* (or *satisfiability checking*) of a repaired KB.

Moreover, the aforementioned differentiation due to the open world nature of DLs gives rise to a duality in the case of *removing* concepts. While the update action $\mathsf{add}(A)$

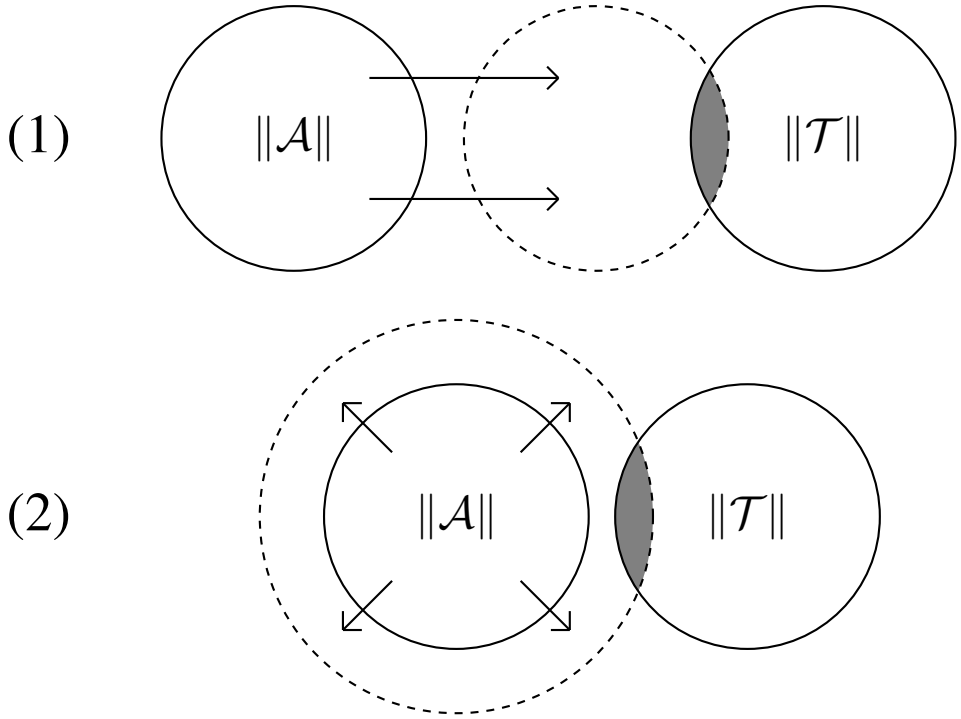

Figure 3.2: Removing (1) vs. forgetting (2)

clearly indicates that an individual is *added* to the instances of the atomic concept A, the update action $\mathsf{remove}(A)$ can be interpreted in two ways. The first one amounts to *removing* an individual from the instances of the atomic concept A and is the meaning which we will adopt in the following chapters. The second one amounts to *forgetting* the status of the atomic concept A on an individual, which means that there exist interpretations that include the individual in the instances of A and there exist interpretations that exclude the individual from the instances of A. Given an ABox $\mathcal{A}$ then, let $\|\mathcal{A}\|$ denote the set comprising the models of $\mathcal{A}$ and let $\mathcal{A}'$ be the outcome of removing an atomic concept from an individual. The sets $\|\mathcal{A}\|$ and $\|\mathcal{A}'\|$ are clearly different when *removing* is interpreted in the first way, while $\|\mathcal{A}\| \subset \|\mathcal{A}'\|$ when *removing* is interpreted in the second way. This is akin to the distinction between revision and contraction from the belief change literature [Alchourrón et al., 1985]. As we mentioned before, achieving consistency with a TBox $\mathcal{T}$ in both cases amounts to finding an interpretation satisfying both $\mathcal{A}'$ and $\mathcal{T}$, i.e., $\|\mathcal{A}'\| \cap \|\mathcal{T}\| \neq \emptyset$. Figure 3.2 shows the difference between *removing* and *forgetting* concepts in order to achieve consistency inside a KB. In the current chapter we do not differentiate between the two and $\mathsf{remove}(A)$ can be interpreted both ways. In the following chapters though we adopt the former meaning as it behaves better semantically and the

logics created this way come closer to the logic DL-PA we investigated in Chapter 2 (and whose atomic programs $A \leftarrow \bot$ are clearly interpreted in the former way).

3.2.1 The Active TBoxes

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

Definition 3.1. *A conjunctive constraint is any inclusion of the form* $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq \bot$. *A static constraint is any inclusion that is either a conjunctive constraint or equivalent to a conjunctive constraint.*

For example, in the TBox of Figure 3.1, 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:

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

In fact, any inclusion axiom of this chapter is a static constraint. We continue with the definition of an active constraint.

Definition 3.2. *Let* ρ *be a static constraint. If* ρ *is not a conjunctive constraint, let* $\rho_\sqcap$ *be the conjunctive constraint that is equivalent to* ρ. *An active constraint* η *is a couple* $\rho \to S$, *where* S *is a set of* $\mathsf{add}(A)$ *and* $\mathsf{remove}(A)$ *actions such that:*

- *if* $\mathsf{add}(A) \in S$ *then* $\neg A$ *is a literal on the conjunction of* ρ *(or* $\rho_\sqcap$*).*
- *if* $\mathsf{remove}(A) \in S$ *then* A *is a literal on the conjunction of* ρ *(or* $\rho_\sqcap$*).*

For instance, for the static constraint $A \sqcap \exists r.C \sqsubseteq B$ all the possible active constraints extending it are the following:

$\eta_1 : \ A \sqcap \exists r.C \sqsubseteq B \rightarrow \emptyset$

$\eta_2 : \ A \sqcap \exists r.C \sqsubseteq B \rightarrow \{\mathsf{add}(B)\}$

$\eta_3 : \ A \sqcap \exists r.C \sqsubseteq B \rightarrow \{\mathsf{remove}(A)\}$

$\eta_4 : \ A \sqcap \exists r.C \sqsubseteq B \rightarrow \{\mathsf{add}(B), \mathsf{remove}(A)\}$

We use the expression $\mathsf{body}(\eta)$ to denote the static constraint ρ and the expression $\mathsf{head}(\eta)$ to denote the set of update actions $\mathsf{add}(A)$ and $\mathsf{remove}(A)$ for atomic concepts A.

We can already witness by the above example how specific concept constructors provide no counterparts for preferred update actions if we simply adapt the idea behind $\mathcal{AIC}$s to DLs and only allow additions or deletions of atomic concepts. As we can see, η_2 and η_3 give a preference between one of the two atomic concepts and η_4 gives no preference to any of them. In case however one doesn't want to change the status of these atomic concepts then the only other solution is to not change anything through η_1, as there is no repairing path through the concept $\exists r.C$. So in the current state of active TBoxes, we have to keep in mind this limitation (that will be overcome in Chapter 5).

We formalize all the above by the relation $\rho \rightsquigarrow \eta$, where ρ is a static constraint and η is an active constraint extending it as described in Definition 3.2. The next step is to generalise this construction to TBoxes. We extend the relation $\rightsquigarrow$ and define *active* TBoxes as follows.

Definition 3.3. *Let $\mathcal{T}$ be a TBox. $\mathsf{a}\mathcal{T}$ is an active TBox extending $\mathcal{T}$, viz. $\mathcal{T} \rightsquigarrow \mathsf{a}\mathcal{T}$, iff for each static constraint ρ in $\mathcal{T}$ there is an active constraint η in $\mathsf{a}\mathcal{T}$ such that $\rho \rightsquigarrow \eta$ and, vice versa, for each active constraint η in $\mathsf{a}\mathcal{T}$ there is a static constraint ρ in $\mathcal{T}$ such that $\rho \rightsquigarrow \eta$.*

In Figure 3.3 we see an example of an active TBox, based on the TBox of Figure 3.1 and the discussion about the preferred update actions in order to repair it. Finally, for any active TBox $\mathsf{a}\mathcal{T}$ we denote with $\mathsf{static}(\mathsf{a}\mathcal{T})$ the TBox $\mathcal{T}$ for which $\mathcal{T} \rightsquigarrow \mathsf{a}\mathcal{T}$ and say that

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

Figure 3.3: Example of an active TBox, based on the TBox of Figure 3.1

an ABox $\mathcal{A}$ is consistent (respectively inconsistent) with respect to $\mathsf{a}\mathcal{T}$ iff $\mathcal{A}$ is consistent (respectively inconsistent) with respect to $\mathsf{static}(\mathsf{a}\mathcal{T})$.

In what follows, we present a first, syntactic approach 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.

3.3 A Syntactic Way of ABox Repairing

While updating a simple ABox (i.e., an ABox whose assertions consist only of concept literals) is quite straightforward, the update of an ABox which has complex concepts may not be easy (or even impossible) [Flouris et al., 2005, Flouris et al., 2009, Liu et al., 2011]. Consider for instance the active constraint $\eta = A \sqcap B \sqsubseteq \bot \rightarrow \{\mathsf{remove}(B)\}$ 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 like to repair this ABox with respect to η into the ABox having either $a : \forall r.(A \sqcap \neg B)$ or $a : \forall r.A$ as an assertion for the individual a[2]. 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 section we investigate how we could transform an initial ABox, inconsistent with respect

[2]Whereas the latter seems like a better candidate for a repair (taking into account the open-world nature of DLs we discussed in Section 3.2) 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.

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 could look like.

We start by defining a relation between two ABoxes, so that the second ABox is the outcome of applying a small change to the first ABox. Let $S_{\mathcal{A}}$ be the set that consists of all concept symbols in the ABox $\mathcal{A}$. For $A \in S_{\mathcal{A}}$ we 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, let:

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

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 3.4. *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 generalise this definition to an n-step relation between two ABoxes.

Definition 3.5. *Let $\mathcal{A}$ and $\mathcal{A}'$ be ABoxes and $n > 0$. Then $\mathcal{A} \sim_n \mathcal{A}'$ iff there are ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_{n+1}$ 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. $\mathcal{A}_1, \ldots, \mathcal{A}_{n+1}$ *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 $\mathcal{A}_2 = \mathcal{A}_1\,[A_1 \,|\, C_1]$ and $\mathcal{A}_3 = \mathcal{A}_2\,[A_2 \,|\, C_2]$ 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 3.6. *Let* $\mathcal{A}$ *and* $\mathcal{A}'$ *be ABoxes. Then* $\mathcal{A} \sim \mathcal{A}'$ *iff there exists* $n > 0$ *such that* $\mathcal{A} \sim_n \mathcal{A}'$.

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 $\mathcal{A} \not\sim \mathcal{A}$). 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 $\mathsf{a}\mathcal{T}$ be an ABox and an active TBox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathsf{a}\mathcal{T}$ 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 3.1. *Let* $\mathcal{A}$ *and* $\mathsf{a}\mathcal{T}$ *be an ABox and an active TBox respectively. Then for each* $n > 0$ *the set* $\mathcal{R}_n^{\mathcal{A}}$ *is finite.*

Proof. We start by noticing that since the ABox is always finite, the set $S_{\mathcal{A}}$ containing its 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.

The proof continues by induction. Let us 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 $\mathcal{A}' = \mathcal{A}\,[A \,|\, C]$ or $\mathcal{A} = \mathcal{A}'\,[A \,|\, C]$ 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 $\mathcal{A}' = \mathcal{A}\,[A \,|\, C]$ or $\mathcal{A} = \mathcal{A}'\,[A \,|\, C]$. As a result the set $\mathcal{R}_1^{\mathcal{A}}$ is also finite. Next, we make the induction hypothesis, i.e., we 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 us 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 3.2. *Let $\mathcal{A}$ and $\mathsf{a}\mathcal{T}$ be an ABox and an active TBox respectively. Then the set $\mathcal{R}^{\mathcal{A}}$ is finite.*

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 $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$, and also the set of individuals occurring in $\mathcal{A}$ is finite and doesn't increase, 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 3.7. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two ABoxes such that $\mathcal{A} \sim_1 \mathcal{A}'$. The syntactic modification from $\mathcal{A}$ to $\mathcal{A}'$ is the singleton set:*

$$U_{\mathcal{A}}^{\mathcal{A}'} = \begin{cases} \{A \mapsto C\} & \textit{if} \quad \mathcal{A}' = \mathcal{A}\,[A \,|\, C] \\ \{C \mapsto A\} & \textit{if} \quad \mathcal{A} = \mathcal{A}'\,[A \,|\, C] \end{cases}$$

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 3.8. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two ABoxes such that $\mathcal{A} \sim_n \mathcal{A}'$. The update sequence from $\mathcal{A}$ to $\mathcal{A}'$ is the sequence:*

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

where $\mathcal{A}_1, \ldots, \mathcal{A}_{n+1}$ are the semantically different ABoxes of Definition 3.5.

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 concepts that are counterparts of those that we already presented in Chapter 2 about active integrity constraints to show the relation between the two settings. We start by the definitions of *weak repair* and *PMA repair*.

Definition 3.9. *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 3.10. *Let $\mathcal{A}$ and $\mathsf{a}\mathcal{T}$ be an ABox and an active TBox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathsf{a}\mathcal{T}$. A syntactic modification $U_{\mathcal{A}}^{\mathcal{A}'}$ is* founded *if there is an active constraint η on $\mathsf{a}\mathcal{T}$ such that:*

1. *$\mathcal{A}$ is not consistent with respect to $\mathsf{body}(\eta)$.*
2. *$\mathcal{A}'$ is consistent with respect to $\mathsf{body}(\eta)$.*
3. *$U_{\mathcal{A}}^{\mathcal{A}'}$ either adds or removes a concept according to the update actions in $\mathsf{head}(\eta)$.*

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 η on $\mathsf{a}\mathcal{T}$ such that U is founded.

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

Definition 3.11. *Let $\mathcal{A}$ and $\mathsf{a}\mathcal{T}$ be an ABox and an active TBox respectively such that $\mathcal{A}$ is inconsistent with respect to $\mathsf{a}\mathcal{T}$.*

1. *An update sequence $S_{\mathcal{A}}^{\mathcal{A}'}$ is a founded weak repair of $\mathcal{A}$ by $\mathsf{a}\mathcal{T}$ if $S_{\mathcal{A}}^{\mathcal{A}'}$ is a weak repair of $\mathcal{A}$ achieving $\mathsf{static}(\mathsf{a}\mathcal{T})$ 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 $\mathsf{a}\mathcal{T}$ if $S_{\mathcal{A}}^{\mathcal{A}'}$ is a PMA repair of $\mathcal{A}$ achieving $\mathsf{static}(\mathsf{a}\mathcal{T})$ and $S_{\mathcal{A}}^{\mathcal{A}'}$ is founded.*

Summing up, let $\mathcal{A}$ be an ABox and $\mathsf{a}\mathcal{T}$ an active TBox such that $\mathcal{A}$ is inconsistent with respect to $\mathsf{a}\mathcal{T}$. A repaired ABox with respect to $\mathsf{a}\mathcal{T}$ 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 $\mathsf{a}\mathcal{T}$. A minimally repaired ABox with respect to $\mathsf{a}\mathcal{T}$ 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 $\mathsf{a}\mathcal{T}$. Since by

Proposition 3.2 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 a$\mathcal{T}$ 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 a$\mathcal{T}$.

We now return to the original example of Figures 3.1 and 3.3 and examine an ABox (which is inconsistent with respect to this TBox) and two of its possible repairs. Let a$\mathcal{T}$ be the active TBox of Figure 3.3 and consider the following $\mathcal{ALC}$-ABox:

$$\mathcal{A} = \{\mathsf{John} : \mathsf{Person} \sqcap \mathsf{Married} \sqcap \mathsf{Bachelor} \sqcap \exists\, \mathsf{hasChild}.\, (\mathsf{Divorced} \sqcap \mathsf{Widowed}),$$
$$\mathsf{Mary} : \mathsf{Person} \sqcap \neg\mathsf{Married} \sqcap \neg\mathsf{Divorced} \sqcap \neg\mathsf{Bachelor} \sqcap \neg\mathsf{Widowed}\}$$

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

$$\mathcal{A}' = \{\mathsf{John} : \mathsf{Person} \sqcap \mathsf{Married} \sqcap \exists\, \mathsf{hasChild}.\, \mathsf{Divorced},$$
$$\mathsf{Mary} : \mathsf{Person} \sqcap \neg\mathsf{Married} \sqcap \neg\mathsf{Divorced} \sqcap \mathsf{Bachelor} \sqcap \neg\mathsf{Widowed}\}$$

A founded repair of $\mathcal{A}$ by a$\mathcal{T}$ then is the update sequence $S_{\mathcal{A}}^{\mathcal{A}'} = (U_1, U_2, U_3)$ where $U_1 = \{\mathsf{Married} \sqcap \mathsf{Bachelor} \mapsto \mathsf{Married}\}, U_2 = \{\mathsf{Divorced} \sqcap \mathsf{Widowed} \mapsto \mathsf{Divorced}\}$ and $U_3 = \{\neg\mathsf{Bachelor} \mapsto \mathsf{Bachelor}\}$. Notice also that if we replace U_1 by $U_1' = \{\mathsf{Bachelor} \mapsto \neg\mathsf{Bachelor}\}$ and U_2 by $U_2' = \{\mathsf{Widowed} \mapsto \neg\mathsf{Widowed}\}$ 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 a$\mathcal{T}$, where $\mathcal{A} \diamond S_{\mathcal{A}}^{\mathcal{A}''}$ is the ABox:

$$\mathcal{A}'' = \{\mathsf{John} : \mathsf{Person} \sqcap \mathsf{Married} \sqcap \neg\mathsf{Bachelor} \sqcap \exists\, \mathsf{hasChild}.\, (\mathsf{Divorced} \sqcap \neg\mathsf{Widowed}),$$
$$\mathsf{Mary} : \mathsf{Person} \sqcap \neg\mathsf{Married} \sqcap \neg\mathsf{Divorced} \sqcap \mathsf{Bachelor} \sqcap \neg\mathsf{Widowed}\}$$

The ABoxes $\mathcal{A}'$ and $\mathcal{A}''$ also showcase in practice the difference between removing and forgetting concepts that we discussed in Section 3.2. Since we defined *minimality* to be relative to the number of syntactic modifications needed and not interpretation-wise, both $\mathcal{A}'$ and $\mathcal{A}''$ are considered to be minimally repaired.

3.4 Discussion and Conclusion

In this chapter we explored and discussed 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 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.

As we discussed, applying the idea behind $\mathcal{AIC}$s to the DL setting via the extension of TBox axioms with preferred update actions is not as straightforward as it is for databases. The main limitations of this chapter that we will try to tackle moving forward are (1) the fact that a syntactic repairing method cannot address *inferred* inconsistencies and (2) the fact that the preferred update actions of the active TBoxes deal only with atomic concepts. An example of the first case is when a concept is defined in the TBox and does not explicitly appear inside the ABox but is inferred, as is showcased in the following example.

Example 3.1. *Consider the following active TBoxes:*

$$\mathsf{a}\mathcal{T}_1 = \{B \equiv E \sqcap F, A \sqcap B \sqsubseteq \bot \rightarrow \{\mathsf{remove}(B)\}\}$$

$$\begin{aligned}\mathsf{a}\mathcal{T}_2 = \{&B \sqsubseteq E \sqcap F \rightarrow \{\mathsf{remove}(B)\},\\ &E \sqcap F \sqsubseteq B \rightarrow \{\mathsf{remove}(E)\},\\ &A \sqcap B \sqsubseteq \bot \rightarrow \{\mathsf{remove}(B)\}\}\end{aligned}$$

$$\mathsf{a}\mathcal{T}_3 = \{B \equiv E \sqcap F, A \sqcap E \sqcap F \sqsubseteq \bot \rightarrow \{\mathsf{remove}(E)\}, A \sqcap B \sqsubseteq \bot \rightarrow \{\mathsf{remove}(B)\}\}$$

and the ABoxes $\mathcal{A}_1 = \{\alpha : A \sqcap E \sqcap F\}$ *and* $\mathcal{A}_2 = \{\alpha : A \sqcap B \sqcap E \sqcap F\}$*. Using* $\mathsf{a}\mathcal{T}_1$ *we would not be able to provide a founded repair for any of the two ABoxes. For* $\mathcal{A}_1$ *the source of inconsistency (the concept B) is inferred by the first definition but cannot be removed as is. Similarly for* $\mathcal{A}_2$*. The only way to overcome this issue is by defining more precise active TBoxes. We can witness this with* $\mathsf{a}\mathcal{T}_2$*, which provides a founded repair for* $\mathcal{A}_2$*, and even more with* $\mathsf{a}\mathcal{T}_3$*, which provides a founded repair for both ABoxes.*

A semantic approach seems to behave better in general and provide solutions to problems that the syntactic one is unable to handle. This doesn't mean though that the syntactic investigations of this chapter are to no avail: the way to reconstruct new ABoxes through syntactic modifications of an initial ABox will prove very useful in combination with the subsequent semantic approaches, since the effectiveness of the latter will be mainly in evaluating whether an ABox is repaired but not in constructing it. We will thus revisit this chapter's ideas in the concluding Chapter 6 together with the semantic investigations to come.

Looking ahead, the most crucial thing will be to make the landscape of active TBoxes and the associated repairs more clear and intuitive. The logics that we will define are inspired by the way DL-PA was utilised to provide new kinds of repairs in the database literature. In this chapter we provided a first (small) step into this direction, carefully discussing the difficulties and differences of the DL setting as well as important limitations that we have to tackle. In Chapter 4 we give an emphasis on overcoming limitation (1), whereas Chapter 5 goes further and mainly focuses on limitation (2).

CHAPTER 4

A Semantic Approach to Repairing ABoxes: the Logic dyn$\mathcal{ALCO}$

Contents

4.1 Introduction

In this chapter we take a semantic approach and investigate how the various definitions of repairs from the database literature behave in the DL setting. As with Chapter 2, we represent update actions as atomic dynamic logic programs and show that repairs can be represented as complex programs. The main reason for using this framework is to account for dynamic repairing procedures that check in multiple steps the status of a possible repair and terminate once consistency with the TBox has been achieved. As we have seen before, this contrasts with active integrity constraint-based repairs where the update is applied in one go and later checked if it adheres to the preferences using certain

criteria. The dynamic logic framework also allows us to lift the approach of [Feuillade and Herzig, 2014] from propositional databases to DLs.

We consider that TBoxes are sets of *concept inclusion axioms* in the language of $\mathcal{ALC}$, without the unique name assumption. Unfortunately, there exist $\mathcal{ALC}$ ABoxes that when semantically updated cannot be expressed in $\mathcal{ALC}$. A simple example from [Liu et al., 2011] is the following: consider the ABox $\mathcal{A} = \{\mathsf{John} : \exists \mathsf{hasChild}.\mathsf{Happy}, \mathsf{Mary} : \mathsf{Happy} \sqcap \mathsf{Clever}\}$ which we want to update by making Mary unhappy. The resulting ABox then would have the form $\mathcal{A}' = \{\mathsf{John} : \exists \mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{Mary}\}), \mathsf{Mary} : \neg\mathsf{Happy} \sqcap \mathsf{Clever}\}$. It is for that reason that our ABoxes may contain nominals. Both the TBox and the ABox can be represented in our language by single formulas.

Before moving on let us showcase an example of an active TBox based on the intuitions described in Section 1.1 and the example therein. First, consider the following TBox $\mathcal{T}$:

$$\{\mathsf{Sibling} \sqsubseteq \mathsf{Brother} \sqcup \mathsf{Sister}, \forall \mathsf{hasSibling}.\bot \sqsubseteq \mathsf{OnlyChild}\}$$

An active TBox extending $\mathcal{T}$ then will be $\mathsf{a}\mathcal{T} = \{\eta_1, \eta_2\}$ where:

$$\eta_1 : \langle \mathsf{Sibling} \sqcap \neg\mathsf{Brother} \sqcap \neg\mathsf{Sister} \sqsubseteq \bot, \{-\mathsf{Sibling}\}\rangle$$
$$\eta_2 : \langle \forall \mathsf{hasSibling}.\bot \sqcap \neg\mathsf{OnlyChild} \sqsubseteq \bot, \{+\mathsf{OnlyChild}\}\rangle$$

As we already explained, through these enhanced concept inclusions we aim to be able to propose the update actions that we prefer when repairing an ABox that is inconsistent with $\mathcal{T}$. More specifically, the active axioms of $\mathsf{a}\mathcal{T}$ should dictate that an individual who is neither a brother nor a sister of someone should drop its sibling status, whereas an individual who has no siblings should change its status and be an only child. The direction that we pursue in this chapter is a *local* one, with preferred update actions like those showcased above being applied only to the level of the *individuals*, i.e., they have the form $+\mathsf{OnlyChild}(a)$ and $-\mathsf{Sibling}(b)$ for specific individuals a and b. This local approach to update actions comes close to the approach we have witnessed in Chapter 2 and thus the logic we propose bears many similarities with DL-PA.

The chapter is structured as follows. Section 4.2 presents syntax and semantics of formulas and programs. In Section 4.3 we prove decidability of the logic through reduction

axioms eliminating programs from formulas in a similar fashion to DL-PA. In Section 4.4 we transpose the notions of *weak repairs* and *PMA repairs* to the DL setting and use the programs of our language to constructively represent them. Section 4.5 is the main contribution of the chapter where we apply the idea behind active integrity constraints to Description Logic TBoxes and explore *founded* and *dynamic* repairs in a similar manner as in the database literature. We conclude in Section 4.6.

4.2 Syntax and Semantics

The update of an ABox is represented with the help of PDL programs. Their behavior is identical with the programs of PDL, with formulas of the form $\langle\pi\rangle\varphi$ expressing that φ is true after *some* possible execution of the program π. The main difference is at the atomic level: whereas PDL has abstract atomic programs, the atomic programs of our language have the form $\pm A(a)$ where $A(a)$ is an atomic assertion. This makes a big difference: the programs of our language can be eliminated and every formula is reducible to a *static formula*, i.e., a formula without programs. This is crucial in acquiring a unique representation of the update of an ABox through a static formula. We note that the models of our logic are different from (and quite simpler than) the Kripke models of PDL: the familiar interpretations of Description Logic suffice.

4.2.1 Language

Let Con be a countable set of atomic concepts, R a countable set of roles and Ind a countable set of individual names. The language of formulas and programs is defined by the following grammar:

$$\varphi ::= A \mid \{a\} \mid \bot \mid \varphi \to \varphi \mid \exists r.\varphi \mid \mathcal{U}\varphi \mid \langle\pi\rangle\varphi \mid \langle\pi\rangle^{\mathsf{c}}\varphi$$
$$\pi ::= +A(a) \mid -A(a) \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \varphi?$$

where $A \in$ Con, $a \in$ Ind, and $r \in$ R. *Atomic programs* have the form $+A(a)$ and $-A(a)$. The expression $\pm A(a)$ is used when we want to talk about both. The operators

of sequential and nondeterministic composition, the Kleene star and the test are familiar from PDL. $\mathcal{U}$ is the universal operator indicating that a formula is true for every individual of an interpretation.

The logical connectives $\neg$, $\top$, $\wedge$, $\vee$ and $\leftrightarrow$ as well as the universal quantifier $\forall$ are abbreviated in the usual way. The expression π^n abbreviates the program $\pi; \ldots; \pi$ where π appears n times, and $\pi^{\leq n}$ abbreviates the program $\Big(\pi \cup \top?\Big)^n$. We also define π^+ to be $\pi; \pi^*$. Furthermore, $\mathbf{while}\ \varphi\ \mathbf{do}\ \pi$ abbreviates the program $(\varphi?; \pi)^*; \neg\varphi?$ and $a : \varphi$ abbreviates the formula $\mathcal{U}(\{a\} \rightarrow \varphi)$. Finally, $r(a, b)$ abbreviates the formula $\mathcal{U}(\{a\} \rightarrow \exists r.\{b\})$.

We use the language of the basic description logic $\mathcal{ALC}$ for TBoxes and that of its extension $\mathcal{ALCO}$ with nominals for ABoxes. The TBoxes are composed of concept inclusion axioms and the ABoxes contain concept assertions $a\!:\!C$ and role assertions $r(a, b)$ where C is any concept description in $\mathcal{ALCO}$, r is a role and a, b are individuals. We will identify a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, respectively, with the formulas:

$$\bigwedge_{C \sqsubseteq D \in \mathcal{T}} \mathcal{U}\Big(C \rightarrow D\Big) \quad \text{and} \quad \bigwedge_{a:C \in \mathcal{A}} (a\!:\!C) \wedge \bigwedge_{r(a,b) \in \mathcal{A}} r(a, b)$$

Finally, $\mathsf{Con}(\pi)$ denotes the set of all atomic concepts occurring in the program π. Similarly, $\mathsf{Ind}(\pi)$ denotes the set of all individuals occurring in π.

4.2.2 Interpretations and their Updates

Interpretations are couples $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ maps each atomic concept A to a subset of $\Delta^{\mathcal{I}}$, each role r to a binary relation on $\Delta^{\mathcal{I}}$ and each individual name a to an element of $\Delta^{\mathcal{I}}$, i.e., $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

An *indexical update action* is an expression of the form $+A$ or $-A$, where A is an atomic concept. A *(non-indexical) update action* is an expression of the form $+A(a)$ or $-A(a)$, where A is an atomic concept and a an individual. A set of update actions U is consistent iff it does not contain both $+A(a)$ and $-A(a)$ for some assertion $A(a)$.

Definition 4.1. *Let U be a consistent set of update actions. The update of the interpreta-*

tion $\mathcal{I}$ by U, denoted by $\mathcal{I} \diamond U$, is the interpretation $\mathcal{I}'$ such that:

- $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$
- $a^{\mathcal{I}'} = a^{\mathcal{I}}$
- $A^{\mathcal{I}'} = \Big(A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid +A(a) \in U\}\Big) \setminus \{a^{\mathcal{I}} \mid -A(a) \in U\}$
- $r^{\mathcal{I}'} = r^{\mathcal{I}}$

Note that a consistent U can be identified with a program π_U, supposing that the update actions of U are applied in sequence (where, thanks to consistency, the order does not matter).

4.2.3 Semantics

The semantics of a formula is w.r.t. an interpretation $\mathcal{I}$: it is a set of individuals of $\Delta^{\mathcal{I}}$. The semantics of a program is a relation on interpretations. The extension of $\cdot^{\mathcal{I}}$ to complex formulas is defined inductively as follows:

$$
\begin{aligned}
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
\bot^{\mathcal{I}} &= \emptyset \\
\Big(\varphi \to \psi\Big)^{\mathcal{I}} &= \Big(\Delta^{\mathcal{I}} \setminus \varphi^{\mathcal{I}}\Big) \cup \psi^{\mathcal{I}} \\
\Big(\exists r.\varphi\Big)^{\mathcal{I}} &= \{\delta \in \Delta^{\mathcal{I}} \mid \text{ there is } \delta' \in \Delta^{\mathcal{I}} \text{ such that } (\delta, \delta') \in r^{\mathcal{I}} \text{ and } \delta' \in \varphi^{\mathcal{I}}\} \\
\Big(\mathcal{U}\varphi\Big)^{\mathcal{I}} &= \begin{cases} \Delta^{\mathcal{I}} & \text{if } \varphi^{\mathcal{I}} = \Delta^{\mathcal{I}} \\ \emptyset & \text{otherwise} \end{cases} \\
\Big(\langle \pi \rangle \varphi\Big)^{\mathcal{I}} &= \bigcup_{(\mathcal{I},\mathcal{I}') \in \| \pi \|} \varphi^{\mathcal{I}'} \\
\Big(\langle \pi \rangle^{\mathsf{c}} \varphi\Big)^{\mathcal{I}} &= \bigcup_{(\mathcal{I}',\mathcal{I}) \in \| \pi \|} \varphi^{\mathcal{I}'}
\end{aligned}
$$

while the semantics of programs is:

$$(\mathcal{I}, \mathcal{I}') \in ||+A(a)|| \text{ iff } \mathcal{I}' = \mathcal{I} \diamond \{+A(a)\}$$

$$(\mathcal{I}, \mathcal{I}') \in ||-A(a)|| \text{ iff } \mathcal{I}' = \mathcal{I} \diamond \{-A(a)\}$$

$$(\mathcal{I},\mathcal{I}') \in ||\pi_1;\pi_2|| \text{ iff there exists } \mathcal{I}'' \text{ such that } (\mathcal{I},\mathcal{I}'') \in ||\pi_1|| \text{ and } (\mathcal{I}'',\mathcal{I}') \in ||\pi_2||$$

$$(\mathcal{I},\mathcal{I}') \in ||\pi_1 \cup \pi_2|| \text{ iff } (\mathcal{I},\mathcal{I}') \in ||\pi_1|| \cup ||\pi_2||$$

$$(\mathcal{I},\mathcal{I}') \in ||\varphi?|| \text{ iff } \mathcal{I}' = \mathcal{I} \text{ and } \varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$(\mathcal{I},\mathcal{I}') \in ||\pi^*|| \text{ iff } (\mathcal{I},\mathcal{I}') \in \bigcup_{k\in\mathbb{N}_0} ||\pi||^k$$

Note that the test program is interpreted globally. An interpretation $\mathcal{I}$ is a *model of* a formula φ iff $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$. We say that φ is *(globally) satisfiable* iff there exists a model of φ. We say that φ is *valid* iff every interpretation is a model of φ. We call the logic that is built using this syntax and semantics dyn$\mathcal{ALCO}$.

The *update of an ABox $\mathcal{A}$ by a consistent set of update actions U* is the set:

$$\mathcal{A} \diamond U = \{\mathcal{I} \diamond U \mid \mathcal{I} \text{ is a model of } \mathcal{A}\}$$

This is a semantic definition. $\mathcal{A} \diamond U$ has also at least one syntactic representation, but it is not unique: there are many ABoxes that can describe it. What is of interest to us is that the set $\mathcal{A} \diamond U$ equals the set of interpretations satisfying $\langle \pi_U \rangle^{\mathsf{c}} \mathcal{A}$, where π_U is the program that applies the update actions of U. In this way, as we will soon see, we will be able to obtain a unique syntactic representation of the set $\mathcal{A} \diamond U$ through the formula $\langle \pi_U \rangle^{\mathsf{c}} \mathcal{A}$ of our logic.

4.3 Reduction Axioms and Decidability

We now show how to convert any dyn$\mathcal{ALCO}$ formula to an equivalent *static* formula, i.e., a formula without programs. We first reduce complex programs to atomic programs and then eliminate atomic programs from formulas. This is familiar e.g. from Dynamic Epistemic Logics, see [Ditmarsch et al., 2007]. We start with the reduction axioms for complex programs.

Proposition 4.1. *The following equivalences are valid:*

$$\langle \pi_1;\pi_2 \rangle \varphi \leftrightarrow \langle \pi_1 \rangle \langle \pi_2 \rangle \varphi$$

$$\langle \pi_1 \cup \pi_2 \rangle \varphi \leftrightarrow \langle \pi_1 \rangle \varphi \vee \langle \pi_2 \rangle \varphi$$

$$\langle \pi^* \rangle \varphi \leftrightarrow \langle \pi^{\leq 2^n} \rangle \varphi$$

$$\langle \psi? \rangle \varphi \leftrightarrow \varphi \wedge \mathcal{U}\psi$$

where $n = \mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))$.

Proof. For an equivalence $\varphi \leftrightarrow \psi$ to be valid, it suffices to show that $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}}$ for any interpretation $\mathcal{I}$. For an arbitrary interpretation $\mathcal{I}$ then we have:

- $\Big(\langle \pi_1; \pi_2 \rangle \varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi_1;\pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}'') \in \| \pi_1 \|} \bigcup\limits_{(\mathcal{I}'',\mathcal{I}') \in \| \pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}'') \in \| \pi_1 \|} \Big(\langle \pi_2 \rangle \varphi\Big)^{\mathcal{I}''} = \Big(\langle \pi_1 \rangle \langle \pi_2 \rangle \varphi\Big)^{\mathcal{I}}$
- $\Big(\langle \pi_1 \cup \pi_2 \rangle \varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi_1 \cup \pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi_1 \| \cup \| \pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi_1 \|} \varphi^{\mathcal{I}'} \cup \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi_2 \|} \varphi^{\mathcal{I}'} = \Big(\langle \pi_1 \rangle \varphi\Big)^{\mathcal{I}} \cup \Big(\langle \pi_2 \rangle \varphi\Big)^{\mathcal{I}} = \Big(\langle \pi_1 \rangle \varphi \vee \langle \pi_2 \rangle \varphi\Big)^{\mathcal{I}}$
- In dyn$\mathcal{ALCO}$, programs can only modify the assertions (i.e., the status of atomic concepts on individuals) that occur in them. This means that for any program π with $\mathsf{Con}(\pi) = \{A_1, \ldots, A_n\}$:

  $$\text{if } (\mathcal{I}, \mathcal{I}') \in ||\pi^*|| \text{ then } A^{\mathcal{I}} = A^{\mathcal{I}'} \text{ for } A \neq A_i, i \in \{1, \ldots, n\}$$

  Furthermore, by the semantics of update actions:

  $$\text{if } (\mathcal{I}, \mathcal{I}') \in ||\pi^*|| \text{ then } A^{\mathcal{I}} \setminus A^{\mathcal{I}'} \subseteq \mathsf{Ind}(\pi) \text{ and } A^{\mathcal{I}'} \setminus A^{\mathcal{I}} \subseteq \mathsf{Ind}(\pi) \text{ for all } A \in \mathsf{Con}(\pi)$$

  More formally, let us define $\pi(\mathcal{I}) = \{\mathcal{I}' \mid (\mathcal{I}, \mathcal{I}') \in ||\pi||\}$. Then we have that:

  $$\pi^*(\mathcal{I}) = \{\mathcal{I}' \mid A^{\mathcal{I}} \setminus A^{\mathcal{I}'} \subseteq \mathsf{Ind}(\pi) \text{ and } A^{\mathcal{I}'} \setminus A^{\mathcal{I}} \subseteq \mathsf{Ind}(\pi) \text{ for all } A \in \mathsf{Con}(\pi)\}$$

  It follows that for a given interpretation $\mathcal{I}$ there are at most $2^{\mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))}$ different interpretations $\mathcal{I}'$ such that $\mathcal{I}' \in \pi^*(\mathcal{I})$ or, equivalently, such that $(\mathcal{I}, \mathcal{I}') \in ||\pi^*||$. This means that $(\mathcal{I}, \mathcal{I}') \in ||\pi^*||$ iff $(\mathcal{I}, \mathcal{I}') \in ||\pi^{\leq 2^n}||$, where $n = \mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))$. Therefore: $\Big(\langle \pi^* \rangle \varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi^* \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pi^{\leq 2^n} \|} \varphi^{\mathcal{I}'} = \Big(\langle \pi^{\leq 2^n} \rangle \varphi\Big)^{\mathcal{I}}$ where $n = \mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))$
- If $\psi^{\mathcal{I}} = \Delta^{\mathcal{I}}$: $\Big(\langle \psi? \rangle \varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \psi? \|} \varphi^{\mathcal{I}'} = \varphi^{\mathcal{I}}$ and $\Big(\varphi \wedge \mathcal{U}\psi\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}} \cap \Big(\mathcal{U}\psi\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}} \cap \Delta^{\mathcal{I}} = \varphi^{\mathcal{I}}$

If $\psi^{\mathcal{I}} \neq \Delta^{\mathcal{I}}$: $\Big(\langle \psi? \rangle\, \varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \psi? \|} \varphi^{\mathcal{I}'} = \emptyset$ and $\Big(\varphi \wedge \mathcal{U}\psi\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}} \cap \Big(\mathcal{U}\psi\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}} \cap \emptyset = \emptyset$

□

Observe that, contrarily to PDL and just as in DL-PA, the Kleene star can be eliminated. The next proposition shows how to reduce atomic programs.

Proposition 4.2. *The following equivalences are valid:*

$$\langle +A(a) \rangle B \leftrightarrow \begin{cases} \{a\} \vee B & \textit{if } A = B \\ B & \textit{otherwise} \end{cases}$$

$$\langle -A(a) \rangle B \leftrightarrow \begin{cases} \neg\{a\} \wedge B & \textit{if } A = B \\ B & \textit{otherwise} \end{cases}$$

$$\langle \pm A(a) \rangle\, \{b\} \leftrightarrow \{b\}$$

$$\langle \pm A(a) \rangle \bot \leftrightarrow \bot$$

$$\langle \pm A(a) \rangle\, (\varphi \to \psi) \leftrightarrow \Big(\langle \pm A(a) \rangle\, \varphi \to \langle \pm A(a) \rangle\, \psi\Big)$$

$$\langle \pm A(a) \rangle \exists r.\varphi \leftrightarrow \exists r.\langle \pm A(a) \rangle \varphi$$

$$\langle \pm A(a) \rangle\, \mathcal{U}\varphi \leftrightarrow \mathcal{U}\langle \pm A(a) \rangle \varphi$$

where $A, B \in \mathsf{Con}$, $r \in \mathsf{R}$ *and* $a, b \in \mathsf{Ind}$.

Proof. For an arbitrary interpretation $\mathcal{I}$ we have:

- $\Big(\langle +A(a) \rangle A\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| +A(a) \|} A^{\mathcal{I}'} = A^{\mathcal{I}'}$ where $\mathcal{I}' = \mathcal{I} \diamond \{+A(a)\}$. But $A^{\mathcal{I}'} = A^{\mathcal{I}} \cup \{a^{\mathcal{I}}\} = \Big(A \vee \{a\}\Big)^{\mathcal{I}}$
- $\Big(\langle +A(a) \rangle B\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| +A(a) \|} B^{\mathcal{I}'} = B^{\mathcal{I}'}$ $\Big(\text{where } \mathcal{I}' = \mathcal{I} \diamond \{+A(a)\}\Big) = B^{\mathcal{I}}$
- $\Big(\langle -A(a) \rangle A\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| -A(a) \|} A^{\mathcal{I}'} = A^{\mathcal{I}'}$ where $\mathcal{I}' = \mathcal{I} \diamond \{-A(a)\}$. But $A^{\mathcal{I}'} = A^{\mathcal{I}} \setminus \{a^{\mathcal{I}}\} = A^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus \{a\}^{\mathcal{I}}) = \Big(A \wedge \neg\{a\}\Big)^{\mathcal{I}}$
- $\Big(\langle -A(a) \rangle B\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| -A(a) \|} B^{\mathcal{I}'} = B^{\mathcal{I}'}$ $\Big(\text{where } \mathcal{I}' = \mathcal{I} \diamond \{-A(a)\}\Big) = B^{\mathcal{I}}$
- $\Big(\langle \pm A(a) \rangle\, \{b\}\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}') \in \| \pm A(a) \|} \{b\}^{\mathcal{I}'} = \{b\}^{\mathcal{I}'}$ $\Big(\text{where } \mathcal{I}' = \mathcal{I} \diamond \{\pm A(a)\}\Big) = \{b\}^{\mathcal{I}}$

- $\Big(\langle \pm A(a)\rangle \bot\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} \bot^{\mathcal{I}'} = \emptyset = \bot^{\mathcal{I}}$

- $\Big(\langle \pm A(a)\rangle (\varphi \to \psi)\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} (\varphi \to \psi)^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} \Big((\Delta^{\mathcal{I}'} \setminus \varphi^{\mathcal{I}'}) \cup \psi^{\mathcal{I}'}\Big) = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} (\Delta^{\mathcal{I}'} \setminus \varphi^{\mathcal{I}'}) \cup \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} \psi^{\mathcal{I}'}$. But $\bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} (\Delta^{\mathcal{I}'} \setminus \varphi^{\mathcal{I}'}) = \Delta^{\mathcal{I}'} \setminus \varphi^{\mathcal{I}'}$ where $\mathcal{I}' = \mathcal{I} \diamond \{\pm A(a)\}$. Furthermore, $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and thus $\Delta^{\mathcal{I}'} \setminus \varphi^{\mathcal{I}'} = \Delta^{\mathcal{I}} \setminus \varphi^{\mathcal{I}'} = \Delta^{\mathcal{I}} \setminus \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} \varphi^{\mathcal{I}'} = \Delta^{\mathcal{I}} \setminus \Big(\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}}$. This gives that $\Big(\langle \pm A(a)\rangle (\varphi \to \psi)\Big)^{\mathcal{I}} = \Big(\Delta^{\mathcal{I}} \setminus \Big(\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}}\Big) \cup \Big(\langle \pm A(a)\rangle \psi\Big)^{\mathcal{I}} = \Big(\langle \pm A(a)\rangle \varphi \to \langle \pm A(a)\rangle \psi\Big)^{\mathcal{I}}$

- $\Big(\langle \pm A(a)\rangle \exists r.\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} (\exists r.\varphi)^{\mathcal{I}'} = \{\delta \in \Delta^{\mathcal{I}'} \mid$ there is $\delta' \in \Delta^{\mathcal{I}'}$ such that $(\delta,\delta') \in r^{\mathcal{I}'}$ and $\delta' \in \varphi^{\mathcal{I}'}$, where $\mathcal{I}' = \mathcal{I} \diamond \{\pm A(a)\}\}$. But $\Delta^{\mathcal{I}'}{=}\Delta^{\mathcal{I}}$ and $r^{\mathcal{I}'}{=}r^{\mathcal{I}}$ and thus $\Big(\langle \pm A(a)\rangle \exists r.\varphi\Big)^{\mathcal{I}}{=}\{\delta \in \Delta^{\mathcal{I}} \mid$ there is $\delta' \in \Delta^{\mathcal{I}}$ such that $(\delta,\delta') \in r^{\mathcal{I}}$ and $\delta' \in \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} \varphi^{\mathcal{I}'} = \Big(\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}}\} = \Big(\exists r.\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}}$

- $\Big(\langle \pm A(a)\rangle \mathcal{U}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \pm A(a) \|} (\mathcal{U}\varphi)^{\mathcal{I}'} = \begin{cases} \Delta^{\mathcal{I}'} & \text{if } \varphi^{\mathcal{I}'} = \Delta^{\mathcal{I}'} \\ \emptyset & \text{otherwise} \end{cases}$, where $\mathcal{I}' = \mathcal{I} \diamond \{\pm A(a)\}$. But $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\varphi^{\mathcal{I}'} = \Big(\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}}$ and thus:
$\Big(\langle \pm A(a)\rangle \mathcal{U}\varphi\Big)^{\mathcal{I}} = \begin{cases} \Delta^{\mathcal{I}} & \text{if } \Big(\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}} = \Delta^{\mathcal{I}} \\ \emptyset & \text{otherwise} \end{cases} = \Big(\mathcal{U}\langle \pm A(a)\rangle \varphi\Big)^{\mathcal{I}}$

□

Finally, the reduction axioms for the converse operator follow.

Proposition 4.3. *The following equivalences are valid:*

$$\langle +A(a)\rangle^{\mathsf{c}} \varphi \leftrightarrow (a\!:\!A) \wedge \Big(\varphi \vee \langle -A(a)\rangle \varphi\Big)$$
$$\langle -A(a)\rangle^{\mathsf{c}} \varphi \leftrightarrow (a\!:\!\neg A) \wedge \Big(\varphi \vee \langle +A(a)\rangle \varphi\Big)$$
$$\langle \pi_1;\pi_2\rangle^{\mathsf{c}} \varphi \leftrightarrow \langle \pi_2\rangle^{\mathsf{c}} \langle \pi_1\rangle^{\mathsf{c}} \varphi$$
$$\langle \pi_1 \cup \pi_2\rangle^{\mathsf{c}} \varphi \leftrightarrow \langle \pi_1\rangle^{\mathsf{c}} \varphi \vee \langle \pi_2\rangle^{\mathsf{c}} \varphi$$
$$\langle \pi^*\rangle^{\mathsf{c}} \varphi \leftrightarrow \langle \pi^{\leq 2^n}\rangle^{\mathsf{c}} \varphi$$
$$\langle \psi?\rangle^{\mathsf{c}} \varphi \leftrightarrow \langle \psi?\rangle \varphi$$

where $n = \mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))$.

Proof. For an arbitrary interpretation $\mathcal{I}$ we have:

- If $a^{\mathcal{I}} \notin A^{\mathcal{I}}$: $\Big(\langle +A(a)\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| +A(a) \|} \varphi^{\mathcal{I}'} = \emptyset$ as there is no $\mathcal{I}'$ such that $\mathcal{I} = \mathcal{I}' \diamond \{+A(a)\}$ and $\Big((a\!:\!A) \wedge \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)\Big)^{\mathcal{I}} = (a:A)^{\mathcal{I}} \cap \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)^{\mathcal{I}} = \Big(\mathcal{U}(\{a\} \to A)\Big)^{\mathcal{I}} \cap \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)^{\mathcal{I}} = \emptyset \cap \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)^{\mathcal{I}} = \emptyset$
  If $a^{\mathcal{I}} \in A^{\mathcal{I}}$: $\Big(\langle +A(a)\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| +A(a) \|} \varphi^{\mathcal{I}'} = \varphi^{\mathcal{I}_1} \cup \varphi^{\mathcal{I}_2}$ where $\mathcal{I}_1 = \mathcal{I}$ and $\mathcal{I}_2 = \mathcal{I} \diamond \{-A(a)\}$. Indeed, it is easy to check that $(\mathcal{I}_i, \mathcal{I}) \in \| +A(a) \|$ only for $i \in \{1,2\}$ when $a^{\mathcal{I}} \in A^{\mathcal{I}}$. Furthermore: $\Big((a\!:\!A) \wedge \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)\Big)^{\mathcal{I}} = (a:A)^{\mathcal{I}} \cap \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)^{\mathcal{I}} = \Big(\mathcal{U}(\{a\} \to A)\Big)^{\mathcal{I}} \cap \Big(\varphi^{\mathcal{I}} \cup \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| -A(a) \|} \varphi^{\mathcal{I}'}\Big) = \Delta^{\mathcal{I}} \cap (\varphi^{\mathcal{I}} \cup \varphi^{\mathcal{I}'})$ where $\mathcal{I}' = \mathcal{I} \diamond \{-A(a)\}$ and consequently: $\Big((a\!:\!A) \wedge \Big(\varphi \vee \langle -A(a)\rangle\, \varphi\Big)\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}_1} \cup \varphi^{\mathcal{I}_2}$
- $\Big(\langle -A(a)\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \Big((a\!:\!\neg A) \wedge \Big(\varphi \vee \langle +A(a)\rangle\, \varphi\Big)\Big)^{\mathcal{I}}$ similarly to the previous procedure but reversing the arguments.
- $\Big(\langle \pi_1;\pi_2\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi_1;\pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I}',\mathcal{I}'')\in \| \pi_1 \|} \bigcup\limits_{(\mathcal{I}'',\mathcal{I})\in \| \pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I}'',\mathcal{I})\in \| \pi_2 \|} \bigcup\limits_{(\mathcal{I}',\mathcal{I}'')\in \| \pi_1 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I}'',\mathcal{I})\in \| \pi_2 \|} \Big(\langle \pi_1\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}''} = \Big(\langle \pi_2\rangle^{\mathsf{c}}\langle \pi_1\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}}$
- $\Big(\langle \pi_1 \cup \pi_2\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi_1\cup\pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi_1 \| \cup \| \pi_2 \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi_1 \|} \varphi^{\mathcal{I}'} \cup \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi_2 \|} \varphi^{\mathcal{I}'} = \Big(\langle \pi_1\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} \cup \Big(\langle \pi_2\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \Big(\langle \pi_1\rangle^{\mathsf{c}}\varphi \vee \langle \pi_2\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}}$
- From the proof of Proposition 4.1 we have that $(\mathcal{I},\mathcal{I}') \in ||\pi^*||$ iff $(\mathcal{I},\mathcal{I}') \in ||\pi^{\leq 2^n}||$, where $n = \mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))$. Therefore: $\Big(\langle \pi^*\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi^* \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \pi^{\leq 2^n} \|} \varphi^{\mathcal{I}'} = \Big(\langle \pi^{\leq 2^n}\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}}$ where $n = \mathsf{card}(\mathsf{Con}(\pi)) \times \mathsf{card}(\mathsf{Ind}(\pi))$
- $\Big(\langle \psi?\rangle^{\mathsf{c}}\varphi\Big)^{\mathcal{I}} = \bigcup\limits_{(\mathcal{I}',\mathcal{I})\in \| \psi? \|} \varphi^{\mathcal{I}'} = \bigcup\limits_{(\mathcal{I},\mathcal{I}')\in \| \psi? \|} \varphi^{\mathcal{I}'} = \Big(\langle \psi?\rangle\, \varphi\Big)^{\mathcal{I}}$ since $(\mathcal{I}_1,\mathcal{I}_2) \in \| \psi? \|$ iff $(\mathcal{I}_2,\mathcal{I}_1) \in \| \psi? \|$

□

Using now Propositions 4.1, 4.2 and 4.3 we obtain the following theorem.

Theorem 4.1. *Every formula of* dyn$\mathcal{ALCO}$ *is equivalent to a static formula.*

Proof. Let φ be any arbitrary dyn$\mathcal{ALCO}$-formula. As we mentioned at the beginning of this section, we start by using Proposition 4.1 together with the last four equivalences of Proposition 4.3 iteratively until all complex programs inside φ are eliminated and φ ends up comprising only atomic programs. We then use the last four equivalences of Proposition 4.2 and the first two of Proposition 4.3 iteratively until no instance of the converse operator exists inside φ and all atomic programs are pushed through the boolean connectives as well as the universal/existential quantifiers and the universal operator. Finally, through the first three equivalences of Proposition 4.2 we substitute any atomic program on the left part of each equivalence with the equivalent formula on the right. Since all equivalences are valid the resulting reduced formula is equivalent to φ and contains no programs, i.e., φ is equivalent to a static formula. □

Through Theorem 4.1 we can now obtain a unique syntactic representation of the set $\mathcal{A} \diamond U$ by reducing the formula $\langle \pi_U \rangle^{\mathsf{c}} \mathcal{A}$ to a static one containing no programs.

Example 4.1 ([Liu et al., 2011])**.** *Let* $\mathcal{A} = \{\mathsf{John} : \exists \mathsf{hasChild}.\mathsf{Happy}, \mathsf{Mary} : \mathsf{Happy} \sqcap \mathsf{Clever}\}$ *and* $U = \{-\mathsf{Happy}(\mathsf{Mary})\}$. *Applying the reduction axioms to:*

$$\langle -\mathsf{Happy}(\mathsf{Mary}) \rangle^{\mathsf{c}} \Big((\mathsf{John} : \exists \mathsf{hasChild}.\mathsf{Happy}) \wedge (\mathsf{Mary} : \mathsf{Happy} \wedge \mathsf{Clever}) \Big)$$

we obtain the static formula:

$$\Big(\mathsf{John} : \exists \mathsf{hasChild}.(\mathsf{Happy} \vee \{\mathsf{Mary}\}) \Big) \wedge \Big(\mathsf{Mary} : \neg \mathsf{Happy} \wedge \mathsf{Clever} \Big)$$

which accurately represents $\mathcal{A} \diamond U$.

Last but not least, decidability of global satisfiability in dyn$\mathcal{ALCO}$ follows from the fact that any static formula can be mapped to an $\mathcal{ALCO}(\mathcal{U})$-concept and vice versa and that concept satisfiability in $\mathcal{ALCO}(\mathcal{U})$ is decidable [Horrocks et al., 2006]. This correspondence between static dyn$\mathcal{ALCO}$-formulas and $\mathcal{ALCO}(\mathcal{U})$-concepts is realized through the following one-to-one and onto mapping τ (note that we use the connectives $\neg$ and $\vee$ in the place of $\bot$ and $\rightarrow$ since the former are easier to present):

- $\tau(A) = A$

- $\tau(\{a\}) = \{a\}$
- $\tau(\neg\varphi) = \neg\tau(\varphi)$
- $\tau(\varphi \vee \psi) = \tau(\varphi) \sqcup \tau(\psi)$
- $\tau(\exists r.\varphi) = \exists r.\tau(\varphi)$
- $\tau(\mathcal{U}\varphi) = \forall r_U.\tau(\varphi)$

where $A \in \mathsf{Con}$, $a \in \mathsf{Ind}$, and $r \in \mathsf{R}$ (recall also that r_U is the universal role). Thus, (global) satisfiability checking of the static fragment of our logic can be reduced to the satisfiability problem of $\mathcal{ALCO}(\mathcal{U})$.

Lemma 4.1. *Let φ be a static* dyn$\mathcal{ALCO}$*-formula which is mapped to the $\mathcal{ALCO}(\mathcal{U})$-concept C, i.e., $\tau(\varphi) = C$. Then φ is (globally) satisfiable iff the concept $\forall r_U.C$ is satisfiable.*

Proof. First of all, by induction on φ it is easy to prove that $\Big(\tau(\varphi)\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}}$ for any interpretation $\mathcal{I}$. Since the first four cases are quite trivial we only give the details for the last two.

- Let $\varphi = \exists r.\psi$ and suppose the induction hypothesis holds for ψ, i.e., $\Big(\tau(\psi)\Big)^{\mathcal{I}} = \psi^{\mathcal{I}}$. Then:

$$\Big(\tau(\varphi)\Big)^{\mathcal{I}} = \Big(\tau(\exists r.\psi)\Big)^{\mathcal{I}} = \Big(\exists r.\tau(\psi)\Big)^{\mathcal{I}} =$$
$$\{\delta \in \Delta^{\mathcal{I}} \mid \text{ there is } \delta' \in \Delta^{\mathcal{I}} \text{ such that } (\delta, \delta') \in r^{\mathcal{I}} \text{ and } \delta' \in \Big(\tau(\psi)\Big)^{\mathcal{I}}\} =$$
$$\{\delta \in \Delta^{\mathcal{I}} \mid \text{ there is } \delta' \in \Delta^{\mathcal{I}} \text{ such that } (\delta, \delta') \in r^{\mathcal{I}} \text{ and } \delta' \in \psi^{\mathcal{I}}\} =$$
$$\Big(\exists r.\psi\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}}$$

- Let $\varphi = \mathcal{U}\psi$ and suppose the induction hypothesis holds for ψ, i.e., $\Big(\tau(\psi)\Big)^{\mathcal{I}} = \psi^{\mathcal{I}}$. Then:

$$\Big(\tau(\varphi)\Big)^{\mathcal{I}} = \Big(\tau(\mathcal{U}\psi)\Big)^{\mathcal{I}} = \Big(\forall r_U.\tau(\psi)\Big)^{\mathcal{I}} =$$
$$\{\delta \in \Delta^{\mathcal{I}} \mid \text{ for all } \delta' \in \Delta^{\mathcal{I}}, \text{ if } (\delta, \delta') \in r_U^{\mathcal{I}} \text{ then } \delta' \in \Big(\tau(\psi)\Big)^{\mathcal{I}}\} =$$
$$\{\delta \in \Delta^{\mathcal{I}} \mid \text{ for all } \delta' \in \Delta^{\mathcal{I}}, \delta' \in \psi^{\mathcal{I}}\} =$$

$$\{\delta \in \Delta^{\mathcal{I}} \mid \psi^{\mathcal{I}} = \Delta^{\mathcal{I}}\} = \begin{cases} \Delta^{\mathcal{I}} & \text{if } \psi^{\mathcal{I}} = \Delta^{\mathcal{I}} \\ \emptyset & \text{otherwise} \end{cases} = \Big(\mathcal{U}\psi\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}}$$

Now let φ be globally satisfiable. This means that there exists a model of φ, i.e., there is an interpretation $\mathcal{I}$ such that $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and subsequently $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ since $\varphi^{\mathcal{I}} = \Big(\tau(\varphi)\Big)^{\mathcal{I}} = C^{\mathcal{I}}$. Similarly to before, we have that $(\forall r_U.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid C^{\mathcal{I}} = \Delta^{\mathcal{I}}\}$. But $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and thus $(\forall r_U.C)^{\mathcal{I}} = \Delta^{\mathcal{I}}$. This gives $(\forall r_U.C)^{\mathcal{I}} \neq \emptyset$, i.e., the concept $\forall r_U.C$ is satisfiable. On the other direction, if the concept $\forall r_U.C$ is satisfiable then there exists an interpretation $\mathcal{I}$ such that $(\forall r_U.C)^{\mathcal{I}} \neq \emptyset$. This means that there exists a $\delta \in (\forall r_U.C)^{\mathcal{I}}$ and since $(\forall r_U.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid C^{\mathcal{I}} = \Delta^{\mathcal{I}}\}$ then $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. From $C^{\mathcal{I}} = \Big(\tau(\varphi)\Big)^{\mathcal{I}} = \varphi^{\mathcal{I}}$ it follows that $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$ as well, i.e., φ be globally satisfiable. □

Using Lemma 4.1 now we obtain the following theorem.

Theorem 4.2. *Global satisfiability in the logic* dyn$\mathcal{ALCO}$ *is decidable.*

Proof. Deciding whether a dyn$\mathcal{ALCO}$-formula φ is (globally) satisfiable is realized by (1) finding a static formula ψ that is equivalent to φ through Theorem 4.1 and (2) deciding whether ψ is (globally) satisfiable through Lemma 4.1. If there exists an interpretation $\mathcal{I}$ such that $\psi^{\mathcal{I}} = \Delta^{\mathcal{I}}$ then $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$ as well, since $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}}$ for any $\mathcal{I}$. Similarly if $\psi^{\mathcal{I}} \neq \Delta^{\mathcal{I}}$ for all $\mathcal{I}$. □

4.4 Weak Repairs and Repairs

From now on we consider a fixed satisfiable TBox $\mathcal{T}$ together with a fixed finite set Con of atomic concepts and Ind of individual names. Furthermore, we consider for the rest of the chapter that all ABoxes we work with are consistent. We proceed to define the notions of *weak repair* and *repair* of an ABox $\mathcal{A}$ with respect to $\mathcal{T}$, i.e., a modification of the ABox such that $\mathcal{A}$ is consistent with the concept inclusion axioms of $\mathcal{T}$. We recall that an ABox $\mathcal{A}$ is inconsistent with respect to $\mathcal{T}$ iff there is no interpretation satisfying both $\mathcal{A}$ and $\mathcal{T}$, and that in dyn$\mathcal{ALCO}$ this amounts to unsatisfiability of the formula $\mathcal{T} \wedge \mathcal{A}$. We start with the definition of a *weak repair*.

Definition 4.2. *Let $\mathcal{T} \wedge \mathcal{A}$ be unsatisfiable. A* weak repair *of $\mathcal{A}$ is a consistent set of update actions U such that $\mathcal{T} \wedge (\mathcal{A} \diamond U)$ is satisfiable.*

We continue with the definition of a *repair*, which is based on the principle of minimal change and Winslett's possible models approach [Winslett, 1988, Winslett, 1990, Herzig and Rifi, 1999].

Definition 4.3. *Let $\mathcal{T} \wedge \mathcal{A}$ be unsatisfiable. A* (PMA) repair *U of $\mathcal{A}$ is a weak repair of $\mathcal{A}$ that is minimal with respect to set inclusion, i.e., such that $\mathcal{T} \wedge (\mathcal{A} \diamond U')$ is unsatisfiable for all $U' \subset U$.*

We showcase the above definitions in the following simple example.

Example 4.2. *Let $\mathcal{T} = \{\, A \sqsubseteq \forall r.B \,\}$ and $\mathcal{A} = \{a : A, b : \neg B, r(a,b)\}$. Clearly, $\mathcal{T} \wedge \mathcal{A}$ is unsatisfiable. Any consistent set of update actions containing $-A(a)$ or $+B(b)$ is a weak repair of $\mathcal{A}$. Only the sets $U_1 = \{-A(a)\}$ and $U_2 = \{+B(b)\}$ are repairs of $\mathcal{A}$.*

Note that these definitions are based on the respective *weak repairs* and *PMA repairs* we have come across in Chapter 2. We now repair an ABox that is inconsistent with respect to $\mathcal{T}$ by means of dyn$\mathcal{ALCO}$ programs. We first define the program:

$$\mathsf{weakRepair} = \Big(\bigcup_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \big(+A(a) \cup -A(a) \big) \Big)^* \,;\, \mathcal{T}\,?$$

It is easy to see that, starting from an interpretation $\mathcal{I}$ that is a model of an ABox, any interpretation $\mathcal{I}'$ for which $(\mathcal{I}, \mathcal{I}') \in ||\mathsf{weakRepair}||$ is a model of a repaired ABox that is consistent with $\mathcal{T}$. Moreover, since by hypothesis $\mathcal{T}$ is satisfiable there is at least one such $\mathcal{I}'$.

Next, we show how to capture (PMA) repairs. The first step is to extend the set of atomic concepts Con by new concepts A^+ and A^- uniquely associated with each concept A. They allow us to keep track of the concepts that are added or removed. We use these then to define the following program:

$$\mathsf{toggle}^{\pm}(A(a)) = \big(\neg(a\!:\!A^+) \wedge \neg(a\!:\!A^-) \big)\,? \,;\, \big((+A(a)\,;+A^+(a)) \cup (-A(a)\,;+A^-(a)) \big)$$

which is intuitively the program $+A(a) \cup -A(a)$ but enhanced with update operations that keep track of any changes. The second step is to define the program:

$$\mathsf{undo}(A(a)) = \Big((a\!:\!A^{+})\,?\,;-A(a)\,;-A^{+}(a)\Big) \;\cup\; \Big((a\!:\!A^{-})\,?\,;+A(a)\,;-A^{-}(a)\Big)$$

which, as the name suggests, will undo any change that was imposed on an assertion.

Now, in order to redo any changes that are stored through A^{+} and A^{-} to an alternative interpretation, we use the program:

$$\mathsf{redo}(A(a)) = \Big((a\!:\!A^{+})\,?\,;+A(a)\Big) \cup \Big((a\!:\!A^{-})\,?\,;-A(a)\Big)$$

The last step in checking minimality is to define a program that visits all models of the original ABox $\mathcal{A}$. This will allow us to check whether we can obtain a weak repair using less update actions. It is easy to see that this can be achieved through the program:

$$\mathsf{gotoAltInt}(\mathcal{A}) = \Big(\bigcup_{\substack{A\in\mathsf{Con}\\ a\in\mathsf{Ind}}}\Big(+A(a)\cup -A(a)\Big)\Big)^{*}\,;\,\mathcal{A}\,?$$

Summing up, we use the above to create the formula:

$$\mathsf{Minimal}(\mathcal{A}) = \neg\,\langle\,\mathsf{gotoAltInt}(\mathcal{A})\,;\Big(\bigcup_{\substack{A\in\mathsf{Con}\\ a\in\mathsf{Ind}}}\mathsf{redo}(A(a))\Big)^{*};\Big(\bigcup_{\substack{A\in\mathsf{Con}\\ a\in\mathsf{Ind}}}\mathsf{undo}(A(a))\Big)^{+}\rangle\,\mathcal{T}$$

which is the key ingredient in checking if a repair is minimal with respect to other weak repairs. Consequently:

$$\mathsf{repair}(\mathcal{A}) = \Big(\bigcup_{\substack{A\in\mathsf{Con}\\ a\in\mathsf{Ind}}}\mathsf{toggle}^{\pm}(A(a))\Big)^{*};\mathcal{T}\,?\,;\,\mathsf{Minimal}(\mathcal{A})\,?$$

Note that the $\Big(\bigcup_{\substack{A\in\mathsf{Con}\\ a\in\mathsf{Ind}}}\mathsf{redo}(A(a))\Big)^{*}$ program may not actually reapply all of the update actions that may have been stored earlier through the $\mathsf{toggle}^{\pm}(A(a))$ program: it does not need to, as reapplying only some of them is equivalent to reapplying all of them and undoing all those that were left out.

Last but not least, let $\mathsf{mkFalse}^{\pm}$ abbreviate the program which falsifies all the asser-

tions of the form $A^+(a)$ and $A^-(a)$ for all $A \in \mathsf{Con}$ and $a \in \mathsf{Ind}$. The program $\mathsf{mkFalse}^{\pm}$ will be used in the final step to make sure that none of the new concepts that were used for storing purposes survive in the repair.

We showcase all of the above in the following theorem.

Theorem 4.3. *Let $\mathcal{A}$ be inconsistent with respect to $\mathcal{T}$ and let U be a consistent set of update actions. Furthermore, let no atomic concept A^+ or A^- appear in any of them.*

- *U is a weak repair of $\mathcal{A}$ iff there exists an $\mathcal{I}$ such that:*

$$(\mathcal{I}, \mathcal{I} \diamond U) \in \Big|\Big| \mathcal{A}\,? \,;\, \mathsf{weakRepair} \Big|\Big|$$

- *U is a repair of $\mathcal{A}$ iff there exists an $\mathcal{I}$ such that:*

$$(\mathcal{I}, \mathcal{I} \diamond U) \in \Big|\Big| \Big(\mathcal{A} \wedge \bigwedge_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \big(\neg(a : A^+) \wedge \neg(a : A^-) \big) \Big)\,? \,;\, \mathsf{repair}(\mathcal{A}) \,;\, \mathsf{mkFalse}^{\pm} \Big|\Big|$$

Proof. For the first item, suppose there exists an interpretation $\mathcal{I}$ such that $(\mathcal{I}, \mathcal{I} \diamond U) \in \Big|\Big| \mathcal{A}\,? \,;\, \mathsf{weakRepair} \Big|\Big|$. This means that $\mathcal{I}$ is a model of $\mathcal{A}$, since $\mathcal{A}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ by the initial test. By definition then, $\mathcal{I} \diamond U$ is a model of $\mathcal{A} \diamond U$, where U is the consistent set of update actions. Furthermore, the final test of the weakRepair program implies that $\mathcal{I} \diamond U$ is a model of $\mathcal{T}$. Given these we have that $\mathcal{I} \diamond U$ is both a model of $\mathcal{T}$ and a model of $\mathcal{A} \diamond U$, i.e., the formula $\mathcal{T} \wedge (\mathcal{A} \diamond U)$ is satisfiable. By Definition 4.2 then U is a weak repair of $\mathcal{A}$. For the other direction, let U be a weak repair of $\mathcal{A}$. By Definition 4.2 this means that $\mathcal{T} \wedge (\mathcal{A} \diamond U)$ is satisfiable, i.e., there is an interpretation $\mathcal{I}'$ that is both a model of $\mathcal{T}$ and a model of $\mathcal{A} \diamond U$. By definition again, we have that $\mathcal{I}'$ is a model of $\mathcal{A} \diamond U$ iff $\mathcal{I}'$ has the form $\mathcal{I} \diamond U$ where $\mathcal{I}$ is a model of $\mathcal{A}$. This means that there is a model $\mathcal{I}$ of $\mathcal{A}$ such that $\mathcal{I} \diamond U$ is a model of $\mathcal{T}$. So there exists an $\mathcal{I}$ such that $(\mathcal{I}, \mathcal{I}) \in \Big|\Big| \mathcal{A}\,? \Big|\Big|$ and $(\mathcal{I} \diamond U, \mathcal{I} \diamond U) \in \Big|\Big| \mathcal{T}\,? \Big|\Big|$. Furthermore, it is easy to see that:

$$(\mathcal{I}, \mathcal{I} \diamond U) \in \Big|\Big| \Big(\bigcup_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \big(+A(a) \cup -A(a) \big) \Big)^* \Big|\Big|$$

and thus $(\mathcal{I}, \mathcal{I} \diamond U) \in \Big|\Big| \mathcal{A}\,? \,;\, \mathsf{weakRepair} \Big|\Big|$.

For the second item, we can see how the construction of $\mathsf{repair}(\mathcal{A})$ via the programs

$\mathsf{toggle}^{\pm}(A(a))$, $\mathsf{gotoAltInt}(\mathcal{A})$, $\mathsf{redo}(A(a))$ and $\mathsf{undo}(A(a))$ makes sure that there is no alternative model of $\mathcal{A}$ that achieves consistency with $\mathcal{T}$ using less update actions. The procedure is based on the respective one for the propositional case of Chapter 2, with the difference being that now it doesn't suffice to just undo some update actions and check if the ensuing interpretation is a model of $\mathcal{T}$. Indeed, due to the open world semantics of ABoxes there is no unique model of $\mathcal{A}$ (as was the case with databases) and we may begin from a 'problematic' model of $\mathcal{A}$ which needs extra update actions to tackle inconsistencies between (possibly irrelevant) individuals of the model and axioms of the TBox. In such a case, the update actions chosen from the $\mathsf{toggle}^{\pm}(A(a))$ program may indeed be minimal with respect to the specific interpretation and just retracting them may lead to interpretations that are not models of $\mathcal{T}$, thus incorrectly identifying them as a minimal set of update actions repairing $\mathcal{A}$. To be sure we did not choose such an inappropriate interpretation, we need to visit all possible models of $\mathcal{A}$ and check for each one that it cannot achieve consistency with less update actions by means of the $\mathsf{redo}(A(a))$ and $\mathsf{undo}(A(a))$ programs. All of these are realized with the use of the new concepts of the form A^{+} and A^{-}, similarly to those of Chapter 2, ending with the $\mathsf{mkFalse}^{\pm}$ program which removes them from U. □

Let us illustrate this by continuing Example 4.2: using the sets $U_1 = \{-A(a)\}$ and $U_2 = \{+B(b)\}$ we can see now that they are indeed repairs of $\mathcal{A} = \{a : A, b : \neg B, r(a,b)\}$ with respect to $\mathcal{T} = \{\, A \sqsubseteq \forall r.B \,\}$, and that they are the only such repairs.

4.5 Active Inclusion Axioms in $\mathcal{ALC}$ TBoxes

In this section we once again import the idea behind active integrity constraints into DL, but this time we also examine *dynamic* ways an ABox can be repaired via the preferred update actions suggested by the active axioms. To do this we first have to enrich concept inclusions with update actions, indicating the preferred ways to be repaired in case of inconsistency.

We start with the definitions of static and active concept inclusions. The following are reformulations of Definitions 3.1, 3.2 and 3.3.

Definition 4.4. *A concept inclusion of the form* $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq \bot$ *is called a* static *concept inclusion.*

Note that in $\mathcal{ALC}$, any concept inclusion axiom is equivalent to a static concept inclusion.

Definition 4.5. *Let* $r = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq \bot$ *be a static concept inclusion. An* active *concept inclusion is of the form* $\langle r, V \rangle$ *where* V *is a set of indexical update actions such that:*

- *if* $+A \in V$ *then there exists* $C_i = \neg A$.
- *if* $-A \in V$ *then there exists* $C_i = A$.

An active *TBox, denoted by* $\mathsf{a}\mathcal{T}$*, is a set of active concept inclusions.*

For an active concept inclusion $\eta = \langle r, V \rangle$ we let $\mathsf{static}(\eta) = r$ and $\mathsf{active}(\eta) = V$. Note that $\mathsf{active}(\eta)$ can be empty. For an active TBox $\mathsf{a}\mathcal{T}$ we let $\mathsf{static}(\mathsf{a}\mathcal{T}) = \{\mathsf{static}(\eta) : \eta \in \mathsf{a}\mathcal{T}\}$ and $\mathsf{active}(\mathsf{a}\mathcal{T}) = \bigcup_{\eta \in \mathsf{a}\mathcal{T}} \mathsf{active}(\eta)$. We say that $\mathsf{a}\mathcal{T}$ extends $\mathcal{T}$ iff $\mathcal{T}$ is equivalent to $\mathsf{static}(\mathsf{a}\mathcal{T})$.

Going back to the example of Section 4.1, we observe that the active TBox $\mathsf{a}\mathcal{T}$ conforms to the above definitions. Note also that $\mathsf{static}(\mathsf{a}\mathcal{T})$ is equivalent to $\mathcal{T}$. We also recall that we have fixed a satisfiable TBox $\mathcal{T}$ together with a finite set Con of atomic concepts and Ind of individuals. From now on, we also consider a fixed active TBox $\mathsf{a}\mathcal{T}$ that extends $\mathcal{T}$. In the next two subsections, we define the notions of *founded* and *dynamic* repairs of an ABox $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$, which choose among the update actions in $\mathsf{active}(\mathsf{a}\mathcal{T})$ and modify the ABox such that $\mathcal{A}$ is consistent with the concept inclusion axioms of $\mathcal{T}$. Similarly to Chapter 2, the former are based on [Caroprese et al., 2009, Caroprese and Truszczynski, 2011] while the latter are based on [Feuillade and Herzig, 2014].

4.5.1 Founded Weak Repairs and Founded Repairs

We start with the notion of *foundedness*, which is a key condition that a repair should satisfy for its update actions to be supported by the active concept inclusions.

Definition 4.6. *Let $\mathcal{I}$ be an interpretation. A set of update actions U is founded with respect to $\mathsf{a}\mathcal{T}$ and $\mathcal{I}$ if for every $\pm A(a) \in U$ there exists an $\eta \in \mathsf{a}\mathcal{T}$ such that:*

- $\pm A \in \mathsf{active}(\eta)$
- *$\mathcal{I} \diamond U$ is a model of $\mathsf{static}(\eta)$*
- *$\mathcal{I} \diamond (U \setminus \{\pm A(a)\})$ is not a model of $\mathsf{static}(\eta)$*

Based on this, the definitions of a *founded weak repair* and a *founded repair* follow.

Definition 4.7. *Let $\mathsf{static}(\mathsf{a}\mathcal{T}) \wedge \mathcal{A}$ be unsatisfiable. A* founded weak repair *of $\mathcal{A}$ is a consistent set of update actions U such that (1) $\mathcal{T} \wedge (\mathcal{A} \diamond U)$ is satisfiable and (2) there is a model $\mathcal{I}$ of $\mathcal{A}$ such that U is founded with respect to $\mathsf{a}\mathcal{T}$ and $\mathcal{I}$. If moreover $\mathcal{T} \wedge (\mathcal{A} \diamond U')$ is unsatisfiable for all $U' \subset U$, then U is a* founded repair *of $\mathcal{A}$.*

The following simple example showcases this definition.

Example 4.3. *Consider the TBox $\mathcal{T} = \{\mathsf{Born} \sqsubseteq \mathsf{Alive}, \top \sqsubseteq \mathsf{Alive} \sqcup \mathsf{Dead}\}$. Consider also the following active TBox which extends $\mathcal{T}$:*

$$\mathsf{a}\mathcal{T} = \{\langle \mathsf{Born} \sqcap \neg\mathsf{Alive} \sqsubseteq \bot, \{+\mathsf{Alive}\}\rangle, \langle \neg\mathsf{Alive} \sqcap \neg\mathsf{Dead} \sqsubseteq \bot, \{+\mathsf{Dead}\}\rangle\}$$

Furthermore, consider the ABox $\mathcal{A} = \{\mathsf{John} : \mathsf{Born} \sqcap \neg\mathsf{Alive} \sqcap \neg\mathsf{Dead}\}$ which is inconsistent with $\mathcal{T}$. The set $\{+\mathsf{Alive}(\mathsf{John})\}$ is the only founded weak repair of $\mathcal{A}$. Indeed, the second update action in $\{+\mathsf{Alive}(\mathsf{John}), +\mathsf{Dead}(\mathsf{John})\}$ cannot be founded on the second active axiom of $\mathsf{a}\mathcal{T}$. It is also the only founded repair.

In order to find a founded weak repair we have to ensure that the foundedness condition of Definition 4.6 holds after the update actions of the $\mathsf{toggle}^{\pm}(A(a))$ program of the previous section. To do this, we define the following formula:

$$\mathsf{Founded} = \bigwedge_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \left(\left(a : A^{+}\right) \vee \left(a : A^{-}\right) \rightarrow \bigvee_{\substack{\eta \in \mathsf{a}\mathcal{T} \\ \pm A \in \mathsf{active}(\eta)}} \left\langle + A(a) \cup -A(a) \right\rangle \neg\mathsf{static}(\eta) \right)$$

which does exactly that: it checks for all concepts that have been added to or removed from an assertion that they belong to the active part of some concept inclusion and that, without them, the static part of this same concept inclusion is violated.

Using the aforementioned, we can now define the program that searches for founded weak repairs:

$$\mathsf{foundedWeakRepair} = \Big(\bigcup_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \mathsf{toggle}^{\pm}(A(a)) \Big)^{*} ; \mathcal{T}\,? \,; \mathsf{Founded}?$$

as well as the program that searches for founded repairs:

$$\mathsf{foundedRep}(\mathcal{A}) = \mathsf{foundedWeakRepair} \,;\, \mathsf{Minimal}(\mathcal{A})\,?$$

Then the following theorem completes the picture.

Theorem 4.4. *Let $\mathcal{A}$ be inconsistent with respect to* $\mathsf{static}(\mathsf{a}\mathcal{T})$ *and let U be a consistent set of update actions. Furthermore, let no atomic concept A^{+} or A^{-} appear in any of them.*

- *U is a founded weak repair of $\mathcal{A}$ iff there exists an $\mathcal{I}$ such that:*

$$(\mathcal{I}, \mathcal{I} \diamond U) \in \Big\| \Big(\mathcal{A} \wedge \bigwedge_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \big(\neg(a : A^{+}) \wedge \neg(a : A^{-}) \big) \Big) ?; \mathsf{foundedWeakRepair}; \mathsf{mkFalse}^{\pm} \Big\|$$

- *U is a founded repair of $\mathcal{A}$ iff there exists an $\mathcal{I}$ such that:*

$$(\mathcal{I}, \mathcal{I} \diamond U) \in \Big\| \Big(\mathcal{A} \wedge \bigwedge_{\substack{A \in \mathsf{Con} \\ a \in \mathsf{Ind}}} \big(\neg(a : A^{+}) \wedge \neg(a : A^{-}) \big) \Big) \,?\,;\, \mathsf{foundedRep}(\mathcal{A}) \,;\, \mathsf{mkFalse}^{\pm} \Big\|$$

Proof. The construction of the Founded formula is based on the respective one of Section 2.4 for propositional databases and thus the arguments for the foundedWeakRepair and foundedRep($\mathcal{A}$) programs can be easily adapted from the proof of Theorem 2.3. The remaining points are similar to those of Theorem 4.3. □

Going back to Example 4.3 now we can witness how, taking $\mathcal{I}$ to be a model of $\{\mathsf{John} : \mathsf{Born} \sqcap \neg\mathsf{Alive} \sqcap \neg\mathsf{Dead}\}$, the set $\{+\mathsf{Alive}(\mathsf{John})\}$ satisfies both of the conditions of Theorem 4.4. On the other hand, the set $\{+\mathsf{Alive}(\mathsf{John}), +\mathsf{Dead}(\mathsf{John})\}$ doesn't satisfy them for any interpretation $\mathcal{I}$.

4.5.2 Dynamic Weak Repairs and Dynamic Repairs

We now investigate *dynamic* repairs which we have met before in Chapter 2 and which exploit even better the programs and the setting of Dynamic Logic in order to provide a different view of repairs. We begin with some definitions.

For every active concept inclusion $\eta \in \mathsf{a}\mathcal{T}$ and individual a, the programs π^a_η and ${}^{\pm}\pi^a_\eta$ are defined as follows:

$$\begin{aligned}
\pi^a_\eta &= \neg\Big(a : \mathsf{static}(\eta)\Big)\,?\,; \bigcup_{\pm A \in \mathsf{active}(\eta)} \pm A(a) \\
{}^{+}\pi^a_\eta &= \bigcup_{+A \in \mathsf{active}(\eta)} \Big(+A(a)\,; +A^{+}(a) \Big) \\
{}^{-}\pi^a_\eta &= \bigcup_{-A \in \mathsf{active}(\eta)} \Big(-A(a)\,; +A^{-}(a) \Big) \\
{}^{\pm}\pi^a_\eta &= \neg\Big(a : \mathsf{static}(\eta)\Big)\,?\,; \Big({}^{+}\pi^a_\eta \;\cup\; {}^{-}\pi^a_\eta \Big)
\end{aligned}$$

Intuitively, the program π^a_η will check for each active concept inclusion η and individual a whether the static concept inclusion in η is violated at a, and if so, will try to repair it using only update actions that are specified by η. The program ${}^{\pm}\pi^a_\eta$ furthermore stores the concepts that have been changed.

Using the program π^a_η we can now formally define *dynamic weak repairs* and *dynamic repairs*.

Definition 4.8. *Let* $\mathsf{static}(\mathsf{a}\mathcal{T}) \wedge \mathcal{A}$ *be unsatisfiable. A consistent set of update actions* U *is a* dynamic weak repair *of* $\mathcal{A}$ *iff there exists an* $\mathcal{I}$ *such that:*

$$(\mathcal{I}, \mathcal{I} \diamond U) \in \left\| \mathcal{A}\,?\,; \mathbf{while}\,\Big(\neg\mathcal{T}\Big)\,\mathbf{do}\,\Big(\bigcup_{\substack{a \in \mathsf{Ind} \\ \eta \in \mathsf{a}\mathcal{T}}} \pi^a_\eta \Big) \right\|$$

If $\mathcal{T} \wedge (\mathcal{A} \diamond U')$ *is unsatisfiable for all* $U' \subset U$*, then* U *is a* dynamic repair *of* $\mathcal{A}$*.*

It is worth noting that, just as in the propositional case, dynamic weak repairs are not necessarily founded. The next example showcases this.

Example 4.4 (Example 4.3, ctd.)**.** *Consider again the active TBox:*

$$\mathsf{a}\mathcal{T} = \{\langle \mathsf{Born} \sqcap \neg\mathsf{Alive} \sqsubseteq \bot, \{+\mathsf{Alive}\}\rangle, \langle \neg\mathsf{Alive} \sqcap \neg\mathsf{Dead} \sqsubseteq \bot, \{+\mathsf{Dead}\}\rangle\}$$

and the ABox $\mathcal{A} = \{\mathsf{John} : \mathsf{Born} \sqcap \neg\mathsf{Alive} \sqcap \neg\mathsf{Dead}\}$*, whose only founded weak repair was* $\{+\mathsf{Alive}(\mathsf{John})\}$*. There are two dynamic weak repairs of* $\mathcal{A}$*, namely* $U_1 = \{+\mathsf{Alive}(\mathsf{John})\}$ *and* $U_2 = \{+\mathsf{Alive}(\mathsf{John}), +\mathsf{Dead}(\mathsf{John})\}$*. Only* U_1 *is a dynamic repair.*

We now use the program ${}^{\pm}\pi^a_\eta$ together with the formula $\mathsf{Minimal}(\mathcal{A})$ to show how a dynamic repair can be extracted using the familiar procedure of the previous theorem. Defining:

$$\mathsf{dynamicRep}(\mathcal{A}) = \mathbf{while}\left(\neg\mathcal{T}\right)\mathbf{do}\left(\bigcup_{\substack{a\in\mathsf{Ind}\\ \eta\in\mathsf{a}\mathcal{T}}}{}^{\pm}\pi^a_\eta\right);\,\mathsf{Minimal}(\mathcal{A})\,?$$

we have the following theorem.

Theorem 4.5. *Let* $\mathcal{A}$ *be inconsistent with respect to* $\mathsf{static}(\mathsf{a}\mathcal{T})$ *and let* U *be a consistent set of update actions. Furthermore, let no atomic concept* A^+ *or* A^- *appear in any of them.* U *is a dynamic repair of* $\mathcal{A}$ *iff there exists an* $\mathcal{I}$ *such that:*

$$(\mathcal{I}, \mathcal{I}\diamond U) \in \left\|\left(\mathcal{A}\wedge\bigwedge_{\substack{A\in\mathsf{Con}\\ a\in\mathsf{Ind}}}\left(\neg(a:A^+)\wedge\neg(a:A^-)\right)\right)?\,;\,\mathsf{dynamicRep}(\mathcal{A})\,;\mathsf{mkFalse}^{\pm}\right\|$$

Proof. Once again the proof is trivial and based on the definitions, since the only additional thing we do is keep track of the update actions that take place in the course of the **while** program in order to check for minimality with the familiar procedure of the previous theorems. As always, we end the procedure by removing the stored concepts. □

Going back to Example 4.4 this time we can witness how, taking $\mathcal{I}$ to be a model of $\{\mathsf{John} : \mathsf{Born} \sqcap \neg\mathsf{Alive} \sqcap \neg\mathsf{Dead}\}$, only the set $\{+\mathsf{Alive}(\mathsf{John})\}$ satisfies the condition of Theorem 4.5 whereas the set $\{+\mathsf{Alive}(\mathsf{John}), +\mathsf{Dead}(\mathsf{John})\}$ does not for any $\mathcal{I}$.

4.6 Discussion and Conclusion

We have seen how different repairing methods based on preferred update actions of the database literature translate into Description Logics. We have defined a *repair* to be a set of update actions U which, when applied to an ABox, produces a repaired ABox

that follows the active axioms of—and is consistent with—the active TBox. The way to find such a U is the following: starting with a model $\mathcal{I}$ of an ABox $\mathcal{A}$, we can extract a founded or dynamic repair U of $\mathcal{A}$ by finding an $\mathcal{I}'$ such that $(\mathcal{I}, \mathcal{I}') \in ||\mathsf{foundedRep}(\mathcal{A})||$ or $(\mathcal{I}, \mathcal{I}') \in ||\mathsf{dynamicRep}(\mathcal{A})||$, respectively, and setting $U = \{+A(a) : (a : A^+)^{\mathcal{I}'} = \Delta^{\mathcal{I}'}\} \cup \{-A(a) : (a : A^-)^{\mathcal{I}'} = \Delta^{\mathcal{I}'}\}$. The dynamic logic framework on which we rely is useful for not only representing the various repairs but also for constructing them. Constructing the repaired ABox then would amount to: either extract $\mathcal{A} \diamond U$ by reducing the formula $\langle \pi_U \rangle^{\mathsf{c}} \mathcal{A}$ as in Example 4.1 or immediately apply the update U in the initial $\mathcal{ALCO}$ ABox and characterize $\mathcal{A} \diamond U$ via a semantic update of $\mathcal{A}$ by U.

The foundedWeakRepair program is defined in terms of a simple 'generate and test' schema. We just observe here that one might however further exploit dyn$\mathcal{ALCO}$ by replacing foundedWeakRepair by a program that is the dyn$\mathcal{ALCO}$ counterpart of some algorithm computing founded weak repairs.

A slightly different route, more in line with Description Logic, would be to define a *repair* to be any repaired ABox $\mathcal{A}'$ and, given an ABox $\mathcal{A}$, evaluate if $\mathcal{A}'$ is indeed a repair of $\mathcal{A}$ by checking if the formula $\mathcal{A} \wedge \langle \mathsf{repair} \rangle \mathcal{A}'$ is satisfiable, where repair is any of the repair programs defined in the previous sections and used in Theorems 4.3, 4.4 and 4.5. Of course we then would have to *guess* such a repair $\mathcal{A}'$, but the satisfiability check could also be used for assertions instead of whole ABoxes, in order to check if something more specific follows from repairing the initial ABox under the active axioms of an active TBox.

Deciding the existence of founded and dynamic repairs also amounts to satisfiability checking. More specifically, taking repair to be once again any of the repair programs, we can decide the existence of a repair of $\mathcal{A}$ by checking whether the formula $\langle \mathsf{repair} \rangle^{\mathsf{c}} \mathcal{A}$ is satisfiable, where $\mathcal{A}$ is the initial ABox. Indeed, any interpretation that satisfies the formula $\langle \mathsf{repair} \rangle^{\mathsf{c}} \mathcal{A}$ will be a model of at least one repaired ABox.

A limitation of our approach comes from the boolean nature of the active concept inclusions and the fact that complex concepts cannot be paired with a preferred update action in the active part of a concept inclusion. For instance, it is easy to see that, according to Definition 4.5, the TBox $\mathcal{T} = \{\exists \mathsf{hasFather}.\mathsf{Female} \sqsubseteq \bot\}$ can be only extended to

the active TBox $\mathsf{a}\mathcal{T} = \{ \langle \exists \mathsf{hasFather}.\mathsf{Female} \sqsubseteq \bot, \emptyset \rangle \}$. Remember that we talked about this limitation at the end of Chapter 3. At first sight, going through roles to apply the atomic programs or add/remove roles between individuals seems to make the logic undecidable, mainly because of the interplay between these more powerful programs and the Kleene star. Nevertheless, we pursue this direction through a star-free approach in the next chapter.

Summing up, we have taken our first semantic approach to repairing ABoxes with regard to active TBoxes, the latter being an extension of regular TBoxes with update actions in a similar fashion to Chapter 3. To that end we have exploited a dynamic logic framework on which various repairing procedures are introduced and discussed, based on their propositional counterparts of Chapter 2. In the following we investigate a more elaborate logic that builds upon the results achieved in this chapter.

CHAPTER 5

Repairing ABoxes Semantically: the more Elaborate dyn$\mathcal{ALCIO}$

Contents

5.1 Introduction

The semantic approach we take in this chapter is based again on Dynamic Logic, but this time we show how repaired ABoxes (which constitute the *repairs* from now on) can be assessed by complex programs. Since we want to be as general as possible, we once again consider that TBoxes are composed of *concept inclusion axioms* but now we work with the language of $\mathcal{ALCI}$ (without the unique name assumption). As we noted in the introduction of Chapter 4, a first difficulty is that after updating an $\mathcal{ALCI}$ ABox the resulting repair may not be expressible in $\mathcal{ALCI}$ anymore (the same simple example

from [Liu et al., 2011] applies here as well). Consequently, there may exist repairs that we cannot express in order to check if they result from the preferences specified by an active TBox. For this reason the ABoxes we will be working with will also incorporate nominals. Similarly to the previous chapter, both the TBox and the ABox are represented in our language by single formulas.

We continue with the presentation style of the previous chapters and give a glimpse of what we are able to achieve in this one. More specifically, we are able to construct exactly the active TBoxes of Section 1.1 and have their intended meaning as discussed. We recall that the starting TBox $\mathcal{T}$ was the following:

$$\{\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}, \mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot\}$$

The active TBoxes $\mathsf{a}\mathcal{T}_1 = \{\eta_1, \eta_2, \eta_3\}$ and $\mathsf{a}\mathcal{T}_2 = \{\eta_4, \eta_5\}$ then which we want to extend $\mathcal{T}$ have the following form:

$$\eta_1 : +\mathsf{Male} \text{ if } \neg(\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent})$$

$$\eta_2 : +\mathsf{Parent} \text{ if } \neg(\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent})$$

$$\eta_3 : -\mathsf{OnlyChild} \text{ if } \neg(\mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot)$$

$$\eta_4 : -\mathsf{Father} \text{ if } \neg(\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent})$$

$$\eta_5 : -\mathsf{hasSibling}.\top \text{ if } \neg(\mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot)$$

with the preferred update actions on the left part of each active axiom having the respective meaning. This structure for active axioms also enables us to transform them directly into atomic programs. Moreover, contrarily to the first semantic approach that we investigated in the previous chapter, the preferred update actions this time can be applied on any subset of the domain rather than to specific individuals. More precisely, each of them is applied to the subset of the domain containing all individuals which violate its respective axiom.

The chapter is structured as follows. In Section 5.2 we present syntax and semantics of the formulas and programs. In Section 5.3 we show decidability of the logic with the same procedure of reducing it to the fragment which contains no programs. In Section 5.4 we define the notion of *standard repairs* which are not based on preferred update actions

but merely on repairing assertions based on the axioms that they violate. Section 5.5 comprises the definition of this chapter's active TBoxes and Section 5.6 explores *dynamic repairs* in the current setting. Finally, we conclude in Section 5.7 with a discussion around the presented approach.

5.2 Syntax and Semantics

We recall that a *knowledge base* KB $= (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The ABox is constructed from a finite set of concept assertions $a : C$ and role assertions $r(a, b)$ where C is a complex concept of the DL language, r is a role and a, b are individuals. Similarly, the TBox is constructed from a finite set of axioms whose form again depends on the DL at hand. In this chapter we use the language of the description logic $\mathcal{ALCI}$ for TBoxes and its extension $\mathcal{ALCIO}$ with nominals for ABoxes. The TBoxes are composed of concept inclusion axioms and the ABoxes contain (complex) concept assertions $a : C$ and role assertions $r(a, b)$ and $r^{\mathsf{c}}(a, b)$, where C is any concept description in $\mathcal{ALCIO}$, r any role and a, b individuals.

As for the dynamic logic framework, it has PDL-style formulas of the form $\langle\pi\rangle\varphi$ expressing that φ is true after *some* possible execution of the program π. The main differences with PDL are the absence of the Kleene star and the atomic level of the programs: whereas PDL has abstract atomic programs, the atomic programs of our language have the form 'λ if φ' where λ is a program applied on the individuals that satisfy the formula φ. Moreover, the programs of our language can be once again eliminated and every formula is reducible to a *static formula*, i.e., a formula without programs. Last but not least, the models of our logic are identical with the models of Chapter 4, i.e., interpretations of the underlying Description Logic.

5.2.1 Language

Let Con be a countable set of atomic concepts, R a countable set of roles and Ind a countable set of individual names. The language of formulas and programs is defined by the

following grammar:

$$\varphi ::= A \mid \{a\} \mid \bot \mid \varphi \to \varphi \mid \exists r.\varphi \mid \exists r^{\mathsf{c}}.\varphi \mid \mathcal{U}\varphi \mid \langle\gamma\rangle\varphi$$
$$\lambda ::= +A \mid -A \mid -r.\varphi \mid -r^{\mathsf{c}}.\varphi \mid \forall r.\lambda \mid \forall r^{\mathsf{c}}.\lambda$$
$$\gamma ::= \lambda \text{ if } \varphi \mid \gamma;\gamma \mid \gamma \cup \gamma \mid \varphi?$$

where $A \in \mathsf{Con}$, $a \in \mathsf{Ind}$ and $r \in \mathsf{R}$. The logical connectives $\neg$, $\top$, $\wedge$, $\vee$ and $\leftrightarrow$ as well as the universal quantifier $\forall$ are abbreviated in the usual way.

The first level is used to build formulas φ which represent TBoxes and ABoxes. For instance, the formula:

$$\varphi = \mathcal{U}\Big((A \wedge \exists r^{\mathsf{c}}.(A \vee B)) \to \neg A \wedge \forall r.\bot\Big) \wedge \mathcal{U}A \wedge \mathcal{U}\,\forall r.\exists r.B$$

represents the TBox:

$$\mathcal{T} = \{A \sqcap \exists r^{\mathsf{c}}.(A \sqcup B) \sqsubseteq \neg A \sqcap \forall r.\bot, \top \sqsubseteq A, \top \sqsubseteq \forall r.\exists r.B\}$$

and vice versa. Similarly, the formula:

$$\psi = \mathcal{U}\Big(\{a\} \to (A \vee \forall r.B)\Big) \wedge \mathcal{U}\Big(\{b\} \to \exists r^{\mathsf{c}}.\top\Big)$$

represents the ABox $\mathcal{A} = \{a : A \sqcup \forall r.B, b : \exists r^{\mathsf{c}}.\top\}$ and vice versa. So the formula $\varphi \wedge \psi$ represents the KB $(\mathcal{T}, \mathcal{A})$ and vice versa. We recall that $\mathcal{U}$ is the universal operator that makes a formula true in every individual of an interpretation.

In the second level are the programs that are used on individuals. For this reason we call the λ-programs *local*. More specifically, the programs $\pm A$ add or remove the atomic concept A from an individual and the programs $-r.\varphi$ and $-r^{\mathsf{c}}.\varphi$ are used to remove roles: when applied to an individual a they remove all $r(a, b)$ (respectively $r^{\mathsf{c}}(a, b)$) such that b has property φ. We have already seen this in the example of Section 5.1, in the form of the program $-\mathsf{hasSibling}.\top$. Although φ can represent entire KBs or even incorporate programs, in practice when we use the programs $-r.\varphi$ and $-r^{\mathsf{c}}.\varphi$ the formula φ will be the representation of a (possibly complex) concept. Furthermore, the programs $\forall r.\lambda$ and $\forall r^{\mathsf{c}}.\lambda$ are used for going through roles in order to apply a local program λ, i.e., for

an individual a they apply λ to all individuals b such that $r(a, b)$ (respectively $r^{\mathsf{c}}(a, b)$). A more complex example is $\forall r_1.{-}r_2.(B \wedge \exists r_3.A)$ which when applied to an individual a is the program that removes all relations $r_2(b, c)$ for each b such that $r_1(a, b)$ where c satisfies $B \sqcap \exists r_3.A$. Last but not least, the reader may have noticed that there is an asymmetry between the local programs concerning (1) addition of atomic concepts but not roles and (2) application of local programs through universal quantification but not through existential quantification. This is due to some technical limitations that the latter cases would have as well as the existence of scenarios where either the addition of roles or the addition of concepts to only one individual through a role prove to be unintuitive. We will discuss more on this distinction later in the concluding section.

In the third level are the programs that are used on all individuals. For this reason we call the γ-programs *global*. The *atomic programs* have the form λ if φ, whose meaning is: apply the local program λ to all individuals that have property φ. The operators of sequential and nondeterministic composition and the test are the familiar operators of PDL.

We again use some abbreviations here that we have already seen before: we use the expression γ^n as an abbreviation of the program $\gamma; \ldots; \gamma$ where γ appears n times and the expression $\gamma^{\leq n}$ as an abbreviation of the program $\Big(\gamma \cup \top?\Big)^n$. Furthermore, the expression while φ do$^{\leq n}\gamma$ abbreviates the program $(\varphi?; \gamma)^{\leq n}; \neg\varphi?$ and $a : \varphi$ abbreviates the formula $\mathcal{U}(\{a\} \rightarrow \varphi)$. Finally, $r(a, b)$ abbreviates the formula $\mathcal{U}(\{a\} \rightarrow \exists r.\{b\})$ and $r^{\mathsf{c}}(a, b)$ abbreviates the formula $\mathcal{U}(\{a\} \rightarrow \exists r^{\mathsf{c}}.\{b\})$.

Then we identify an $\mathcal{ALCI}$ TBox $\mathcal{T}$ and an $\mathcal{ALCIO}$ ABox $\mathcal{A}$, respectively, with the following formulas:

$$\bigwedge_{C \sqsubseteq D \in \mathcal{T}} \mathcal{U}\Big(C \rightarrow D\Big) \qquad \text{and} \qquad \bigwedge_{a:C \in \mathcal{A}} (a : C) \wedge \bigwedge_{r(a,b) \in \mathcal{A}} r(a, b) \wedge \bigwedge_{r^{\mathsf{c}}(a,b) \in \mathcal{A}} r^{\mathsf{c}}(a, b)$$

Finally, we denote by $\mathsf{Ind}(\mathcal{A})$, $\mathsf{Con}(\mathcal{A})$ and $\mathsf{Role}(\mathcal{A})$ the sets of all individuals, atomic concepts and roles, respectively, that occur in the ABox $\mathcal{A}$.

5.2.2 Semantics

We recall that an interpretation is a couple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ is the interpretation function that maps each atomic concept A to a subset of the domain, each role r to a binary relation on the domain and each individual a to an element of the domain, i.e., $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to inverse roles and complex formulas is defined inductively as follows:

$$
\begin{aligned}
\left(r^{\mathsf{c}}\right)^{\mathcal{I}} &= \{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}}\} \\
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
\bot^{\mathcal{I}} &= \emptyset \\
\left(\varphi \to \psi\right)^{\mathcal{I}} &= \left(\Delta^{\mathcal{I}} \setminus \varphi^{\mathcal{I}}\right) \cup \psi^{\mathcal{I}} \\
\left(\exists r.\varphi\right)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{ there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}} \text{ and } b \in \varphi^{\mathcal{I}}\} \\
\left(\exists r^{\mathsf{c}}.\varphi\right)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{ there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in \left(r^{\mathsf{c}}\right)^{\mathcal{I}} \text{ and } b \in \varphi^{\mathcal{I}}\} \\
\left(\mathcal{U}\varphi\right)^{\mathcal{I}} &= \begin{cases} \Delta^{\mathcal{I}} & \text{if } \varphi^{\mathcal{I}} = \Delta^{\mathcal{I}} \\ \emptyset & \text{otherwise} \end{cases} \\
\left(\langle\gamma\rangle\varphi\right)^{\mathcal{I}} &= \bigcup_{(\mathcal{I},\mathcal{I}') \in \| \gamma \|} \varphi^{\mathcal{I}'}
\end{aligned}
$$

while the semantics of global programs is:

$$
\begin{aligned}
(\mathcal{I},\mathcal{I}') \in \| \pm A \text{ if } \varphi \| &\text{ iff } \Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}, a^{\mathcal{I}'} = a^{\mathcal{I}}, r^{\mathcal{I}'} = r^{\mathcal{I}}, B^{\mathcal{I}'} = B^{\mathcal{I}} \text{ for } B \neq A \text{ and:} \\
& A^{\mathcal{I}'} = \begin{cases} A^{\mathcal{I}} \cup \varphi^{\mathcal{I}} & \text{if } \pm A \text{ is } +A \\ A^{\mathcal{I}} \setminus \varphi^{\mathcal{I}} & \text{if } \pm A \text{ is } -A \end{cases} \\
(\mathcal{I},\mathcal{I}') \in \| -r.\varphi_1 \text{ if } \varphi_2 \| &\text{ iff } \Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}, a^{\mathcal{I}'} = a^{\mathcal{I}}, A^{\mathcal{I}'} = A^{\mathcal{I}}, R^{\mathcal{I}'} = R^{\mathcal{I}} \text{ for } R \neq r \text{ and:} \\
& r^{\mathcal{I}'} = r^{\mathcal{I}} \setminus \{(a,b) \mid a \in \varphi_2^{\mathcal{I}}, b \in \varphi_1^{\mathcal{I}}\} \\
(\mathcal{I},\mathcal{I}') \in \| -r^{\mathsf{c}}.\varphi_1 \text{ if } \varphi_2 \| &\text{ iff } (\mathcal{I},\mathcal{I}') \in \| -r.\varphi_2 \text{ if } \varphi_1 \| \\
(\mathcal{I},\mathcal{I}') \in \| \forall r.\lambda \text{ if } \varphi \| &\text{ iff } (\mathcal{I},\mathcal{I}') \in \| \lambda \text{ if } \exists r^{\mathsf{c}}.\varphi \| \\
(\mathcal{I},\mathcal{I}') \in \| \forall r^{\mathsf{c}}.\lambda \text{ if } \varphi \| &\text{ iff } (\mathcal{I},\mathcal{I}') \in \| \lambda \text{ if } \exists r.\varphi \| \\
(\mathcal{I},\mathcal{I}') \in \| \gamma_1;\gamma_2 \| &\text{ iff there exists } \mathcal{I}'' \text{ such that } (\mathcal{I},\mathcal{I}'') \in \| \gamma_1 \| \text{ and } (\mathcal{I}'',\mathcal{I}') \in \| \gamma_2 \| \\
(\mathcal{I},\mathcal{I}') \in \| \gamma_1 \cup \gamma_2 \| &\text{ iff } (\mathcal{I},\mathcal{I}') \in \| \gamma_1 \| \cup \| \gamma_2 \| \\
(\mathcal{I},\mathcal{I}') \in \| \varphi? \| &\text{ iff } \mathcal{I}' = \mathcal{I} \text{ and } \varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}
\end{aligned}
$$

Remember that the test program is interpreted globally. Furthermore, the semantics for formulas and programs follows the intuition we have given for them in the previous subsection. For example, the following formula:

$$\Big\langle \Big(\forall r.{-}B \text{ if } \neg A \wedge B\Big)\,;\Big({-}r.A \text{ if } \neg A \wedge B\Big)\Big\rangle\ \Big(a : (\neg A \wedge \forall r.\neg B)\Big)$$

is true in an interpretation $\mathcal{I}$ if the assertion $a : \neg A \sqcap \forall r.\neg B$ is true in the interpretation $\mathcal{I}'$, where $\mathcal{I}'$ is the interpretation $\mathcal{I}$ after applying first $\forall r.{-}B$ and second ${-}r.A$ to all individuals of $\mathcal{I}$ that satisfy the concept $\neg A \sqcap B$.

Lastly, let φ be a formula. We say that φ is *(globally) satisfiable* iff there exists an interpretation $\mathcal{I}$ such that $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$. We say that φ is *valid* iff $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every interpretation $\mathcal{I}$. Paralleling the previous chapter, we call the logic that is built using this syntax and semantics dyn$\mathcal{ALCIO}$ since it is an extension of the logic $\mathcal{ALCIO}$ with dynamic logic elements.

5.3 Reduction and Mathematical Properties

This time we convert the formulas of dyn$\mathcal{ALCIO}$ that contain programs to equivalent *static* formulas which contain no programs. Exactly as in Section 4.3, we achieve this reduction by first reducing complex programs to atomic programs and second by eliminating an atomic program which is inside a formula by finding an equivalent formula without it. We start with the reduction axioms for complex programs.

Proposition 5.1. *The following equivalences are valid:*

$$\langle \pi_1 ; \pi_2 \rangle\, \varphi \leftrightarrow \langle \pi_1 \rangle \langle \pi_2 \rangle\, \varphi$$

$$\langle \pi_1 \cup \pi_2 \rangle\, \varphi \leftrightarrow \langle \pi_1 \rangle\, \varphi \vee \langle \pi_2 \rangle\, \varphi$$

$$\langle \psi? \rangle\, \varphi \leftrightarrow \varphi \wedge \mathcal{U}\psi$$

Proof. The proof for all three equivalences is identical to the respective cases on the proof of Proposition 4.1. □

The next three proportions show how to reduce the atomic programs.

Proposition 5.2. *The following equivalences are valid:*

$$\langle \forall r.\lambda \text{ if } \varphi\rangle\psi \leftrightarrow \langle \lambda \text{ if } \exists r^{\mathsf{c}}.\varphi\rangle\psi$$
$$\langle \forall r^{\mathsf{c}}.\lambda \text{ if } \varphi\rangle\psi \leftrightarrow \langle \lambda \text{ if } \exists r.\varphi\rangle\psi$$

where $r \in \mathsf{R}$.

Proof. For an arbitrary interpretation $\mathcal{I}$ we have:

- $\Big(\langle \forall r.\lambda \text{ if } \varphi\rangle\psi\Big)^{\mathcal{I}} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| \forall r.\lambda \text{ if } \varphi \|} \psi^{\mathcal{I}'} \quad = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| \lambda \text{ if } \exists r^{\mathsf{c}}.\varphi \|} \psi^{\mathcal{I}'} = \Big(\langle \lambda \text{ if } \exists r^{\mathsf{c}}.\varphi\rangle\psi\Big)^{\mathcal{I}}$
- $\Big(\langle \forall r^{\mathsf{c}}.\lambda \text{ if } \varphi\rangle\psi\Big)^{\mathcal{I}} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| \forall r^{\mathsf{c}}.\lambda \text{ if } \varphi \|} \psi^{\mathcal{I}'} \quad = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| \lambda \text{ if } \exists r.\varphi \|} \psi^{\mathcal{I}'} = \Big(\langle \lambda \text{ if } \exists r.\varphi\rangle\psi\Big)^{\mathcal{I}}$

□

Proposition 5.3. *The following equivalences are valid:*

$$\langle \lambda \text{ if } \varphi\rangle\,\{a\} \leftrightarrow \{a\}$$
$$\langle \lambda \text{ if } \varphi\rangle\bot \leftrightarrow \bot$$
$$\langle \lambda \text{ if } \varphi\rangle(\psi_1 \to \psi_2) \leftrightarrow \Big(\langle \lambda \text{ if } \varphi\rangle\,\psi_1 \to \langle \lambda \text{ if } \varphi\rangle\,\psi_2\Big)$$
$$\langle \lambda \text{ if } \varphi\rangle\exists r.\psi \leftrightarrow \begin{cases} \Big(\neg\varphi \wedge \exists r.\langle \lambda \text{ if } \varphi\rangle\psi\Big) \vee \Big(\varphi \wedge \exists r.(\neg\varphi' \wedge \langle \lambda \text{ if } \varphi\rangle\psi)\Big) & \textit{if } \ \lambda{=}{-}r.\varphi' \\ \langle -r.\varphi \text{ if } \varphi'\rangle\exists r.\psi & \textit{if } \ \lambda{=}{-}r^{\mathsf{c}}.\varphi' \\ \exists r.\langle \lambda \text{ if } \varphi\rangle\,\psi & \textit{otherwise} \end{cases}$$
$$\langle \lambda \text{ if } \varphi\rangle\exists r^{\mathsf{c}}.\psi \leftrightarrow \begin{cases} \Big(\neg\varphi' \wedge \exists r^{\mathsf{c}}.\langle \lambda \text{ if } \varphi\rangle\psi\Big) \vee \Big(\varphi' \wedge \exists r^{\mathsf{c}}.(\neg\varphi \wedge \langle \lambda \text{ if } \varphi\rangle\psi)\Big) & \textit{if } \ \lambda{=}{-}r.\varphi' \\ \langle -r.\varphi \text{ if } \varphi'\rangle\exists r^{\mathsf{c}}.\psi & \textit{if } \ \lambda{=}{-}r^{\mathsf{c}}.\varphi' \\ \exists r^{\mathsf{c}}.\langle \lambda \text{ if } \varphi\rangle\,\psi & \textit{otherwise} \end{cases}$$
$$\langle \lambda \text{ if } \varphi\rangle\mathcal{U}\psi \leftrightarrow \mathcal{U}\langle \lambda \text{ if } \varphi\rangle\psi$$

where $a \in \mathsf{Ind}$, $r \in \mathsf{R}$ *and* λ *has one of the following forms:* $+A$, $-A$, $-r.\varphi$, $-r^{\mathsf{c}}.\varphi$.

Proof. The first three equivalences as well as the last one are proved similarly to the respective cases of Proposition 4.2. Regarding the remaining two, suppose $\mathcal{I}$ is an arbitrary interpretation. Then we have that:

- $\Big(\langle -r.\varphi' \text{ if } \varphi\rangle \exists r.\psi\Big)^{\mathcal{I}} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| -r.\varphi' \text{ if } \varphi \|} (\exists r.\psi)^{\mathcal{I}'} = (\exists r.\psi)^{\mathcal{I}'}$ for the unique interpretation $\mathcal{I}'$ such that $(\mathcal{I},\mathcal{I}') \in \| -r.\varphi' \text{ if } \varphi \|$. By the definitions then $(\exists r.\psi)^{\mathcal{I}'} =$

$$\{a \in \Delta^{\mathcal{I}'} \mid \text{there is } b \in \Delta^{\mathcal{I}'} \text{ such that } (a,b) \in r^{\mathcal{I}'} \text{ and } b \in \psi^{\mathcal{I}'}\} =$$

$$\{a \in \Delta^{\mathcal{I}} \mid \text{there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}} \setminus \{(c,d) \mid c \in \varphi^{\mathcal{I}}, d \in \varphi'^{\mathcal{I}}\}$$

$$\text{and } b \in \bigcup_{(\mathcal{I},\mathcal{I}')\in \| -r.\varphi' \text{ if } \varphi \|} \psi^{\mathcal{I}'}\} =$$

$$\{a \in \Delta^{\mathcal{I}} \setminus \varphi^{\mathcal{I}} \mid \text{there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}} \text{ and } b \in \Big(\langle -r.\varphi' \text{ if } \varphi\rangle \psi\Big)^{\mathcal{I}}\} \cup$$

$$\{a \in \varphi^{\mathcal{I}} \mid \text{there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}} \text{ and } b \notin \varphi'^{\mathcal{I}}$$

$$\text{and } b \in \Big(\langle -r.\varphi' \text{ if } \varphi\rangle \psi\Big)^{\mathcal{I}}\} =$$

$$\Big(\big(\Delta^{\mathcal{I}} \setminus \varphi^{\mathcal{I}}\big) \cap \Big(\exists r.\langle -r.\varphi' \text{ if } \varphi\rangle \psi\Big)^{\mathcal{I}}\Big) \cup$$

$$\Big(\varphi^{\mathcal{I}} \cap \{a \in \Delta^{\mathcal{I}} \mid \text{there is } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}}$$

$$\text{and } b \in \big(\Delta^{\mathcal{I}} \setminus \varphi'^{\mathcal{I}}\big) \cap \Big(\langle -r.\varphi' \text{ if } \varphi\rangle \psi\Big)^{\mathcal{I}}\}\Big) =$$

$$\Big(\big(\neg\varphi\big)^{\mathcal{I}} \cap \Big(\exists r.\langle -r.\varphi' \text{ if } \varphi\rangle \psi\Big)^{\mathcal{I}}\Big) \cup \Big(\varphi^{\mathcal{I}} \cap \Big(\exists r.(\neg\varphi' \wedge \langle -r.\varphi' \text{ if } \varphi\rangle \psi)\Big)^{\mathcal{I}}\Big) =$$

$$\Big(\big(\neg\varphi \wedge \exists r.\langle -r.\varphi' \text{ if } \varphi\rangle \psi\big) \vee \big(\varphi \wedge \exists r.(\neg\varphi' \wedge \langle -r.\varphi' \text{ if } \varphi\rangle \psi)\big)\Big)^{\mathcal{I}}$$

- $\Big(\langle -r^{\mathsf{c}}.\varphi' \text{ if } \varphi\rangle \exists r.\psi\Big)^{\mathcal{I}} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| -r^{\mathsf{c}}.\varphi' \text{ if } \varphi \|} (\exists r.\psi)^{\mathcal{I}'} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| -r.\varphi \text{ if } \varphi' \|} (\exists r.\psi)^{\mathcal{I}'} = \Big(\langle -r.\varphi \text{ if } \varphi'\rangle \exists r.\psi\Big)^{\mathcal{I}}$
- Otherwise, λ has one of the following forms: $+A$, $-A$, $-R.\varphi$, $-R^{\mathsf{c}}.\varphi$ where $R \neq r$. It is easy then to see that $\Big(\langle \lambda \text{ if } \varphi\rangle \exists r.\psi\Big)^{\mathcal{I}} = \Big(\exists r.\langle \lambda \text{ if } \varphi\rangle\ \psi\Big)^{\mathcal{I}}$ based on the definitions.
- Similarly for $\langle \lambda \text{ if } \varphi\rangle \exists r^{\mathsf{c}}.\psi$

□

Proposition 5.4. *The following equivalences are valid:*

$$\langle +A \text{ if } \varphi\rangle\ B \leftrightarrow \begin{cases} A \vee \varphi & \textit{if } A = B \\ B & \textit{otherwise} \end{cases}$$

$$\langle -A \text{ if } \varphi\rangle\ B \leftrightarrow \begin{cases} A \wedge \neg\varphi & \textit{if } A = B \\ B & \textit{otherwise} \end{cases}$$

$$\langle -r.\varphi_1 \text{ if } \varphi_2\rangle\ A \leftrightarrow A$$

$$\langle -r^{\mathsf{c}}.\varphi_1 \text{ if } \varphi_2\rangle\ A \leftrightarrow A$$

where $A, B \in \mathsf{Con}$ *and* $r \in \mathsf{R}$.

Proof. The first two equivalences are once again similar to those of Proposition 4.2. For the other two, let $\mathcal{I}$ be an arbitrary interpretation. We have:

- $\Big(\langle -r.\varphi_1 \text{ if } \varphi_2\rangle\, A\Big)^{\mathcal{I}} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| -r.\varphi_1 \text{ if } \varphi_2 \|} A^{\mathcal{I}'} = A^{\mathcal{I}'}$ for the unique interpretation $\mathcal{I}'$ such that $(\mathcal{I},\mathcal{I}') \in \| -r.\varphi_1 \text{ if } \varphi_2 \|$. By definition then $A^{\mathcal{I}'} = A^{\mathcal{I}}$
- $\Big(\langle -r^{\mathsf{c}}.\varphi_1 \text{ if } \varphi_2\rangle\, A\Big)^{\mathcal{I}} = \bigcup_{(\mathcal{I},\mathcal{I}')\in \| -r.\varphi_2 \text{ if } \varphi_1 \|} A^{\mathcal{I}'} = A^{\mathcal{I}'}$ for the unique interpretation $\mathcal{I}'$ such that $(\mathcal{I},\mathcal{I}') \in \| -r.\varphi_2 \text{ if } \varphi_1 \|$. By definition again $A^{\mathcal{I}'} = A^{\mathcal{I}}$

□

The next example showcases such a reduction using many of these reduction axioms.

Example 5.1. *Consider the atomic program* $\forall r.-A$ if B *and the assertion* $a : \neg A \sqcap \neg B$ *which is represented by the formula* $\mathcal{U}\Big(\{a\} \to (\neg A \wedge \neg B)\Big)$. *We show how to reduce the formula* $\varphi = \langle \forall r.-A \text{ if } B\rangle\, \mathcal{U}\Big(\{a\} \to (\neg A \wedge \neg B)\Big)$ *to a static one by means of the equivalences of Propositions 5.2, 5.3 and 5.4. First,* φ *is equivalent to* $\langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, \mathcal{U}\Big(\{a\} \to (\neg A \wedge \neg B)\Big)$ *which is in turn equivalent to* $\mathcal{U}\Big(\{a\} \to \langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, (\neg A \wedge \neg B)\Big)$. *Second, we have that* $\langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, (\neg A \wedge \neg B)$ *is equivalent to* $\neg \langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, A \wedge \neg \langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, B$. *Furthermore,* $\langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, A$ *is equivalent to* $A \wedge \neg \exists r^{\mathsf{c}}.B$ *and* $\langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, B$ *is equivalent to* B. *These give then that:*

$$\begin{aligned} \langle -A \text{ if } \exists r^{\mathsf{c}}.B\rangle\, (\neg A \wedge \neg B) &\leftrightarrow \neg(A \wedge \neg \exists r^{\mathsf{c}}.B) \wedge \neg B \\ &\leftrightarrow (\neg A \vee \exists r^{\mathsf{c}}.B) \wedge \neg B \end{aligned}$$

So φ *is equivalent to the formula* $\mathcal{U}\Big(\{a\} \to (\neg A \vee \exists r^{\mathsf{c}}.B) \wedge \neg B\Big)$ *which represents the assertion* $a : (\neg A \sqcup \exists r^{\mathsf{c}}.B) \sqcap \neg B$. *This means that the assertion* $a : (\neg A \sqcup \exists r^{\mathsf{c}}.B) \sqcap \neg B$ *is equivalent to the assertion* $a : \neg A \sqcap \neg B$ *after executing the atomic program* $\langle \forall r.-A \text{ if } B\rangle$ *which is clearly the expected outcome.*

From Propositions 5.1, 5.2, 5.3 and 5.4 we then obtain the following theorem.

Theorem 5.1. *Every formula of* dyn$\mathcal{ALCIO}$ *is equivalent to a static formula.*

Proof. The procedure is the same as in Theorem 4.1: given a dyn$\mathcal{ALCIO}$-formula φ, iterative applications of the equivalences in Proposition 5.1 will result in a formula without complex programs that is equivalent to φ. With the equivalences of Proposition 5.2 then all local programs of the form $\forall r.\lambda$ and $\forall r^{\mathsf{c}}.\lambda$ are eliminated from within the atomic programs. Next, using Proposition 5.3 the atomic programs can be pushed through the boolean connectives, the universal and existential quantifiers, as well as the universal operator. Finally, all of the atomic programs are eliminated by applying iteratively the equivalences of Proposition 5.4 together with the first equivalence of Proposition 5.3. Since all equivalences are valid we obtain a static formula that is equivalent to φ. □

Through Theorem 5.1 we can now reduce every formula of the logic which we work with to the static fragment of the logic which contains no programs. Decidability of global satisfiability in dyn$\mathcal{ALCIO}$ then follows using the same argument of the previous chapter: any static formula can be mapped to an $\mathcal{ALCIO}(\mathcal{U})$-concept and vice versa and concept satisfiability in $\mathcal{ALCIO}(\mathcal{U})$ is decidable [Horrocks et al., 2006]. The mapping τ that achieves the correspondence between static dyn$\mathcal{ALCIO}$-formulas and $\mathcal{ALCIO}(\mathcal{U})$-concepts extends the previous mapping in the obvious way (where we once again use the connectives $\neg$ and $\vee$ in the place of $\bot$ and $\rightarrow$), i.e.:

- $\tau(A) = A$
- $\tau(\{a\}) = \{a\}$
- $\tau(\neg\varphi) = \neg\tau(\varphi)$
- $\tau(\varphi \vee \psi) = \tau(\varphi) \sqcup \tau(\psi)$
- $\tau(\exists r.\varphi) = \exists r.\tau(\varphi)$
- $\tau(\exists r^{\mathsf{c}}.\varphi) = \exists r^{\mathsf{c}}.\tau(\varphi)$
- $\tau(\mathcal{U}\varphi) = \forall r_U.\tau(\varphi)$

where $A \in \mathsf{Con}$, $a \in \mathsf{Ind}$, and $r \in \mathsf{R}$ (with r_U the universal role). Then (global) satisfiability checking of the static fragment of dyn$\mathcal{ALCIO}$ is similarly reduced to the satisfiability problem of $\mathcal{ALCIO}(\mathcal{U})$ and we obtain the following theorem.

Theorem 5.2. *Global satisfiability in the logic* dyn$\mathcal{ALCIO}$ *is decidable.*

Both the proof of Theorem 5.2 as well as the lemma that it is based on are identical to the respective of Section 4.3 and are omitted.

5.3.1 Associating Local Programs with TBox Axioms

We can now make use of the language of dyn$\mathcal{ALCIO}$ and, as a first step, show how to build a (non-dynamic) repairing procedure that checks whether an assertion or ABox can be obtained as a result of repairing an initial ABox that is inconsistent with a TBox through a bounded number of steps. Through this problem we showcase the flexibility of the programs that can be built using our dynamic logic framework. We recall that an ABox $\mathcal{A}$ is inconsistent with a TBox $\mathcal{T}$ iff there is no model of $(\mathcal{T}, \mathcal{A})$, i.e., iff the formula $\mathcal{T} \wedge \mathcal{A}$ is (globally) unsatisfiable in our logic. Furthermore, we consider for the rest of the chapter that all TBoxes we work with are consistent and all ABoxes we work with are consistent: we focus then on cases where their conjunction is inconsistent due to the interaction between them. We start by defining the set of update actions that each TBox axiom can be paired with.

Definition 5.1. *Let ρ be any concept inclusion axiom. The set* $\mathsf{pr}(\rho)$ *of programs associated with ρ is defined inductively as follows:*

$$
\begin{aligned}
\mathsf{pr}(A) &= \{+A, -A\} \\
\mathsf{pr}(\neg C) &= \mathsf{pr}(C) \\
\mathsf{pr}(C_1 \sqcap C_2) &= \mathsf{pr}(C_1) \cup \mathsf{pr}(C_2) \\
\mathsf{pr}(\forall r.C) &= \{-r.\neg C\} \cup \{\forall r.\lambda \mid \lambda \in \mathsf{pr}(C)\} \\
\mathsf{pr}(\forall r^{\mathsf{c}}.C) &= \{-r^{\mathsf{c}}.\neg C\} \cup \{\forall r^{\mathsf{c}}.\lambda \mid \lambda \in \mathsf{pr}(C)\} \\
\mathsf{pr}(C \sqsubseteq D) &= \mathsf{pr}(C) \cup \mathsf{pr}(D)
\end{aligned}
$$

The set $\mathsf{pr}(\rho)$ is defined to contain all local programs that can be constructed from each $\rho \in \mathcal{T}$ based on the intuitions given in Section 5.2. For example, consider the concept inclusion $\rho = A \sqsubseteq \forall r.\neg B$. According to Definition 5.1, the set $\mathsf{pr}(\rho)$ comprises the

following programs: $+A$, $-A$, $-r.B$, $\forall r.+B$, $\forall r.-B$. Clearly, not all of them can provide a repairing solution. For example, consider the assertion $a : A \sqcap \exists r.B$ which is not consistent with ρ. The programs $+A$ and $\forall r.+B$ when applied to a do not provide a repaired assertion because the sources of the inconsistency are not tackled by these programs, while the programs $-A$, $-r.B$ and $\forall r.-B$ do so. So the set $\mathsf{pr}(\rho)$, although finite, contains arbitrary local programs constructed from ρ many of which may not be suitable when repairing an assertion or ABox that is inconsistent with ρ. This does not pose a threat though as that ineffective subset of $\mathsf{pr}(\rho)$ can be ignored during the repairing procedure or, in the worst case, raise the number of computations needed without modifying the result.

5.4 Standard Repairs

Given the set $\mathsf{pr}(\rho)$ we can now define the program that tries to repair an ABox that is inconsistent with a TBox $\mathcal{T}$ by updating the individuals which violate some $\rho \in \mathcal{T}$ with random update actions based on the concepts and roles of each $\rho \in \mathcal{T}$. Since this program does not depend on any active axioms (that we will introduce in the next section) and since there is no preference between the update actions chosen, we call it a *standard repair program*.

Definition 5.2. *Let $\mathcal{T}$ be a TBox and $n \in \mathbb{N}$. The* standard repair program up to n *of $\mathcal{T}$ is the following program:*

$$\pi^n_{\mathcal{T}} = \Big(\bigcup_{\substack{\rho\in\mathcal{T}\\ \lambda\in\mathsf{pr}(\rho)}} \big(\lambda \text{ if } \neg\rho\big)\Big)^{\leq n} ; \mathcal{T}?$$

It is easy to see what this program does: it applies at most n times update actions from the axioms of $\mathcal{T}$ that are violated until $\mathcal{T}$ is ultimately satisfied. So now we are in a position to take advantage of the atomic programs of our logic to build decision procedures that can check whether specific ABoxes are repairs or not. More specifically, if $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox that is inconsistent with $\mathcal{T}$, we can decide if an assertion or ABox $\mathcal{A}'$ is the outcome of repairing $\mathcal{A}$ (by using at most n computations). This is possible by checking if the formula $\mathcal{A} \wedge \langle \pi^n_{\mathcal{T}} \rangle \mathcal{A}'$ is satisfiable in our logic and is articulated in the next proposition.

Proposition 5.5. *Let $\mathcal{T}$ be a TBox, let $\mathcal{A}$ be an ABox, let $\mathcal{A}'$ be either an assertion or an ABox and let $n \in \mathbb{N}$. If the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^n \rangle \mathcal{A}'$ is satisfiable then $\mathcal{T} \wedge \mathcal{A}'$ is consistent.*

Proof. Satisfiability of the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^n \rangle \mathcal{A}'$ implies that there exists an interpretation $\mathcal{I}$ such that $\left(\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^n \rangle \mathcal{A}'\right)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and more specifically $\bigcup_{(\mathcal{I},\mathcal{I}') \in \| \pi_{\mathcal{T}}^n \|} \mathcal{A}'^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ (1). In general, since the formula $\mathcal{A}$ is a conjunction of subformulas all of which begin with the universal operator, it holds that either $\mathcal{A}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ or $\mathcal{A}^{\mathcal{I}} = \emptyset$ for all $\mathcal{I}$. Thus, by (1) there exists an $\mathcal{I}'$ such that $(\mathcal{I}, \mathcal{I}') \in \| \pi_{\mathcal{T}}^n \|$ and $\mathcal{A}'^{\mathcal{I}'} = \Delta^{\mathcal{I}}$. Furthermore, if $(\mathcal{I}, \mathcal{I}') \in \| \pi_{\mathcal{T}}^n \|$ then $(\mathcal{I}', \mathcal{I}') \in \left\| \mathcal{T}? \right\|$. This means that there exists an $\mathcal{I}'$ such that $\mathcal{A}'^{\mathcal{I}'} = \Delta^{\mathcal{I}}$, $\mathcal{T}^{\mathcal{I}'} = \Delta^{\mathcal{I}'}$ and $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ which gives that $\left(\mathcal{T} \wedge \mathcal{A}'\right)^{\mathcal{I}'} = \Delta^{\mathcal{I}'}$, i.e., $\mathcal{T} \wedge \mathcal{A}'$ is consistent. □

The reader may wonder whether we could just check satisfiability of the formula $\mathcal{T} \wedge \mathcal{A}'$ in order to see if $\mathcal{A}'$ is a repair of $\mathcal{A}$. As we show in Example 5.2 below, through the satisfiability of the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^n \rangle \mathcal{A}'$ we can also check if $\mathcal{A}'$ is a repair of $\mathcal{A}$ that results from updating the individuals according to the axioms they violate, so $\mathcal{A}'$ is not any random ABox. Furthermore, we know that we can achieve this with at most n computations.

We continue with the definition of *relevance* between two ABoxes.

Definition 5.3. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two ABoxes. We say that $\mathcal{A}'$ is* relevant *to $\mathcal{A}$ iff:*

- $\mathsf{Ind}(\mathcal{A}') \subseteq \mathsf{Ind}(\mathcal{A})$
- $\mathsf{Con}(\mathcal{A}') \subseteq \mathsf{Con}(\mathcal{A})$
- $\mathsf{Role}(\mathcal{A}') \subseteq \mathsf{Role}(\mathcal{A})$
- *if $r(a, b) \in \mathcal{A}'$ then $r(a, b) \in \mathcal{A}$*

The role of relevance is to make sure that a repair of an ABox does not contain irrelevant concept and role assertions. For instance, when repairing the ABox $\mathcal{A} = \{a : \neg A\}$ which is inconsistent with the TBox $\mathcal{T} = \{\top \sqsubseteq A\}$, the ABox $\mathcal{A}' = \{r(a, b), b : B\}$ should not be a repair of $\mathcal{A}$, although the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^1 \rangle \mathcal{A}'$ is satisfiable, because both of its assertions have nothing to do with $\mathcal{A}$.

Next is the main definition of this section, viz. that of a *standard repair*.

Definition 5.4. *Let $\mathcal{T}$ be a TBox and let $\mathcal{A}$ be an ABox. An ABox $\mathcal{A}'$ is a* standard repair *of $\mathcal{A}$ with respect to $\mathcal{T}$ iff $\mathcal{A}'$ is relevant to $\mathcal{A}$ and the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{n} \rangle \mathcal{A}'$ is satisfiable for some $n \in \mathbb{N}$.*

So, in other words, a standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$ is an ABox that can be built from updating assertions of $\mathcal{A}$ so that they conform with the axioms of $\mathcal{T}$ and that doesn't contain any irrelevant (with respect to $\mathcal{A}$) concept and role assertions or new individuals. Note that since the program $\pi_{\mathcal{T}}^{n}$ can apply quite arbitrary update actions from the set $\mathsf{pr}(\rho)$ for each $\rho \in \mathcal{T}$, standard repairs can diverge more from the initial ABox than desired. A simple example that highlights this follows.

Example 5.2. *Consider the following TBox and ABox:*

$$\mathcal{T} = \{\mathsf{Male} \sqcap \mathsf{Person} \sqsubseteq \mathsf{Man}, \mathsf{Man} \sqsubseteq \forall \mathsf{hasSpouse}.\neg\mathsf{Man}, \neg\mathsf{Man} \sqsubseteq \forall \mathsf{hasSpouse}.\mathsf{Man}\}$$

$$\mathcal{A} = \{\mathsf{John} : \mathsf{Man}, \mathsf{Peter} : \mathsf{Man}, \mathsf{hasSpouse}(\mathsf{John}, \mathsf{Peter}), \mathsf{hasSpouse}(\mathsf{Peter}, \mathsf{John})\}$$

For the ABox $\mathcal{A}_1 = \{\mathsf{John} : \mathsf{Man}, \mathsf{Peter} : \mathsf{Man}\}$ we have that $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{1} \rangle \mathcal{A}_1$ is satisfiable using the atomic program:

$$-\mathsf{hasSpouse}.\mathsf{Man} \text{ if } \neg\,(\mathsf{Man} \sqsubseteq \forall \mathsf{hasSpouse}.\neg\mathsf{Man})$$

and thus, according to Proposition 5.5 and Definition 5.4, $\mathcal{A}_1$ is consistent with $\mathcal{T}$ and is also a standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$. If we just checked satisfiability of the formula $\mathcal{T} \wedge \mathcal{A}_2$ though, where:

$$\mathcal{A}_2 = \{\mathsf{John}{:}\neg\mathsf{Person}, \mathsf{Peter}{:}\neg\mathsf{Person}, \mathsf{hasSpouse}(\mathsf{John}, \mathsf{Peter}), \mathsf{hasSpouse}(\mathsf{Peter}, \mathsf{John})\}$$

then we would have that $\mathcal{A}_2$ is also a repair of $\mathcal{A}$ with respect to $\mathcal{T}$, although the way to obtain $\mathcal{A}_2$ from $\mathcal{A}$ is irrelevant with the axioms of $\mathcal{T}$ that were violated by $\mathcal{A}$ and seems rather undesirable. Last but not least, according to Definition 5.4 a standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$ is also the ABox $\mathcal{A}_3 = \{\mathsf{John} : \neg\mathsf{Man}, \mathsf{Peter} : \neg\mathsf{Man}\}$ using the atomic programs:

$$\forall \mathsf{hasSpouse}.-\mathsf{Man} \text{ if } \neg\,(\mathsf{Man} \sqsubseteq \forall \mathsf{hasSpouse}.\neg\mathsf{Man})$$

$$-\mathsf{hasSpouse}.\neg\mathsf{Man} \text{ if } \neg\,(\neg\mathsf{Man} \sqsubseteq \forall \mathsf{hasSpouse}.\mathsf{Man})$$

which, although consistent with $\mathcal{T}$ and a standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$, is also a debatable repair.

So, although we can check whether a possible repair of the initial ABox can be achieved using a bounded number of computations, this procedure can also provide other, quite undesirable ABoxes through the application of too many update actions to the initial ABox. We can separate then the standard repairs into two classes: those that are achieved through a *minimal* number of computations and those that are not. The next definition singles out the classes of standard repairs based on the minimum number of computations required.

Definition 5.5. *Let $\mathcal{T}$ be a TBox and let $\mathcal{A}$ be an ABox that is inconsistent with $\mathcal{T}$. The set of all standard repairs of $\mathcal{A}$ with respect to $\mathcal{T}$ is denoted by* $\mathsf{StandRep}(\mathcal{T}, \mathcal{A})$. *The subset of* $\mathsf{StandRep}(\mathcal{T}, \mathcal{A})$ *comprising the standard repairs that can be achieved with a minimum of n computations is denoted by* $\mathsf{StandRep}^n(\mathcal{T}, \mathcal{A})$.

Given Definition 5.5, the way to check if an ABox belongs to the set $\mathsf{StandRep}^n(\mathcal{T}, \mathcal{A})$ for some $n \in \mathbb{N}$ is given next.

Proposition 5.6. *Let $\mathcal{T}$ be a TBox and let $\mathcal{A}$ be an ABox that is inconsistent with $\mathcal{T}$. For an ABox $\mathcal{A}'$ then, $\mathcal{A}' \in \mathsf{StandRep}^n(\mathcal{T}, \mathcal{A})$ iff:*

- *$\mathcal{A}'$ is relevant to $\mathcal{A}$*
- *the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^n \rangle \, \mathcal{A}'$ is satisfiable*
- *the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{n-1} \rangle \, \mathcal{A}'$ is unsatisfiable*

Proof. For the left to right direction, if $\mathcal{A}' \in \mathsf{StandRep}^n(\mathcal{T}, \mathcal{A})$ then $\mathcal{A}'$ is a standard repair of $\mathcal{A}$ and thus $\mathcal{A}'$ is relevant to $\mathcal{A}$ as well as the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^n \rangle \, \mathcal{A}'$ is satisfiable for some $n \in \mathbb{N}$. The fact that $\mathcal{A}'$ can be achieved with a minimum of n computations means that $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^m \rangle \, \mathcal{A}'$ is satisfiable for all $m \geq n$. Furthermore, satisfiability of $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{n-1} \rangle \, \mathcal{A}'$ would obviously mean that $\mathcal{A}'$ can also be achieved with $n - 1$ computations and $\mathcal{A}' \notin \mathsf{StandRep}^n(\mathcal{T}, \mathcal{A})$. Thus $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{n-1} \rangle \, \mathcal{A}'$ is unsatisfiable.

For the right to left direction, the first two items give that $\mathcal{A}'$ is a standard repair of $\mathcal{A}$ that can be achieved within n steps. By the third item, for all $m < n-1$ the formulas $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{m} \rangle \mathcal{A}'$ are also unsatisfiable since a repair that can be achieved with at most m steps can also be achieved with more than m (imagine the rest of the steps being the '$\top?$' programs in the definition of '$\gamma^{\leq n}$'). This means that $\mathcal{A}'$ needs at least n steps and $\mathcal{A}' \in \mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A})$. □

As mentioned in this proof, if the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{n} \rangle \mathcal{A}'$ is satisfiable then the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{m} \rangle \mathcal{A}'$ is also satisfiable for any $m > n$. Furthermore, there exists a minimal $k \in \mathbb{N}$ such that the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k} \rangle \mathcal{A}'$ is satisfiable and this k is the minimum number of steps needed to obtain $\mathcal{A}'$ from $\mathcal{A}$. Based on this we have the following proposition.

Proposition 5.7. *For every TBox $\mathcal{T}$ and ABox $\mathcal{A}$, either* $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) = \emptyset$ *or there exists an* $n \in \mathbb{N}$ *such that:*

- $\mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A}) \neq \emptyset$
- $\mathsf{StandRep}^{m}(\mathcal{T}, \mathcal{A}) = \emptyset$ *for all* $m < n$

Proof. Let $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) \neq \emptyset$ (1). By Proposition 5.6 it is easy to see that the sets $\mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A})$, much like the sets $\mathcal{R}_n^{\mathcal{A}}$ of Chapter 3, are pairwise disjoint and their union is the set $\mathsf{StandRep}(\mathcal{T}, \mathcal{A})$ of all standard repairs of $\mathcal{A}$ with respect to $\mathcal{T}$. Therefore, $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) = \bigcup_{n=0}^{\infty} \mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A})$ and by (1) $\bigcup_{n=0}^{\infty} \mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A}) \neq \emptyset$. It follows easily that there exists an $m \in \mathbb{N}$ such that:

$$\bigcup_{n=0}^{\infty} \mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A}) = \bigcup_{n=m}^{\infty} \mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A}) \quad \text{and} \quad \mathsf{StandRep}^{m}(\mathcal{T}, \mathcal{A}) \neq \emptyset$$

which also gives that $\mathsf{StandRep}^{k}(\mathcal{T}, \mathcal{A}) = \emptyset$ for all $k < m$. □

Non-existence of a repair can be the case when there is an inconsistency that cannot be repaired because the sets $\mathsf{pr}(\rho)$ do not provide any adequate update actions based on Definition 5.1. Consider e.g. the TBox $\mathcal{T} = \{\top \sqsubseteq \exists r.\top\}$ and the ABox $\mathcal{A} = \{a : \forall r.\bot\}$. Then $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) = \emptyset$. We will come back to this later in the concluding discussion.

For every $\mathcal{T}$ and $\mathcal{A}$ with $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) \neq \emptyset$ we denote the set $\mathsf{StandRep}^{n}(\mathcal{T}, \mathcal{A})$ satisfying Proposition 5.7 by $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$. This is the set that comprises the

standard repairs which require the least amount of computations. We thus obtain the definition of *minimal standard repairs*.

Definition 5.6. *Let $\mathcal{T}$ be a TBox and let $\mathcal{A}$ be an ABox that is inconsistent with $\mathcal{T}$. An ABox $\mathcal{A}'$ is a* minimal standard repair *of $\mathcal{A}$ with respect to $\mathcal{T}$ iff $\mathcal{A}' \in \mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$.*

Going back to Example 5.2 then we can witness that $\mathcal{A}_1 \in \mathsf{StandRep}^1(\mathcal{T}, \mathcal{A})$ whereas $\mathcal{A}_3 \in \mathsf{StandRep}^2(\mathcal{T}, \mathcal{A})$, and since $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A}) = \mathsf{StandRep}^1(\mathcal{T}, \mathcal{A})$ we conclude that $\mathcal{A}_1$ is a minimal standard repair whereas $\mathcal{A}_3$ is not. Furthermore, $\mathcal{A}_2 \notin \mathsf{StandRep}(\mathcal{T}, \mathcal{A})$. Therefore, there is a way to avoid debatable repairs like $\mathcal{A}_3$ by searching only for minimal standard repairs. The way to obtain the minimum number of computations required for each repairing procedure is given in the following proposition.

Proposition 5.8. *Let $\mathcal{T}$ be a TBox and let $\mathcal{A}$ be an ABox that is inconsistent with $\mathcal{T}$. We have that $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A}) = \mathsf{StandRep}^k(\mathcal{T}, \mathcal{A})$ iff:*

- *the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^k \rangle \top$ is satisfiable*
- *the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \top$ is unsatisfiable*

Proof. For the left to right direction, let $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A}) = \mathsf{StandRep}^k(\mathcal{T}, \mathcal{A})$. Then $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) \neq \emptyset$ by the definition of $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$. More specifically, by Proposition 5.7 we have that (1) $\mathsf{StandRep}^k(\mathcal{T}, \mathcal{A}) \neq \emptyset$ and (2) $\mathsf{StandRep}^m(\mathcal{T}, \mathcal{A}) = \emptyset$ for all $m < k$. By (1) then there exists an ABox $\mathcal{A}'$ such that $\mathcal{A}' \in \mathsf{StandRep}^k(\mathcal{T}, \mathcal{A})$ and subsequently the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^k \rangle \mathcal{A}'$ is satisfiable. Satisfiability of the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^k \rangle \top$ follows immediately. Furthermore, let $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \top$ be satisfiable and let $\mathcal{I}$ be one of its models. It is easy to see then that there is an ABox $\mathcal{A}'$ which is relevant to $\mathcal{A}$ for which the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \mathcal{A}'$ is satisfiable and has $\mathcal{I}$ as a model. This means $\mathsf{StandRep}^m(\mathcal{T}, \mathcal{A}) \neq \emptyset$ for some $m \leq k-1$ which contradicts (2). Therefore $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \top$ is unsatisfiable.

For the right to left direction, since the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \top$ is unsatisfiable the same holds for all formulas $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^m \rangle \top$ as well, where $m < k-1$. By Proposition 5.6 it follows that $\mathsf{StandRep}^m(\mathcal{T}, \mathcal{A}) = \emptyset$ for all $m < k$ (1). Furthermore, since $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^k \rangle \top$ is satisfiable we can find an ABox $\mathcal{A}'$ which is relevant to $\mathcal{A}$ that makes the formula

$\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k} \rangle \mathcal{A}'$ satisfiable. For the same $\mathcal{A}'$ we have that $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \mathcal{A}'$ is unsatisfiable since $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{k-1} \rangle \top$ is unsatisfiable. By Proposition 5.6 again $\mathsf{StandRep}^{k}(\mathcal{T}, \mathcal{A}) \neq \emptyset$ (2). By (1), (2), Proposition 5.7 and the definition of $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$ it follows that $\mathsf{MinStandRep}(\mathcal{T}, \mathcal{A}) = \mathsf{StandRep}^{k}(\mathcal{T}, \mathcal{A})$. □

So given a standard repair $\mathcal{A}'$ of $\mathcal{A}$ with respect to $\mathcal{T}$, in order to check if $\mathcal{A}'$ is a minimal standard repair we need two steps: first, pinpoint the (unique) number k such that $\mathsf{StandRep}^{k}(\mathcal{T}, \mathcal{A}) = \mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$ through Proposition 5.8 and second, check whether $\mathcal{A}' \in \mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$ using Proposition 5.6; more specifically, by the second condition of Proposition 5.8 the third condition of Proposition 5.6 is implied and can be omitted. Moreover, in order to pinpoint the (unique) number k from Proposition 5.8 we can start from $n = 1$ and increment the value of n until the formula $\mathcal{A} \wedge \langle \pi_{\mathcal{T}}^{n} \rangle \top$ becomes satisfiable. If we don't want to exceed a specific bound though and iterate the previous procedure indefinitely, we can start with a satisfiability check of the formula $\mathcal{A} \wedge \Big\langle \bigcup_{m \leq n} \pi_{\mathcal{T}}^{m} \Big\rangle \top$ which will verify the existence of the number k below the specified bound n we are willing to check.

In the next section we are going to provide a framework and tools for more *informed* ways of repairing ABoxes. As with earlier chapters, this will be achieved by endowing TBox axioms with *preferred* update actions through which they should be repaired in case of inconsistency. As we discussed in Section 5.1, the distinguishing feature of the present approach is the *global* nature of the update actions and their greater expressiveness.

5.5 Active Inclusion Axioms in $\mathcal{ALCI}$ TBoxes

In this section we use the structure of the programs in our quest to enrich the axioms of the TBoxes with update actions. We then examine the various results we can obtain when we start from an ABox that is inconsistent with a TBox and modify it via these preferred update actions such that it is ultimately consistent with the concept inclusion axioms of the TBox. The first thing to do is to define the static and active concept inclusions.

Definition 5.7. *Any concept inclusion axiom ρ is called a* static *concept inclusion. An* active *concept inclusion is of the form 'λ if $\neg\rho$' where $\lambda \in \mathsf{pr}(\rho)$. An* active *TBox, denoted*

by $\mathsf{a}\mathcal{T}$*, is a set of active concept inclusions.*

For $\eta =$'λ if $\neg\rho$' we let $\mathsf{static}(\eta) = \rho$. For an active TBox $\mathsf{a}\mathcal{T}$ we let $\mathsf{static}(\mathsf{a}\mathcal{T}) = \{\mathsf{static}(\eta) \ : \ \eta \in \mathsf{a}\mathcal{T}\}$. We say that $\mathsf{a}\mathcal{T}$ extends $\mathcal{T}$ iff $\mathcal{T} = \mathsf{static}(\mathsf{a}\mathcal{T})$. The next definition extends the notion of inconsistency.

Definition 5.8. *Let* $\mathsf{a}\mathcal{T}$ *be an active TBox. We say that an ABox* $\mathcal{A}$ *is consistent with* $\mathsf{a}\mathcal{T}$ *iff* $\mathcal{A}$ *is consistent with* $\mathsf{static}(\mathsf{a}\mathcal{T})$*. Similarly,* $\mathcal{A}$ *is inconsistent with* $\mathsf{a}\mathcal{T}$ *iff* $\mathcal{A}$ *is inconsistent with* $\mathsf{static}(\mathsf{a}\mathcal{T})$*.*

We can see now that the active TBoxes of Section 5.1 can be expressed in this framework and, more specifically, that $\mathsf{static}(\mathsf{a}\mathcal{T}_1) = \mathsf{static}(\mathsf{a}\mathcal{T}_2) = \mathcal{T}$.

Active TBoxes are a way for an ontology engineer to specify preferred update actions to tackle inconsistencies. Our goal now is to provide appropriate tools that check whether an assertion (or a whole ABox) is indeed the outcome of applying these preferred update actions a bounded number of times. More specifically, given an active TBox $\mathsf{a}\mathcal{T}$ and an ABox $\mathcal{A}$ that is inconsistent with $\mathsf{a}\mathcal{T}$, we would like to define a program $\pi^n_{\mathsf{a}\mathcal{T}}$ that uses the active inclusion axioms of $\mathsf{a}\mathcal{T}$ to determine whether or not the formula $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle \mathcal{A}'$ is satisfiable for an assertion or ABox $\mathcal{A}'$. We thus introduce in the following section the notion of *dynamic repair programs*, which choose among the update actions in $\mathsf{a}\mathcal{T}$ in order to provide repair solutions given an upper bound on the number of steps. Their construction resembles the dynamic repairs we have seen in Chapters 2 and 4.

5.6 Dynamic Repairs

We now investigate dynamic repair programs which make use of the setting of Dynamic Logic in order to provide a *dynamic* perspective on the repair procedure; more specifically, by making use of the **restricted while** programs of dyn$\mathcal{ALCIO}$ we are able to iterate the process of applying the preferred update actions and apply each one at a time until all the inconsistencies are eliminated after a bounded number of computations. We begin with the basic definition.

Definition 5.9. *Let $\mathcal{T}$ be a TBox, $\mathsf{a}\mathcal{T}$ an active TBox extending $\mathcal{T}$ and $n \in \mathbb{N}$. The* dynamic repair program up to n *of $\mathsf{a}\mathcal{T}$ is the following program:*

$$\pi^n_{\mathsf{a}\mathcal{T}} = \mathsf{while}\left(\neg\mathcal{T}\right)\mathsf{do}^{\leq n}\left(\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)$$

Just like the standard repair programs, the dynamic repair programs apply update actions from $\mathsf{pr}(\rho)$ for each $\rho \in \mathcal{T}$ that is violated until $\mathcal{T}$ is ultimately satisfied after a bounded number of steps. The difference is in the actual choice of these $\lambda \in \mathsf{pr}(\rho)$: while previously there was no preference between the update actions chosen, the update actions now are exactly those indicated by the active axioms of $\mathsf{a}\mathcal{T}$. In practice, despite their slightly different nature, dynamic repair programs are a special case of standard repair programs as is shown in the following proposition.

Proposition 5.9. *Let $\mathcal{T}$ be a TBox, $\mathsf{a}\mathcal{T}$ an active TBox extending $\mathcal{T}$ and $n \in \mathbb{N}$. Then* $\| \pi^n_{\mathsf{a}\mathcal{T}} \| \subseteq \| \pi^n_{\mathcal{T}} \|$.

Proof. We have $\| \pi^n_{\mathsf{a}\mathcal{T}} \| = \left\| \mathsf{while}\left(\neg\mathcal{T}\right)\mathsf{do}^{\leq n}\left(\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)\right\| = \left\|\left(\neg\mathcal{T}\,?\,;\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)^{\leq n};\mathcal{T}\,?\right\|$. The result follows from the fact that, unlike PDL where the program '$(\varphi?;\gamma)^{\leq n};\neg\varphi?$' is not equivalent to the program '$\gamma^{\leq n};\neg\varphi?$' since the latter can apply γ at most n times even if $\neg\varphi$ is satisfied from the start, for active TBoxes $\mathsf{a}\mathcal{T}$ with $\mathsf{static}(\mathsf{a}\mathcal{T}) = \mathcal{T}$ we have:

$$\left\|\left(\neg\mathcal{T}\,?\,;\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)^{\leq n};\mathcal{T}\,?\right\| = \left\|\left(\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)^{\leq n};\mathcal{T}\,?\right\| \quad (1)$$

This happens because applying any atomic program $(\lambda \text{ if } \neg\rho) \in \mathsf{a}\mathcal{T}$ from the nondeterministic composition $\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta$ in an interpretation $\mathcal{I}$ translates into applying the local program λ to the set $(\neg\rho)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \rho^{\mathcal{I}}$. If $\mathcal{T}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ then $\rho^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every $\rho \in \mathcal{T}$ and thus $(\mathcal{I},\mathcal{I}') \in \left\|\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right\|$ iff $\mathcal{I} = \mathcal{I}'$. It is easy to check then that:

$$(\mathcal{I},\mathcal{I}') \in \left\|\left(\neg\mathcal{T}\,?\,;\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)^{\leq n};\mathcal{T}\,?\right\| \quad \text{iff} \quad \mathcal{I} = \mathcal{I}' \quad \text{iff} \quad (\mathcal{I},\mathcal{I}') \in \left\|\left(\bigcup_{\eta\in\mathsf{a}\mathcal{T}}\eta\right)^{\leq n};\mathcal{T}\,?\right\|$$

If $\mathcal{T}^{\mathcal{I}} \neq \Delta^{\mathcal{I}}$ then $\rho^{\mathcal{I}} \neq \Delta^{\mathcal{I}}$ for some $\rho \in \mathcal{T}$. More specifically, it will be the case that $\mathcal{T}^{\mathcal{I}} = \emptyset$ and $\rho^{\mathcal{I}} = \emptyset$ due to the use of the universal operator in their representation. This means that $(\neg\mathcal{T})^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \mathcal{T}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and consequently $(\mathcal{I},\mathcal{I}) \in \| \neg\mathcal{T}\,? \|$. It follows that:

$$(\mathcal{I},\mathcal{I}') \in \Big\| \neg\mathcal{T}\,?\,;\bigcup_{\eta\in \mathsf{a}\mathcal{T}}\eta \Big\| \quad \text{iff} \quad (\mathcal{I},\mathcal{I}') \in \Big\| \bigcup_{\eta\in \mathsf{a}\mathcal{T}}\eta \Big\|$$

which also extends to the programs $\Big(\neg\mathcal{T}\,?\,;\bigcup_{\eta\in \mathsf{a}\mathcal{T}}\eta\Big)^{\leq n}$ and $\Big(\bigcup_{\eta\in \mathsf{a}\mathcal{T}}\eta\Big)^{\leq n}$. So (1) holds. Furthermore, if $(\mathcal{I},\mathcal{I}') \in \Big\| \bigcup_{\eta\in \mathsf{a}\mathcal{T}}\eta \Big\|$ then $(\mathcal{I},\mathcal{I}') \in \Big\| \bigcup_{\substack{\rho\in\mathcal{T}\\ \lambda\in\mathsf{pr}(\rho)}}\big(\lambda \text{ if } \neg\rho\big) \Big\|$ by Definition 5.7. Thus:

$$\Big\| \Big(\bigcup_{\eta\in \mathsf{a}\mathcal{T}}\eta\Big)^{\leq n};\mathcal{T}\,? \Big\| \subseteq \Big\| \Big(\bigcup_{\substack{\rho\in\mathcal{T}\\ \lambda\in\mathsf{pr}(\rho)}}\big(\lambda \text{ if } \neg\rho\big)\Big)^{\leq n};\mathcal{T}\,? \Big\| \quad (2)$$

By (1) and (2) then $\| \pi^n_{\mathsf{a}\mathcal{T}} \| \subseteq \| \pi^n_{\mathcal{T}} \|$. □

So, in a similar manner as before, we can now utilize the atomic programs of our logic to build decision procedures that check if an assertion or ABox $\mathcal{A}'$ is indeed the outcome of applying the preferred update actions of an active TBox after a bounded number of steps. We can witness this in the following corollary.

Corollary 5.1. *Let $\mathsf{a}\mathcal{T}$ be an active TBox, let $\mathcal{A}$ be an ABox, let $\mathcal{A}'$ be either an assertion or an ABox and let $n \in \mathbb{N}$. If the formula $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle\, \mathcal{A}'$ is satisfiable then $\mathcal{A}'$ is consistent with $\mathsf{a}\mathcal{T}$.*

Proof. Consider $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle\, \mathcal{A}'$ is satisfiable. By Proposition 5.9 the formula $\mathcal{A} \wedge \langle \pi^n_{\mathcal{T}} \rangle\, \mathcal{A}'$ is also satisfiable, where $\mathcal{T} = \mathsf{static}(\mathsf{a}\mathcal{T})$. Proposition 5.5 then gives that $\mathcal{T} \wedge \mathcal{A}'$ is consistent or, equivalently, that $\mathcal{A}'$ is consistent with $\mathcal{T}$. By Definition 5.8 then $\mathcal{A}'$ is also consistent with $\mathsf{a}\mathcal{T}$. □

Once again, satisfiability of the formula $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle\, \mathcal{A}'$ provides much more than just consistency checking: it also makes sure that the assertion or ABox $\mathcal{A}'$ conforms to the active axioms of $\mathsf{a}\mathcal{T}$ as well as ensures we can achieve this with at most n repair steps. The definition of a *dynamic repair* immediately follows.

Definition 5.10. *Let $\mathsf{a}\mathcal{T}$ be an active TBox and let $\mathcal{A}$ be an ABox. An ABox $\mathcal{A}'$ is a* dynamic repair *of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$ iff $\mathcal{A}'$ is relevant to $\mathcal{A}$ and the formula $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle\, \mathcal{A}'$ is satisfiable for some $n \in \mathbb{N}$.*

Going back to the example of Section 5.1 again (see also the example on Section 1.1), we can check now that:

$$\mathcal{A}_1 = \{\mathsf{John} : \mathsf{Male} \sqcap \mathsf{Father} \sqcap \mathsf{Parent}, \mathsf{Mary} : \neg\mathsf{OnlyChild}, \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})\}$$
$$\mathcal{A}_2 = \{\mathsf{John} : \mathsf{Male} \sqcap \neg\mathsf{Father} \sqcap \neg\mathsf{Parent}, \mathsf{Mary} : \mathsf{OnlyChild}\}$$

are indeed dynamic repairs of:

$$\mathcal{A} = \{\mathsf{John} : \mathsf{Male} \sqcap \mathsf{Father} \sqcap \neg\mathsf{Parent}, \mathsf{Mary} : \mathsf{OnlyChild}, \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})\}$$

with respect to $\mathsf{a}\mathcal{T}_1$ and $\mathsf{a}\mathcal{T}_2$, respectively, by checking that both of the formulas $\mathcal{A} \wedge \langle \pi^2_{\mathsf{a}\mathcal{T}_1} \rangle \mathcal{A}_1$ and $\mathcal{A} \wedge \langle \pi^2_{\mathsf{a}\mathcal{T}_2} \rangle \mathcal{A}_2$ are satisfiable. Furthermore, using the reduction axioms of Section 5.3 we have that:

$$\langle \eta_2; \eta_3 \rangle \mathcal{A}_1 \leftrightarrow \mathsf{John} : (\mathsf{Male} \wedge \mathsf{Father}) \wedge \mathsf{Mary} : (\neg\mathsf{OnlyChild} \vee \exists\mathsf{hasSibling}.\top) \wedge \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})$$
$$\langle \eta_4; \eta_5 \rangle \mathcal{A}_2 \leftrightarrow \mathsf{John} : (\mathsf{Male} \wedge \neg\mathsf{Parent}) \wedge \mathsf{Mary} : \mathsf{OnlyChild}$$

and thus $\mathcal{A} \rightarrow \langle \eta_2; \eta_3 \rangle \mathcal{A}_1$ and $\mathcal{A} \rightarrow \langle \eta_4; \eta_5 \rangle \mathcal{A}_2$ are both valid.

Definitions 5.5 and 5.6 and Propositions 5.6, 5.7 and 5.8 then extend in the obvious way, with $\mathsf{DynRep}^n(\mathsf{a}\mathcal{T}, \mathcal{A})$ comprising the dynamic repairs that can be achieved with a minimum of n computations and $\mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$ comprising the dynamic repairs which require the least amount of computations. So similarly with standard repairs, given a dynamic repair $\mathcal{A}'$ of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$, in order to check if $\mathcal{A}'$ is a minimal dynamic repair we first pinpoint the (unique) number k such that $\mathsf{DynRep}^k(\mathsf{a}\mathcal{T}, \mathcal{A}) = \mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$ and then check whether $\mathcal{A}' \in \mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$ by the obvious extensions of the decision procedures from standard to dynamic repairs.

Last but not least, for an active TBox $\mathsf{a}\mathcal{T}$ let us restrict $\mathsf{static}(\mathsf{a}\mathcal{T})$ to the subsets of each $\mathsf{pr}(\rho)$ that comprise the local programs of each active axiom $\eta \in \mathsf{a}\mathcal{T}$. For an ABox $\mathcal{A}$ then it is easy to see that $\mathcal{A}'$ is a dynamic repair of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$ iff $\mathcal{A}'$ is a standard repair of $\mathcal{A}$ with respect to $\mathsf{static}(\mathsf{a}\mathcal{T})$. We end the section with a corollary based on this observation that shows the relationship between standard and dynamic repairs.

Corollary 5.2. *Let $\mathcal{T}$ be a TBox, let $\mathsf{a}\mathcal{T} = \{\lambda \text{ if } \neg\rho \mid \rho \in \mathcal{T} \text{ and } \lambda \in \mathsf{pr}(\rho)\}$ and let $\mathcal{A}$ be an ABox. An ABox $\mathcal{A}'$ is a dynamic repair of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$ iff $\mathcal{A}'$ is a standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$. Furthermore, $\mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A}) = \mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$ and thus $\mathcal{A}'$ is a minimal dynamic repair of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$ iff $\mathcal{A}'$ is a minimal standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$.*

Proof. The left to right direction is immediate since it is easy to check by Definition 5.10 and Proposition 5.9 that every dynamic repair of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$ is also a standard repair of $\mathcal{A}$ with respect to $\mathsf{static}(\mathsf{a}\mathcal{T})$. For the other direction, we recall (from the proof of Proposition 5.9) that $\| \pi^n_{\mathsf{a}\mathcal{T}} \| = \left\| \left(\bigcup_{\eta \in \mathsf{a}\mathcal{T}} \eta \right)^{\leq n}; \mathcal{T}? \right\|$. But since $\mathsf{a}\mathcal{T} = \{\lambda \text{ if } \neg\rho \mid \rho \in \mathcal{T} \text{ and } \lambda \in \mathsf{pr}(\rho)\}$ then $\mathsf{static}(\mathsf{a}\mathcal{T}) = \mathcal{T}$ and $\left\| \bigcup_{\eta \in \mathsf{a}\mathcal{T}} \eta \right\| = \left\| \bigcup_{\substack{\rho \in \mathcal{T} \\ \lambda \in \mathsf{pr}(\rho)}} \left(\lambda \text{ if } \neg\rho\right) \right\|$ which gives:

$$\left\| \pi^n_{\mathsf{a}\mathcal{T}} \right\| = \left\| \left(\bigcup_{\eta \in \mathsf{a}\mathcal{T}} \eta \right)^{\leq n}; \mathcal{T}? \right\| = \left\| \left(\bigcup_{\substack{\rho \in \mathcal{T} \\ \lambda \in \mathsf{pr}(\rho)}} \left(\lambda \text{ if } \neg\rho\right) \right)^{\leq n}; \mathcal{T}? \right\| = \left\| \pi^n_{\mathcal{T}} \right\|$$

Therefore, if $\mathcal{A}'$ is a standard repair of $\mathcal{A}$ with respect to $\mathcal{T}$ then $\mathcal{A}'$ is relevant to $\mathcal{A}$ and the formula $\mathcal{A} \wedge \langle \pi^n_{\mathcal{T}} \rangle \mathcal{A}'$ is satisfiable for some $n \in \mathbb{N}$. This means that $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle \mathcal{A}'$ is satisfiable and, by Definition 5.10, $\mathcal{A}'$ is a dynamic repair of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$. The fact that $\mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A}) = \mathsf{MinStandRep}(\mathcal{T}, \mathcal{A})$ follows similarly. □

5.7 Discussion and Conclusion

In this chapter we have proposed a richer framework for defining extensions of TBoxes with active axioms, specifying preferred repairing routes in case their static part is violated by an ABox. We tackled the problem of checking whether there exists a repair of an ABox that is inconsistent with a TBox in a bounded number of computations through dynamic logic programs. Furthermore, given a possible repair of an ABox, we have shown how to check whether or not it can be obtained from the active axioms of the TBox under a bounded number of steps and, in case it is indeed a repair, whether or not it can do so with the least amount of steps possible.

The reader may have noticed that we make no effort to repair an axiom of the form

'$\top \sqsubseteq \exists r.C$'. This could in principle be done by either allowing the addition of roles, i.e., local programs of the form $+r.\varphi$ and $+r^{\mathsf{c}}.\varphi$, or the addition/removal of a concept to only one individual through a role, i.e., local programs of the form $\exists r.\pm A$ and $\exists r^{\mathsf{c}}.\pm A$. The reason is the following: in the first case (addition of roles), adding to an individual a relation between the individual and any member of the domain that has property C may produce a big change in the resulting interpretation, thus creating possible repairs that are very undesirable. Consider, for instance, the following KB:

$$\mathcal{T} = \{\top \sqsubseteq \exists \mathsf{hasFather.Male}\}$$
$$\mathcal{A} = \{\mathsf{John} : \mathsf{Male} \sqcap \forall \mathsf{hasFather}.\neg \mathsf{Male}\}$$

Applying to John the update action $+\mathsf{hasFather.Male}$ then will result in the ABox $\mathcal{A}' = \{\mathsf{hasFather}(\mathsf{John}, \mathsf{John}), \mathsf{John} : \mathsf{Male}\}$ being a possible repair (and also other repairs with the role assertions $\mathsf{hasFather}(\mathsf{John}, \mathsf{X})$, for every other possible individual X in the ABox who may have the property Male) which is obviously not what a repair should be. Similarly for the addition of inverse roles. On the other hand, both (1) adding to an individual a relation between the individual and only one (abstract) member of the domain and (2) adding or removing a concept only to one individual through a role cannot be pursued because of technical limitations: it is true that enhancing the logic with these extra local programs would produce more repairs in practice, but the reduction axioms would unfortunately fail and the logic would no longer be reducible to its static part.

Even in our framework, there exist cases where one should be careful with concepts like the previous example. More specifically, when a concept appears inside an axiom and, in case of inconsistency of the axiom with an ABox, repairing this concept amounts to '*something should exist*' then an alternative repairing route might be better pursued by the active axioms. For instance, the axiom '$\rho = \mathsf{Grandparent} \sqsubseteq \exists \mathsf{hasChild.Parent}$' should only be extended as follows:

$$-\mathsf{Grandparent\ if\ } \neg\, (\mathsf{Grandparent} \sqsubseteq \exists \mathsf{hasChild.Parent})$$

since the inconsistency that comes from the concept '$\exists \mathsf{hasChild.Parent}$' is not trivial: the assertion $\mathsf{John} : \mathsf{Grandparent} \sqcap \neg \exists \mathsf{hasChild}.\top$ is inconsistent with this axiom but cannot be repaired by adding properties to John's children (since he doesn't have any). Therefore,

in order to assure that a repairing procedure will not fail, an ontology engineer should not 'risk' putting the update action '$\forall$hasChild.$+$Parent' which is the only other meaningful program from the set $\mathsf{pr}(\rho)$ apart from '$-$Grandparent'. Although for another individual (that has children) it could provide a drastic repair (drastic in the sense that *all* children would be parents, although *at least one* is needed), it will always fail for John since the formula:

$$\mathsf{John} : \Big(\mathsf{Grandparent} \wedge \neg\exists \mathsf{hasChild}.\top\Big) \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle \top$$

is unsatisfiable for all $n \in \mathbb{N}$ when the update action '$\forall$hasChild.$+$Parent' is the only choice.

Our approach in this chapter and the logic we use don't allow to check whether a possible (standard or dynamic) repair exists in general, since $\mathsf{StandRep}(\mathcal{T}, \mathcal{A}) \neq \emptyset$ iff the formula $\mathcal{A} \wedge \langle \pi^n_{\mathcal{T}} \rangle \top$ is satisfiable for some $n \in \mathbb{N}$ (and respectively $\mathsf{DynRep}(\mathsf{a}\mathcal{T}, \mathcal{A}) \neq \emptyset$ iff the formula $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle \top$ is satisfiable for some $n \in \mathbb{N}$). We thus have a positive answer in case a (standard or dynamic) repair exists but we are not able to decide in case there isn't. The compromise we make though gives us some extra information, i.e., we can check if a repair exists within a specific number of computations and, in case there is, what is the minimal number of computations through which we can achieve this. This makes sense in real world applications where a big deviation from the original ABox may not be desirable. This means that, in practice, if a transaction has turned an ABox that is consistent with $\mathsf{a}\mathcal{T}$ into one that is inconsistent with $\mathsf{a}\mathcal{T}$ then the dynamic repair should only involve few active axioms. It follows that the maximal allowed number of update actions can reasonably be expected to be rather low.

Now, in order to be in a position to check if a repair exists in the general case, the logic dyn$\mathcal{ALCIO}$ has to be extended with the 'Kleene star' operator of PDL, similarly to the logic dyn$\mathcal{ALCO}$ of Chapter 4. Then we would be able to abbreviate an arbitrary finite number of applications of 'λ if $\neg\rho$' with a single program $\pi_{\mathsf{a}\mathcal{T}}$, thus deciding the existence of a dynamic repair by checking if the formula $\mathcal{A} \wedge \langle \pi_{\mathsf{a}\mathcal{T}} \rangle \top$ is satisfiable or not. However, the addition of this program makes the reduction to the static fragment unachievable and the logic dyn$\mathcal{ALCIO}$ extended with the 'Kleene star' operator is not guaranteed to be

decidable anymore. This is showcased in the following example, where $(.)^*$ is the Kleene star operator.

Example 5.3. *Let $\mathcal{I}$ be an interpretation such that:*

- $\Delta^{\mathcal{I}} = \{\alpha_1, \alpha_2, \dots\}$
- $A^{\mathcal{I}} = \{\alpha_1\}$
- $(\alpha_n, \alpha_{n+1}) \in r^{\mathcal{I}}$ *for all* $n \in \mathbb{N}$

Then we have that $\Big(\langle(\forall r.{+}A \text{ if } A)^*\rangle\, A\Big)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ *and* $\Big(\langle(\forall r.{+}A \text{ if } A)^{\leq n}\rangle\, A\Big)^{\mathcal{I}} \subset \Delta^{\mathcal{I}}$ *for all* $n \in \mathbb{N}$*, i.e., the program* $(\forall r.+A \text{ if } A)^*$ *cannot be reduced to* $(\forall r.+A \text{ if } A)^{\leq n}$ *for any* $n \in \mathbb{N}$.

The reason this didn't happen for dyn$\mathcal{ALCO}$ is the combination of the *local* nature of its update actions with the less expressive power of its atomic programs. But as we argued before, deciding the existence of a standard or dynamic repair in general may not be very interesting, since the repair might be too divergent from the original ABox and difficult to pinpoint. So with the approach we took in this chapter, an ontology engineer can set a benchmark by specifying the maximum number of operations within which a repair should be achieved and, in case this is not possible, it may be an indication that the active axioms defined in the active TBox may be suboptimal in producing a desirable repair.

Summing up, we have taken a more elaborate semantic approach to the investigation of repairing ABoxes with regard to active TBoxes, the latter being an extension of regular TBoxes with update actions inspired by the active integrity constraints of the database literature. This time we have exploited a dynamic logic framework admitting *global* semantics for atomic programs on which various decision procedures are introduced and compared. Having successfully overcome both limitations that we set out to do at the end of Chapter 3, we end the thesis by combining the aforementioned semantic proposal to evaluating repaired ABoxes with the syntactic proposal to constructing them.

CHAPTER 6

Conclusion

Throughout the course of this thesis we investigated methods for maintaining the integrity and consistency of knowledge bases, both in the propositional and in the DL setting, with a specific goal in mind: extend propositional databases and description logic TBoxes with preferred update actions and take them carefully into account when searching for repairing solutions. Whereas the landscape in the propositional case was already explored and understood, the extension to higher level languages like Description Logics proved to be non-trivial and challenging. Relying mainly on Dynamic Logic we acquired tools for both representing known procedures in our setting as well as constructing new, dynamic ones.

Speaking of Dynamic Logic, and as we already mentioned at the end of Section 1.2.3, a correspondence between DLs and (variants of) PDL has been achieved early on [Schild, 1991]. Let us note at this point that we do not attempt something similar here. In practice, we merely extend $\mathcal{ALCO}$ and $\mathcal{ALCIO}$ with simple atomic (and complex) programs that ultimately reduce to static $\mathcal{ALCO}(\mathcal{U})$ and $\mathcal{ALCIO}(\mathcal{U})$ concepts. Thus, our logics are as expressive as plain $\mathcal{ALCO}(\mathcal{U})$ and $\mathcal{ALCIO}(\mathcal{U})$, respectively, and do not admit regular expressions over roles like role *union*, role *composition*, *reflexive-transitive closure* of roles and role *identity*. This makes DLs admitting regular expressions over roles like $\mathcal{ALCO}_{reg}$ and $\mathcal{ALCIO}_{reg}$ strictly more expressive than dyn$\mathcal{ALCO}$ and dyn$\mathcal{ALCIO}$, since the universal role can be expressed in the former by the union and the reflexive-transitive closure of roles.

Our approaches also come close to the work presented in [Ahmetaj et al., 2017]

where dynamic problems on graph-structured data are reduced to static DL reasoning. The Dynamic Logic framework presented here though is different and allows us to not be restricted into finite interpretations, among other things. Furthermore, we introduce and work mainly with active TBoxes, which differentiates us from other approaches to tackle (possibly dynamic) problems in DLs using regular TBoxes.

Last but not least, let us note that our perspective in this thesis differs from [Motik et al., 2009, Tao et al., 2010], since we do not split the TBox up into integrity constraints and other, non-integrity axioms. The direction we pursue is clearly different from their attempts to encode integrity constraints into TBoxes, which differentiate the KB from the set of integrity constraints, treating the latter with a closed-worlds assumption and using different dedicated semantics. Nevertheless, it looks like a promising future direction to import their way of handling integrity constraints inside TBoxes and see how such an approach would compare with our own in this thesis.

We now conclude our investigations with an attempt at connecting the two different directions we took for repairing ABoxes: constructing them syntactically and assessing them semantically. We recall from Chapter 3 that, starting from an ABox $\mathcal{A}$, the set $S_{\mathcal{A}}$ consists of all concept symbols occurring in $\mathcal{A}$. Furthermore, the set $\mathcal{R}^{\mathcal{A}}$ comprises the ABoxes that can be reached from $\mathcal{A}$ by an arbitrary number of syntactic modifications and it is always finite. These syntactic modifications however take place only for atomic concepts and not for nominals or roles. Since in Chapter 5 we are working on $\mathcal{ALCIO}$ and nominals can exist in ABoxes, we now extend the set $S_{\mathcal{A}}$ into the set $S^*_{\mathcal{A}}$ which also includes the nominals of the ABox $\mathcal{A}$ and also define the sets $A^*_{\sqcup}, A^*_{\sqcap}, A^*_{\neg}, \Gamma^*_{\mathcal{A}:A}$ and $\Gamma^*_{\mathcal{A}}$ accordingly. We can then also extend all the definitions of Section 3.3 to incorporate nominals and Propositions 3.1 and 3.2 are still valid. The same goes for removing roles: since one of the distinguishing features of Chapter 5 is the ability to remove roles through the programs $-r.\varphi$ and $-r^{\mathsf{c}}.\varphi$, we can extend Definition 3.4 so that $\mathcal{A} \sim_1 \mathcal{A}'$ if also $\mathcal{A}'$ is the outcome of removing a role assertion from $\mathcal{A}$. This implies that $\mathcal{A}$ and $\mathcal{A}'$ are semantically different from one another and once again the remaining definitions and propositions stay intact. We also write $\mathcal{A}' = \mathcal{A}\,[-r(a,b)]$ when this happens and $U^{\mathcal{A}'}_{\mathcal{A}} = \{r(a,b) \mapsto \bot\}$ if $\mathcal{A}' = \mathcal{A}\,[-r(a,b)]$.

Taking these new syntactic modifications into account, we can see how all ABoxes

in the new set $\mathcal{R}^{\mathcal{A}}$ follow Definition 5.3 and are relevant to $\mathcal{A}$ since no new individuals, concepts or roles are introduced and role assertions can be only removed. This means that searching for repairs in the set $\mathcal{R}^{\mathcal{A}}$ is safe and, given an active TBox $\mathsf{a}\mathcal{T}$, satisfiability of the formula $\mathcal{A} \wedge \langle \pi^n_{\mathsf{a}\mathcal{T}} \rangle \mathcal{A}'$ for some $n \in \mathbb{N}$ and $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$ is enough to identify $\mathcal{A}'$ as a dynamic repair of $\mathcal{A}$ with respect to $\mathsf{a}\mathcal{T}$. So now the number $n \in \mathbb{N}$ can have a dual purpose: both an upper bound that specifies the maximum number of (semantic) updates within which a repair should be found as well as an upper bound on the number of syntactic modifications that the initial ABox should be the subject of. In other words, for an upper bound $k \in \mathbb{N}$, we only check if the formula $\mathcal{A} \wedge \langle \pi^k_{\mathsf{a}\mathcal{T}} \rangle \mathcal{A}'$ is satisfiable where:

$$\mathcal{A}' \in \bigcup_{n=1}^{k} \mathcal{R}^{\mathcal{A}}_n \quad (1)$$

If no dynamic repair is found in this way then we stop. Of course, since all of the sets $\mathcal{R}^{\mathcal{A}}_n$ as well as the set $\mathcal{R}^{\mathcal{A}}$ are finite, we could search if $\mathcal{A} \wedge \langle \pi^k_{\mathsf{a}\mathcal{T}} \rangle \mathcal{A}'$ is satisfiable for all $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$. The purpose of limiting the $\mathcal{A}'$s such that (1) holds is to once again showcase methods for not straying too far away from the initial ABox.

Let us showcase an example of what has been said so far. We return to the initial example of Section 1.1 and the way we represented it in Chapter 5 through the following active TBoxes:

$$\begin{aligned}
\mathsf{a}\mathcal{T}_1 = \{&+\mathsf{Male} \text{ if } \neg\left(\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}\right), \\
&+\mathsf{Parent} \text{ if } \neg\left(\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}\right), \\
&-\mathsf{OnlyChild} \text{ if } \neg\left(\mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot\right)\} \\
\mathsf{a}\mathcal{T}_2 = \{&-\mathsf{Father} \text{ if } \neg\left(\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \mathsf{Parent}\right), \\
&-\mathsf{hasSibling}.\top \text{ if } \neg\left(\mathsf{OnlyChild} \sqsubseteq \forall \mathsf{hasSibling}.\bot\right)\}
\end{aligned}$$

Remember also that the initial ABox was the following:

$$\mathcal{A} = \{\mathsf{John} : \mathsf{Male} \sqcap \mathsf{Father} \sqcap \neg\mathsf{Parent}, \mathsf{Mary} : \mathsf{OnlyChild}, \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})\}$$

We can see how, taking $S^{\mathcal{A}_1}_{\mathcal{A}} = \Big(\{\neg\mathsf{Parent} \mapsto \mathsf{Parent}\}, \{\mathsf{OnlyChild} \mapsto \neg\mathsf{OnlyChild}\}\Big)$ and $S^{\mathcal{A}_2}_{\mathcal{A}} = \Big(\{\mathsf{Father} \mapsto \neg\mathsf{Father}\}, \{\mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John}) \mapsto \bot\}\Big)$, we have $\mathcal{A} \diamond S^{\mathcal{A}_1}_{\mathcal{A}} = \mathcal{A}_1$ and $\mathcal{A} \diamond S^{\mathcal{A}_2}_{\mathcal{A}} = \mathcal{A}_2$ where:

$$\mathcal{A}_1 = \{\mathsf{John} : \mathsf{Male} \sqcap \mathsf{Father} \sqcap \mathsf{Parent}, \mathsf{Mary} : \neg\mathsf{OnlyChild}, \mathsf{hasSibling}(\mathsf{Mary}, \mathsf{John})\}$$

$$\mathcal{A}_2 = \{\mathsf{John} : \mathsf{Male} \sqcap \neg\mathsf{Father} \sqcap \neg\mathsf{Parent}, \mathsf{Mary} : \mathsf{OnlyChild}\}$$

Thus $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{R}_2^{\mathcal{A}}$ and they are minimal repairs with respect to the number of syntactic modifications needed. Furthermore, we saw in Chapter 5 that both of the formulas $\mathcal{A} \wedge \langle \pi^2_{\mathsf{a}\mathcal{T}_1} \rangle \mathcal{A}_1$ and $\mathcal{A} \wedge \langle \pi^2_{\mathsf{a}\mathcal{T}_2} \rangle \mathcal{A}_2$ are satisfiable and $\mathcal{A}_1, \mathcal{A}_2 \in \mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$, so they are minimal dynamic repairs as well. Let us note though that dynamic repairs that are minimal with regards to the number of syntactic modifications are not necessarily minimal dynamic repairs and vice versa. Consider, for instance, the following active TBox and ABox:

$$\begin{aligned}
\mathsf{a}\mathcal{T} = \{&+\mathsf{Man} \text{ if } \neg\left(\mathsf{Person} \sqcap \mathsf{Male} \sqsubseteq \mathsf{Man}\right),\\
&\forall \mathsf{hasFather}.{+}\mathsf{Male} \text{ if } \neg\left(\mathsf{Person} \sqsubseteq \forall \mathsf{hasFather}.\mathsf{Male}\right),\\
&{-}\mathsf{hasFather}.\neg\mathsf{Male} \text{ if } \neg\left(\mathsf{Person} \sqsubseteq \forall \mathsf{hasFather}.\mathsf{Male}\right)\}\\
\mathcal{A} = \{&\mathsf{Anna} : \mathsf{Person}, \mathsf{Mary} : \mathsf{Person}, \mathsf{Peter} : \mathsf{Person}, \mathsf{John} : \mathsf{Person} \sqcap \neg\mathsf{Male} \sqcap \neg\mathsf{Man},\\
&\mathsf{hasFather}(\mathsf{Anna}, \mathsf{John}), \mathsf{hasFather}(\mathsf{Mary}, \mathsf{John}), \mathsf{hasFather}(\mathsf{Peter}, \mathsf{John})\}
\end{aligned}$$

For the ABox $\mathcal{A}_1 = \{\mathsf{Anna} : \mathsf{Person}, \mathsf{Mary} : \mathsf{Person}, \mathsf{Peter} : \mathsf{Person}, \mathsf{John} : \mathsf{Person} \sqcap \neg\mathsf{Male} \sqcap \neg\mathsf{Man}\}$ we have that $\mathcal{A} \wedge \langle \pi^1_{\mathsf{a}\mathcal{T}} \rangle \mathcal{A}_1$ is satisfiable and thus $\mathcal{A}_1 \in \mathsf{DynRep}^1(\mathsf{a}\mathcal{T}, \mathcal{A}) = \mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$. Furthermore, $\mathcal{A}_1 \in \mathcal{R}_3^{\mathcal{A}}$ since:

$$\begin{aligned}
\mathcal{A}_1 = \mathcal{A} \diamond \Big(&\{\mathsf{hasFather}(\mathsf{Anna}, \mathsf{John}) \mapsto \bot\},\\
&\{\mathsf{hasFather}(\mathsf{Mary}, \mathsf{John}) \mapsto \bot\},\\
&\{\mathsf{hasFather}(\mathsf{Peter}, \mathsf{John}) \mapsto \bot\}\Big)
\end{aligned}$$

and it is not minimal with regards to the number of syntactic modifications because there are dynamic repairs of $\mathcal{A}$ that are achieved in less syntactic steps, e.g.:

$$\begin{aligned}
\mathcal{A}_2 = \{&\mathsf{Anna} : \mathsf{Person}, \mathsf{Mary} : \mathsf{Person}, \mathsf{Peter} : \mathsf{Person}, \mathsf{John} : \mathsf{Person} \sqcap \mathsf{Male} \sqcap \mathsf{Man},\\
&\mathsf{hasFather}(\mathsf{Anna}, \mathsf{John}), \mathsf{hasFather}(\mathsf{Mary}, \mathsf{John}), \mathsf{hasFather}(\mathsf{Peter}, \mathsf{John})\}
\end{aligned}$$

and $\mathcal{A}_2 \in \mathcal{R}_2^{\mathcal{A}}$ by means of $S_{\mathcal{A}}^{\mathcal{A}_2} = \Big(\{\neg\mathsf{Man} \mapsto \mathsf{Man}\}, \{\neg\mathsf{Male} \mapsto \mathsf{Male}\}\Big)$. Note also that $\mathcal{A}_2 \in \mathsf{DynRep}^2(\mathsf{a}\mathcal{T}, \mathcal{A})$ and thus $\mathcal{A}_2$ is not a minimal dynamic repair although it is

a dynamic repair achieved with the least number of syntactic modifications. Finally, it is easy to check that the update sequence $S_{\mathcal{A}}^{\mathcal{A}_3} = \Big(\{\neg\mathsf{Man} \mapsto \mathsf{Man}\}\Big)$ is a PMA repair of $\mathcal{A}$ according to Chapter 3, but $\mathcal{A}_3 \notin \mathsf{DynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$ according to Chapter 5.

So one can choose to look for dynamic repairs in $\mathcal{R}^{\mathcal{A}}$ by interpreting minimality either way s/he wants, pursuing one of the following:

- After fixing the number k, go though all ABoxes $\mathcal{A}'$ in $\mathcal{R}_1^{\mathcal{A}}, \mathcal{R}_2^{\mathcal{A}}, \ldots$ and check whether $\mathcal{A} \wedge \langle \pi_{\mathsf{a}\mathcal{T}}^k \rangle \mathcal{A}'$ is satisfiable. The first ABox for which this is the case will be a dynamic repair achieved with the least number of syntactic modifications. We reiterate that we can either exhaust $\mathcal{R}^{\mathcal{A}}$ or stop at $\mathcal{R}_k^{\mathcal{A}}$.

- After pinpointing the number k for which $\mathsf{DynRep}^k(\mathsf{a}\mathcal{T}, \mathcal{A}) = \mathsf{MinDynRep}(\mathsf{a}\mathcal{T}, \mathcal{A})$ with the procedure of Chapter 5, check for any $\mathcal{A}' \in \mathcal{R}^{\mathcal{A}}$ whether $\mathcal{A} \wedge \langle \pi_{\mathsf{a}\mathcal{T}}^k \rangle \mathcal{A}'$ is satisfiable. The first ABox for which this is the case will be a minimal dynamic repair.

It is clear though that immediate applications of the above procedures in real world scenarios may be computationally impractical. Thus, next research steps should consist of finding 'reasonable' complexity bounds for working with active TBoxes. In connection to Section 1.4, an attempt to combine our results with other areas of research could be to look at possible applications of the proposed repairing methods on nonmonotonic scenarios such as *Defeasible Description Logics* [Britz and Varzinczak, 2016, Britz and Varzinczak, 2017]. In particular, the preferential approach for nonmonotonic reasoning that has been established in the propositional setting has already been extended to DLs [Casini and Straccia, 2010, Giordano et al., 2015] and applying the results reported in this thesis to such a defeasible setting seems like an interesting future endeavor.

Summing up, by venturing into the landscape of active TBoxes we exhibited that preferences when repairing Description Logic KBs can constitute very interesting and important research directions, much like $\mathcal{AIC}$s on databases that came before them. Although a first—and quite theoretical—step in this direction, we believe that the basic idea behind active TBoxes is proving to be fruitful.

Bibliography

[Abiteboul, 1988] Abiteboul, S. (1988). Updates, A new frontier. In Gyssens, M., Paredaens, J., and Gucht, D. V., editors, *ICDT'88, 2nd International Conference on Database Theory, Bruges, Belgium, August 31 - September 2, 1988, Proceedings*, volume 326 of *Lecture Notes in Computer Science*, pages 1–18. Springer.

[Ahmetaj et al., 2017] Ahmetaj, S., Calvanese, D., Ortiz, M., and Simkus, M. (2017). Managing change in graph-structured data using description logics. *ACM Trans. Comput. Log.*, 18(4):27:1–27:35.

[Alchourrón et al., 1985] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Log.*, 50(2):510–530.

[Arlo-Costa, 2014] Arlo-Costa, H. (2014). The logic of conditionals. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Summer 2014 edition.

[Askounis et al., 2016] Askounis, D., Koutras, C. D., and Zikos, Y. (2016). Knowledge means 'all', belief means 'most'. *Journal of Applied Non-Classical Logics*, 26(3):173–192.

[Aucher, 2014] Aucher, G. (2014). Principles of knowledge, belief and conditional belief. In Rebuschi, M., Batt, M., G.Heinzmann, Lihoreau, F., Musiol, M., and Trognon, A., editors, *Dialogue, Rationality, and Formalism*, volume 3 of *Logic, Argumentation & Reasoning*, pages 97–134. Springer.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[Baader et al., 2005] Baader, F., Horrocks, I., and Sattler, U. (2005). Description logics as ontology languages for the semantic web. In Hutter, D. and Stephan, W., editors, *Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *Lecture Notes in Computer Science*, pages 228–248. Springer.

[Baader et al., 2006] Baader, F., Lutz, C., and Suntisrivaraporn, B. (2006). CEL - A polynomial-time reasoner for life science ontologies. In Furbach, U. and Shankar, N., editors, *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of *Lecture Notes in Computer Science*, pages 287–291. Springer.

[Baader and Suntisrivaraporn, 2008] Baader, F. and Suntisrivaraporn, B. (2008). Debugging SNOMED CT using axiom pinpointing in the description logic EL+. In Cornet, R. and Spackman, K. A., editors, *Proceedings of the Third International Conference on Knowledge Representation in Medicine, Phoenix, Arizona, USA, May 31st - June 2nd, 2008*, volume 410 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Balbiani et al., 2014] Balbiani, P., Herzig, A., Schwarzentruber, F., and Troquard, N. (2014). DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE. *CoRR*, abs/1411.7825.

[Balbiani et al., 2013] Balbiani, P., Herzig, A., and Troquard, N. (2013). Dynamic logic of propositional assignments: A well-behaved variant of PDL. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 143–152. IEEE Computer Society.

[Bell, 1990] Bell, J. (1990). The logic of nonmonotonicity. *Artif. Intell.*, 41(3):365–374.

[Bertossi, 2011] Bertossi, L. E. (2011). *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

[Bienvenu et al., 2014] Bienvenu, M., Bourgaux, C., and Goasdoué, F. (2014). Querying inconsistent description logic knowledge bases under preferred repair semantics. In Brodley, C. E. and Stone, P., editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 996–1002. AAAI Press.

[Bienvenu et al., 2016] Bienvenu, M., Bourgaux, C., and Goasdoué, F. (2016). Query-driven repairing of inconsistent dl-lite knowledge bases. In Kambhampati, S., editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 957–964. IJCAI/AAAI Press.

[Boutilier, 1992] Boutilier, C. (1992). *Conditional logics for default reasoning and belief revision*. PhD thesis, University of Toronto.

[Boutilier, 1994] Boutilier, C. (1994). Conditional logics of normality: A modal approach. *Artif. Intell.*, 68(1):87–154.

[Britz et al., 2011] Britz, K., Meyer, T., and Varzinczak, I. J. (2011). Preferential reasoning for modal logics. *Electr. Notes Theor. Comput. Sci.*, 278:55–69.

[Britz and Varzinczak, 2016] Britz, K. and Varzinczak, I. J. (2016). Introducing role defeasibility in description logics. In Michael, L. and Kakas, A. C., editors, *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9-11, 2016, Proceedings*, volume 10021 of *Lecture Notes in Computer Science*, pages 174–189.

[Britz and Varzinczak, 2017] Britz, K. and Varzinczak, I. J. (2017). Towards defeasible SROIQ. In Artale, A., Glimm, B., and Kontchakov, R., editors, *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017.*, volume 1879 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Caroprese et al., 2006] Caroprese, L., Greco, S., Sirangelo, C., and Zumpano, E. (2006). Declarative semantics of production rules for integrity maintenance. In Etalle, S. and Truszczynski, M., editors, *Logic Programming, 22nd International Conference, ICLP 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4079 of *Lecture Notes in Computer Science*, pages 26–40. Springer.

[Caroprese et al., 2009] Caroprese, L., Greco, S., and Zumpano, E. (2009). Active integrity constraints for database consistency maintenance. *IEEE Trans. Knowl. Data Eng.*, 21(7):1042–1058.

[Caroprese and Truszczynski, 2011] Caroprese, L. and Truszczynski, M. (2011). Active integrity constraints and revision programming. *TPLP*, 11(6):905–952.

[Casini and Straccia, 2010] Casini, G. and Straccia, U. (2010). Rational closure for defeasible description logics. In Janhunen, T. and Niemelä, I., editors, *Logics in Artificial Intelligence - 12th European Conference, JELIA 2010, Helsinki, Finland, September 13-15, 2010. Proceedings*, volume 6341 of *Lecture Notes in Computer Science*, pages 77–90. Springer.

[Ceri et al., 1994] Ceri, S., Fraternali, P., Paraboschi, S., and Tanca, L. (1994). Automatic generation of production rules for integrity maintenance. *ACM Trans. Database Syst.*, 19(3):367–422.

[Charrier and Schwarzentruber, 2017] Charrier, T. and Schwarzentruber, F. (2017). A succinct language for dynamic epistemic logic. In Larson, K., Winikoff, M., Das, S., and Durfee, E. H., editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 123–131. ACM.

[Chellas, 1975] Chellas, B. F. (1975). Basic conditional logic. *J. Philosophical Logic*, 4(2):133–153.

[Chomicki and Marcinkowski, 2005] Chomicki, J. and Marcinkowski, J. (2005). Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121.

[Cooper et al., 2016] Cooper, M. C., Herzig, A., Maffre, F., Maris, F., and Régnier, P. (2016). A simple account of multi-agent epistemic planning. In Kaminka, G. A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., and van Harmelen, F., editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 193–201. IOS Press.

[Crocco et al., 1996] Crocco, G., del Cerro, L. F., and Herzig, A., editors (1996). *Conditionals: From Philosophy to Computer Science*. Studies in Logic and Computation. Oxford University Press.

[Cruz-Filipe, 2014] Cruz-Filipe, L. (2014). Optimizing computation of repairs from active integrity constraints. In Beierle, C. and Meghini, C., editors, *Foundations of Information and Knowledge Systems - 8th International Symposium, FoIKS 2014, Bordeaux, France, March 3-7, 2014. Proceedings*, volume 8367 of *Lecture Notes in Computer Science*, pages 361–380. Springer.

[Cruz-Filipe et al., 2013] Cruz-Filipe, L., Gaspar, G., Engrácia, P., and Nunes, I. (2013). Computing repairs from active integrity constraints. In *Seventh International Symposium on Theoretical Aspects of Software Engineering, TASE 2013, 1-3 July 2013, Birmingham, UK*, pages 183–190. IEEE Computer Society.

[Delgrande, 1987] Delgrande, J. P. (1987). A first-order conditional logic for prototypical properties. *Artif. Intell.*, 33(1):105–130.

[Delgrande, 1988] Delgrande, J. P. (1988). An approach to default reasoning based on a first-order conditional logic: Revised report. *Artif. Intell.*, 36(1):63–90.

[Delgrande, 1998] Delgrande, J. P. (1998). *Conditional Logics for Defeasible Reasoning*, volume 2 of *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 135 – 173. Kluwer Academic Publishers.

[Delgrande, 2006] Delgrande, J. P. (2006). On a rule-based interpretation of default conditionals. *Ann. Math. Artif. Intell.*, 48(3-4):135–167.

[Delgrande, 2012] Delgrande, J. P. (2012). What's in a default? thoughts on the nature and role of defaults in nonmonotonic reasoning. In Brewka, G., Marek, V. W., and Truszczyński, M., editors, *NonMonotonic Reasoning, Essays celebrating its 30th anniversary*, volume 31 of *Studies in Logic*, pages 89–110. College Publications.

[Ditmarsch et al., 2007] Ditmarsch, H. v., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*. Springer Publishing Company, Incorporated, 1st edition.

[Doutre et al., 2014] Doutre, S., Herzig, A., and Perrussel, L. (2014). A dynamic logic framework for abstract argumentation. In Baral, C., Giacomo, G. D., and Eiter, T.,

editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

[Eiter and Lukasiewicz, 2000] Eiter, T. and Lukasiewicz, T. (2000). Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artif. Intell.*, 124(2):169–241.

[Feuillade and Herzig, 2014] Feuillade, G. and Herzig, A. (2014). A dynamic view of active integrity constraints. In Fermé, E. and Leite, J., editors, *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, pages 486–499. Springer.

[Feuillade et al., 2018] Feuillade, G., Herzig, A., and Rantsoudis, C. (2018). A dynamic extension of ALCO for repairing via preferred updates. In *Proceedings of the 31th International Workshop on Description Logics, Tempe, Arizona, October 27-29, 2018*. To appear.

[Feuillade et al., 2019] Feuillade, G., Herzig, A., and Rantsoudis, C. (2019). A dynamic logic account of active integrity constraints. *Fundamenta Informaticae*. To appear.

[Fischer and Ladner, 1979] Fischer, M. J. and Ladner, R. E. (1979). Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211.

[Flesca et al., 2004] Flesca, S., Greco, S., and Zumpano, E. (2004). Active integrity constraints. In Moggi, E. and Warren, D. S., editors, *Proceedings of the 6th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, 24-26 August 2004, Verona, Italy*, pages 98–107. ACM.

[Flouris et al., 2009] Flouris, G., d'Aquin, M., Antoniou, G., Pan, J. Z., and Plexousakis, D. (2009). Special issue on ontology dynamics. *J. Log. Comput.*, 19(5):717–719.

[Flouris et al., 2005] Flouris, G., Plexousakis, D., and Antoniou, G. (2005). Updating dls using the AGM theory: A preliminary study. In Horrocks, I., Sattler, U., and Wolter, F., editors, *Proceedings of the 2005 International Workshop on Description*

Logics (DL2005), Edinburgh, Scotland, UK, July 26-28, 2005, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Friedman and Halpern, 1997] Friedman, N. and Halpern, J. Y. (1997). Modeling belief in dynamic systems, part I: foundations. *Artif. Intell.*, 95(2):257–316.

[Geffner and Pearl, 1992] Geffner, H. and Pearl, J. (1992). Conditional entailment: Bridging two approaches to default reasoning. *Artif. Intell.*, 53(2-3):209–244.

[Ginsberg, 1986] Ginsberg, M. L. (1986). Counterfactuals. *Artif. Intell.*, 30(1):35–79.

[Giordano et al., 2009] Giordano, L., Gliozzi, V., Olivetti, N., and Pozzato, G. L. (2009). Analytic tableaux calculi for KLM logics of nonmonotonic reasoning. *ACM Trans. Comput. Log.*, 10(3):18:1–18:47.

[Giordano et al., 2015] Giordano, L., Gliozzi, V., Olivetti, N., and Pozzato, G. L. (2015). Semantic characterization of rational closure: From propositional logic to description logics. *Artif. Intell.*, 226:1–33.

[Goldblatt, 1987] Goldblatt, R. (1987). *Logics of Time and Computation*. Center for the Study of Language and Information, Stanford, CA, USA.

[Harel et al., 2000] Harel, D., Kozen, D., and Tiuryn, J. (2000). *Dynamic Logic*. MIT Press.

[Herzig, 2014] Herzig, A. (2014). Belief change operations: A short history of nearly everything, told in dynamic logic of propositional assignments. In Baral, C., Giacomo, G. D., and Eiter, T., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

[Herzig et al., 2014] Herzig, A., de Menezes, M. V., de Barros, L. N., and Wassermann, R. (2014). On the revision of planning tasks. In Schaub, T., Friedrich, G., and O'Sullivan, B., editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 435–440. IOS Press.

[Herzig et al., 2016] Herzig, A., Lorini, E., Maffre, F., and Schwarzentruber, F. (2016). Epistemic boolean games based on a logic of visibility and control. In Kambhampati, S., editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1116–1122. IJCAI/AAAI Press.

[Herzig et al., 2011] Herzig, A., Lorini, E., Moisan, F., and Troquard, N. (2011). A dynamic logic of normative systems. In Walsh, T., editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 228–233. IJCAI/AAAI.

[Herzig and Rifi, 1999] Herzig, A. and Rifi, O. (1999). Propositional belief base update and minimal change. *Artif. Intell.*, 115(1):107–138.

[Horrocks et al., 2006] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible SROIQ. In Doherty, P., Mylopoulos, J., and Welty, C. A., editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 57–67. AAAI Press.

[Jauregui, 2008] Jauregui, V. (2008). *Modalities, Conditionals and Nonmonotonic Reasoning*. PhD thesis, Department of Computer Science and Engineering, University of New South Wales.

[Kalyanpur et al., 2007] Kalyanpur, A., Parsia, B., Horridge, M., and Sirin, E. (2007). Finding all justifications of OWL DL entailments. In Aberer, K., Choi, K., Noy, N. F., Allemang, D., Lee, K., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer.

[Katsuno and Mendelzon, 1992] Katsuno, H. and Mendelzon, A. O. (1992). On the difference between updating a knowledge base and revising it. In Gärdenfors, P., editor, *Belief revision*, pages 183–203. Cambridge University Press. (preliminary version in Allen, J.A., Fikes, R., and Sandewall, E., eds., Principles of Knowledge Representation

and Reasoning: Proc. 2nd Int. Conf., pages 387–394. Morgan Kaufmann Publishers, 1991).

[Koutras et al., 2018] Koutras, C. D., Liaskos, K., Moyzes, C., and Rantsoudis, C. (2018). Default reasoning via topology and mathematical analysis: a preliminary report. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning, KR 2018, Tempe, Arizona, October 30-November 2, 2018*. To appear.

[Koutras et al., 2017a] Koutras, C. D., Moyzes, C., and Rantsoudis, C. (2017a). A reconstruction of default conditionals within epistemic logic. In Seffah, A., Penzenstadler, B., Alves, C., and Peng, X., editors, *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*, pages 977–982. ACM.

[Koutras et al., 2019] Koutras, C. D., Moyzes, C., and Rantsoudis, C. (2019). A reconstruction of default conditionals within epistemic logic. *Fundamenta Informaticae*. To appear.

[Koutras et al., 2017b] Koutras, C. D., Moyzes, C., and Zikos, Y. (2017b). A modal logic of knowledge, belief and estimation. *J. Log. Comput.*, 27(8):2303–2339.

[Koutras and Rantsoudis, 2015] Koutras, C. D. and Rantsoudis, C. (2015). In all, but finitely many, possible worlds: Model-theoretic investigations on 'overwhelming majority' default conditionals. In Destercke, S. and Denoeux, T., editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 13th European Conference, ECSQARU 2015, Compiègne, France, July 15-17, 2015. Proceedings*, volume 9161 of *Lecture Notes in Computer Science*, pages 117–126. Springer.

[Koutras and Rantsoudis, 2017] Koutras, C. D. and Rantsoudis, C. (2017). In all but finitely many possible worlds: Model-theoretic investigations on 'overwhelming majority' default conditionals. *Journal of Logic, Language and Information*, 26(2):109–141.

[Kraus et al., 1990] Kraus, S., Lehmann, D. J., and Magidor, M. (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artif. Intell.*, 44(1-2):167–207.

[Lamarre, 1991] Lamarre, P. (1991). S4 as the conditional logic of nonmonotonicity. In Allen, J. F., Fikes, R., and Sandewall, E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991.*, pages 357–367. Morgan Kaufmann.

[Lamarre and Shoham, 1994] Lamarre, P. and Shoham, Y. (1994). Knowledge, certainty, belief, and conditionalisation (abbreviated version). In Doyle, J., Sandewall, E., and Torasso, P., editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994.*, pages 415–424. Morgan Kaufmann.

[Lehmann and Magidor, 1992] Lehmann, D. J. and Magidor, M. (1992). What does a conditional knowledge base entail? *Artif. Intell.*, 55(1):1–60.

[Lembo et al., 2010] Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., and Savo, D. F. (2010). Inconsistency-tolerant semantics for description logics. In Hitzler, P. and Lukasiewicz, T., editors, *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, volume 6333 of *Lecture Notes in Computer Science*, pages 103–117. Springer.

[Lenzen, 1978] Lenzen, W. (1978). *Recent Work in Epistemic Logic*. North-Holland.

[Lewis, 1973] Lewis, D. (1973). *Counterfactuals*. Blackwell.

[Liu et al., 2011] Liu, H., Lutz, C., Milicic, M., and Wolter, F. (2011). Foundations of instance level updates in expressive description logics. *Artif. Intell.*, 175(18):2170–2197.

[Meyer et al., 2006] Meyer, T. A., Lee, K., Booth, R., and Pan, J. Z. (2006). Finding maximally satisfiable terminologies for the description logic ALC. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 269–274. AAAI Press.

[Motik et al., 2009] Motik, B., Horrocks, I., and Sattler, U. (2009). Bridging the gap between OWL and relational databases. *J. Web Sem.*, 7(2):74–89.

[Nute, 1980] Nute, D. (1980). *Topics in Conditional Logic*. Kluwer.

[Parsia et al., 2005] Parsia, B., Sirin, E., and Kalyanpur, A. (2005). Debugging OWL ontologies. In Ellis, A. and Hagino, T., editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 633–640. ACM.

[Peppas et al., 1996] Peppas, P., Nayak, A. C., Pagnucco, M., Foo, N. Y., Kwok, R. B. H., and Prokopenko, M. (1996). Revision vs. update: Taking a closer look. In Wahlster, W., editor, *12th European Conference on Artificial Intelligence, Budapest, Hungary, August 11-16, 1996, Proceedings*, pages 95–99. John Wiley and Sons, Chichester.

[Pratt, 1976] Pratt, V. R. (1976). Semantical considerations on floyd-hoare logic. In *17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25-27 October 1976*, pages 109–121. IEEE Computer Society.

[Rantsoudis et al., 2017] Rantsoudis, C., Feuillade, G., and Herzig, A. (2017). Repairing ABoxes through active integrity constraints. In Artale, A., Glimm, B., and Kontchakov, R., editors, *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017.*, volume 1879 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Rector and Rogers, 2006] Rector, A. L. and Rogers, J. (2006). Ontological and practical issues in using a description logic to represent medical concept systems: Experience from GALEN. In Barahona, P., Bry, F., Franconi, E., Henze, N., and Sattler, U., editors, *Reasoning Web, Second International Summer School 2006, Lisbon, Portugal, September 4-8, 2006, Tutorial Lectures*, volume 4126 of *Lecture Notes in Computer Science*, pages 197–231. Springer.

[Reiter, 1980] Reiter, R. (1980). A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132.

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95.

[Schild, 1991] Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In Mylopoulos, J. and Reiter, R., editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*, pages 466–471. Morgan Kaufmann.

[Schlechta, 1995] Schlechta, K. (1995). Defaults as generalized quantifiers. *J. Log. Comput.*, 5(4):473–494.

[Schlechta, 1997] Schlechta, K. (1997). Filters and partial orders. *Logic Journal of the IGPL*, 5(5):753–772.

[Schlobach and Cornet, 2003] Schlobach, S. and Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. In Gottlob, G. and Walsh, T., editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 355–362. Morgan Kaufmann.

[Schlobach et al., 2007] Schlobach, S., Huang, Z., Cornet, R., and van Harmelen, F. (2007). Debugging incoherent terminologies. *J. Autom. Reasoning*, 39(3):317–349.

[Segerberg, 1970] Segerberg, K. (1970). Modal logics with linear alternative relations. *Theoria*, 36(3):301–322.

[Stalnaker, 2006] Stalnaker, R. (2006). On logics of knowledge and belief. *Philosophical Studies*, 128(1):169–199.

[Stockmeyer and Meyer, 1973] Stockmeyer, L. J. and Meyer, A. R. (1973). Word problems requiring exponential time: Preliminary report. In Aho, A. V., Borodin, A., Constable, R. L., Floyd, R. W., Harrison, M. A., Karp, R. M., and Strong, H. R., editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM.

[Suntisrivaraporn et al., 2008] Suntisrivaraporn, B., Qi, G., Ji, Q., and Haase, P. (2008). A modularization-based approach to finding all justifications for OWL DL entailments. In Domingue, J. and Anutariya, C., editors, *The Semantic Web, 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. Proceedings*, volume 5367 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

[Tao et al., 2010] Tao, J., Sirin, E., Bao, J., and McGuinness, D. L. (2010). Integrity constraints in OWL. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

[Tiomkin and Makowsky, 1985] Tiomkin, M. L. and Makowsky, J. A. (1985). Propositional dynamic logic with local assignments. *Theor. Comput. Sci.*, 36:71–87.

[Troquard et al., 2018] Troquard, N., Confalonieri, R., Galliani, P., Peñaloza, R., Porello, D., and Kutz, O. (2018). Repairing ontologies via axiom weakening. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

[van Eijck, 2000] van Eijck, J. (2000). Making things happen. *Studia Logica*, 66(1):41–58.

[Winslett, 1988] Winslett, M. (1988). Reasoning about action using a possible models approach. In Shrobe, H. E., Mitchell, T. M., and Smith, R. G., editors, *Proceedings of the 7th National Conference on Artificial Intelligence, St. Paul, MN, USA, August 21-26, 1988.*, pages 89–93. AAAI Press / The MIT Press.

[Winslett, 1990] Winslett, M.-A. (1990). *Updating Logical Databases*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.